OCR GCE A COMPUTER SCIENCE PROJECT H446

@akhi.corporate

CONTENTS

Please Note that the page displayed in the contents table for the 'system diagram' is displayed faulty however the functionality to jump to the page is completely fine. This syntax error is due to the page break caused by the orientation of the page.

Share Copy link Download ... 5 Mark Grid - Matti K.pdf Info 5 / 11

Programming project (Component 03 or 04) marking criteria – 70 marks

AO 2.2 Analysis (maximum 10 marks)			
1–2 marks	**3–5 marks**	**6–8 marks**	**9–10 marks**
The candidate will have:			
• Identified some features that make the problem solvable by computational methods. • Identified suitable stakeholders for the project and described them and some of their requirements. • Identified some appropriate features to incorporate into their solution. • Identified some features of the proposed computational solution. • Identified some limitations of the proposed solution. • Identified some requirements for the solution. • Identified some success criteria for the proposed solution.	• Described the features that make the problem solvable by computational methods. • Identified suitable stakeholders for the project and described how they will make use of the proposed solution. • Researched the problem looking at existing solutions to similar problems identifying some appropriate features to incorporate into their solution. • Identified the essential features of the proposed computational solution. • Identified and described some limitations of the proposed solution. • Identified most requirements for the solution. • Identified some measurable success criteria for the proposed solution.	• Described the features that make the problem solvable by computational methods and why it is amenable to a computational approach. • Identified suitable stakeholders for the project and described them and how they will make use of the proposed solution and why it is appropriate to their needs. • Researched the problem in depth looking at existing solutions to similar problems identifying and describing suitable approaches based on this research. • Identified and described the essential features of the proposed computational solution. • Identified and explained any limitations of the proposed solution. • Specified the requirements for the solution including (as appropriate) any hardware and software requirements. • Identified measurable success criteria for the proposed solution.	• Described and justified the features that make the problem solvable by computational methods, explaining why it is amenable to a computational approach. • Identified suitable stakeholders for the project and described them explaining how they will make use of the proposed solution and why it is appropriate to their needs. • Researched the problem in depth looking at existing solutions to similar problems, identifying and justifying suitable approaches based on this research. • Identified the essential features of the proposed computational solution explaining these choices. • Identified and explained with justification any limitations of the proposed solution. • Specified and justified the requirements for the solution including (as appropriate) any hardware and software requirements. • Identified and justified measurable success criteria for the proposed solution.

0 marks = no response or no response worthy of credit.

ANALYSIS 10/10

This analysis addresses all the key points well. A range of computational methods are used to why the chosen solution is amenable to a computational approach.

All stakeholders are identified, and their use of the proposed solution explained. An in-depth research on similar solution is carried out explaining potential take outs.

Identified the essential features of the proposed computational solution explaining these choices. Furthermore, there is a good explanation and justification of possible limitations of the proposed solution.

The hardware, software requirements, and measurable success criteria for the proposed solution are identified and justified too.

AN OUTLINE OF THE PROBLEM:

For my project, I plan to make a bird's eye view 2D game, where the player takes on the role of a new interrogator, playing a series of distinct interrogations, their aim to follow the right course of questioning consistently, selecting one of three questions provided to them, to make the suspect "crack" and to win the interrogation. To be clear, each 'interview' will not pose 3 questions in total, rather in each question, three question options will be posed to the player to

choose from. The total number of questions per interview will be higher. The player's performance will determine the one of two endings obtained at the end of the game, a win or a loss.

For this to work, the right selection of one of the three questions will begin to fill a hot bar displayed on the screen. An average choice makes no change to the progression of the hot bar, and a bad choice takes away from the progress of the bar. If they hit a certain region of the bar from a series of good questions, the suspect in turn is incriminated.

STAKEHOLDERS

The nature of the program features a thought-provoking gameplay from the end-user, evident by the multiple-choice system, and the varying consequences of following the right and wrong pathway, causing the average age range of the target audience to likely be higher than the typical game. The simple point and click feature with a mouse allow for ease of play, not posing any complexity. The game might be best suited to mature individuals who can undergo some degree of critical thinking and problem-solving to progress through this story-based game. In fact, the point that the game is in of itself story-based, separates it far from the typical "boot and play" game, which is most appealing to younger audiences who may just be seeking a source of continuous enjoyment for short periods in time, seen by many games in the market. Hence from this I can safely assume, an older audience would most likely have more patience and discipline to advance in the game. Lastly, the "cases" presented in the game do depict crime, and a game of such is evidently something which most parents would restrict younger children from playing.

Due to the previous factors mentioned, the game would be most appropriate for a target audience ranged from 16 and above.

HOW THE PROBLEM CAN BE SOLVED BY COMPUTATIONAL METHODS

The numerous computational methods can easily be implemented in development of the game. These are explained further below.

THINKING ABSTRACTLY

By abstraction, I can remove unnecessary detail from the game and focus on the relevant, important elements of the game, through a simplified representation of reality:

- Present a two-dimensional game rather than a three-dimensional reality-similar game
- Represent the characters and objects in the game with sprites – image assets created and imported elsewhere rather than humanoid and reality structures
- Remove all unnecessary sounds, and import necessary sound effects to add to game ambience and gameplay in general

- Adding a hot bar on the screen for the user to be able to visually see the progress they are making in the interrogation, simplifying the game greatly
- Restricting the user to pose three questions to the suspect rather than asking whatever they desire, giving the game and user more direction and making the general interrogation process simpler.
- Details of suspect and case details are restricted to no more than a couple short paragraphs, adverse to the lengthy documents and files which would be the case in reality.

THINKING AHEAD

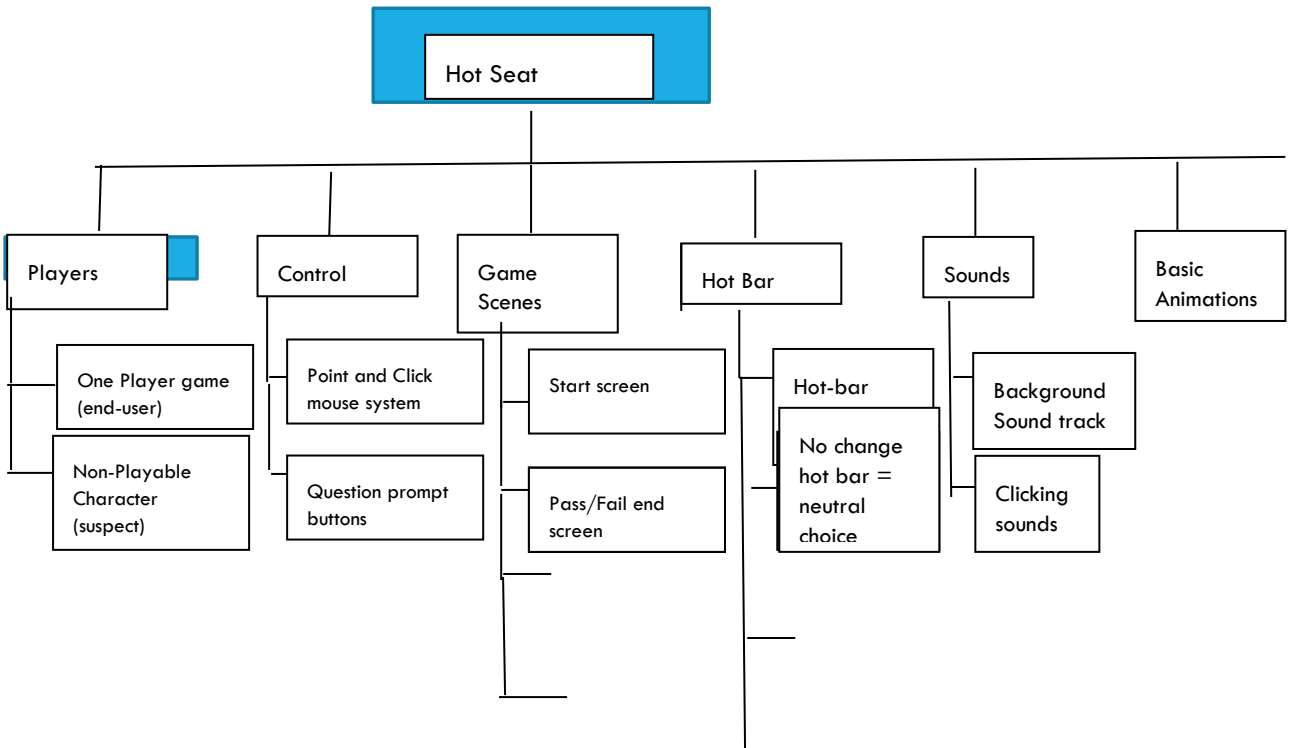
By thinking ahead, I can ensure thorough pre-planning of my code to obtain an efficient and desired outcome. Elements may also be highlighting inputs and outputs, preconditions, reusable program components and caching if needed.

- Inputs from the user would be through 'point and clicking' features of the game such as menu buttons, pause buttons, question boxes etc.
- Outputs of the game would consist of replies in the form of text boxes from the non-playable character within the game, sounds, basic animations and the hot bar filling/remaining neutral/ decreasing

THINKING PROCEDURALLY

By thinking procedurally, I can identify the individual parts of a bigger problem. A large complex task which would represent the game itself can be broken down into smaller sub-tasks, until I am left with a sub task that returns a single function. I can represent this through a structure diagram, using a method known as stepwise refinement.

Main problems: Players, Control, Game Scenes, Hot-bar, Sounds, Basic Animations



Players: How will the game be played by the user? The game will feature a single-player game. The player controls their single character, the interviewer, interacting with a non-playable character, the suspect.

Control: How does the user interact and progress the game? This will be done by utilising a simple input device – a mouse, to start the game from the game's startup screen, click question prompt buttons to carry out the role of the interviewer, and to enter the game menu.

Game scene: How will the game be structured? The user will always be met with a start-up screen upon execution of the program. From here, the user can select the play button when ready to begin playing. Actual gameplay will take place in a pixel-art interrogation room. The user can enter the menu at any point in the game. The ending of the game will either display a 'pass' or 'fail' screen dependent on the success rate of the player's interviews, which must be above a set threshold.

Hot bar: How will the player have a measure of their success whilst playing? The hot bar is an important feature to visually display to the user how well or bad they are doing, and how close they are to making the suspect crack to win the interrogation. It's augmentation and depletion feature give a good measure to the player.

Sounds: Sounds will be added to allow for immersion into the game, and to emphasise the setting the player is in.

Basic animations: Subtle basic animations will be added during the game.

Thinking Logically

By thinking logically, I can identify points in the game where a decision would need to be taken and how they would impact the algorithm and the route it would consequently take.

A decision would be initially taken at the beginning of the game, running the game only iF the player selects the play option.

Decisions would be also be taken in the menu, such as to remove music and sound effects, etc.

Decisions would need to be constantly taken when the player is prompted to select one of three questions to ask the suspect. From their selection, the program can branch differently. For example, if they make a series of good choices, the hot bar will reach/surpass the limit, resulting in a successful interrogation. This can be applied vice versa.

THINKING CONCURRENTLY:

By thinking concurrently, parts of the game can be developed to take place at the same time, making the solution more efficient. Examples would include the continuing background music whilst the user plays the game. The hot bar progressing at the same time the user selects question prompts.

LIMITATIONS OF THE DEVELOPMENT

There have been some limitations considered to the development of my game. Listed below:

- Initially, a feature that was proposed but later dropped was a wide range of various 'suspects' that the player can interview within the game. Due to the target audience considered, and the snappy and quick nature of gameplay, this would ultimately drag out the total gameplay of the game, which defeats its purpose. This may add a tedious tone to the game as the end user would have to play for a longer overall duration to reach an ending. Hence, a solution proposed was to limit the number of suspects and therefore playable cases to around 3.

- Another limitation to the development of the program was the number of endings the end user can obtain. By increasing this further than 2, a greater complexity would be added to the programming of the game, due to the addition of more layers which must be coded. In terms of the end user in general for games with multiple endings, they may seek the thrill in trying to obtain the different endings, however as mentioned prior, a younger audience would likely undergo a playthrough for a game a numbered amount of times, hence it is not convenient to consider more than 2 endings.

- Another limitation was the number of prompts which would be available to the end user playing the game when selecting questions. Initially it was thought to give the player a wider range of questions to select from, however this would in turn increase the difficulty of the game. The aim of the game is for it to be as simple and hence accessible as possible to a large population of end users. The purpose of the game isn't necessarily to challenge the intelligence of the user so much by

assessing if they pick a correct option, but rather see if they are consistent and accurate in selecting a good streak of questions they pose, in order to make the suspect 'crack.'

- Finally, the limitations listed above (if included) would ultimately increase memory usage. The game in itself is meant to be accessible from as many different devices with varying specs, as it is targeted as a low-end game, not too demanding in performance.

IDENTIFY SOME REQUIREMENTS FOR THE SOLUTION

In order to fully develop this game as planned, it will be required from me to choose a suitable programming language. In my case, I will be choosing a fairly simple programming language with readable syntax, this being python. In addition, I will need to also create the scene settings featured in the game manually. This can be done with a third-party application which allows me to paint/draw, and then importing this into my game. When undergoing testing near to the end of development, I will require a student from my class to pose as the end-user, playing the game from start to finish and following as many possible routes as they can to ensure the game is fully functional. My tools of research throughout this demanding project will occur via search engines, such as Google.

Now, in terms of the end user requirements, their system would have to (as a minimum) be able to run low-end to mid-end games. The game within itself is developed in such a way that a complex, high performance end system is not required.

RESEARCH

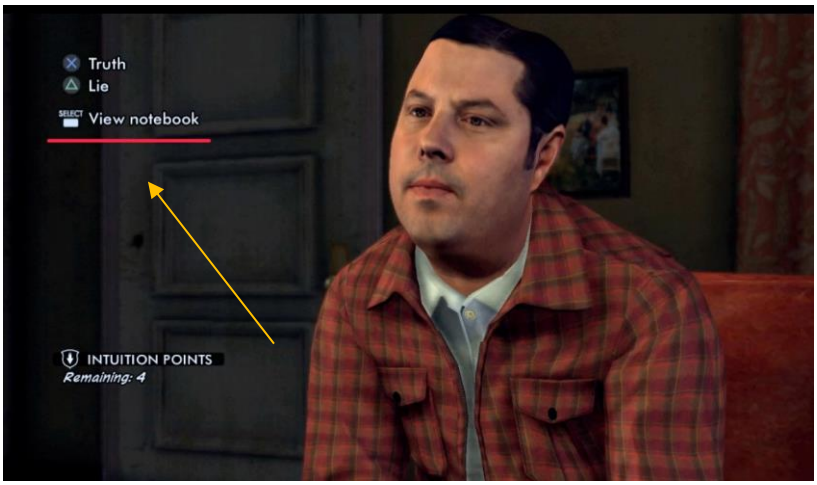
GAME COMPARISON: "EXISTING SOLUTION"

For this section of my research, I decided to pick a game which as closely resembles my current project, even though the differences between the games are relatively distinct. This game has undergone much time in development with most likely a highly skilled team of developers, with a different purpose/storyline. An aspect of the game similar to mine would be the game interrogations. The game I picked was L.A. Noire.

L.A. Noire is a 3rd person detective game which allows the player to play a number of different cases throughout the game, whilst simultaneously progressing through their career as a police officer working in L.A, set in the year 1947. Most notably, when focussing on the interrogation gameplay of the game, we see that L.A Noire incorporates advanced mechanics to increase complexity and make them a real challenge. Let's take a closer look at any similarities and differences:

To begin with, we start by acknowledging how L.A Noire utilises subtle facial cues and a suspect's body language when an interrogation is conducted to encourage the player to work out if they are telling the truth or not. This can be regarded as a complex mechanic due to the animations involved. On the other hand, this is quite demanding from the player as reading a person is seen as a skill which many players are unlikely to have. In my game, the most suitable course of action for the player to take in the interrogation levels would be achieved through reading the case details as best beforehand, unlike what is seen here for the reference game.

In addition, it can be seen here how the game also makes use of a three-option choice system, much similar to a feature I intend to incorporate in my game, encouraging the player to make use of their initiative and knowledge on the case to choose the most appropriate option. The game also allows for different input devices, such as keyboards or controllers, whereas in my game only a keyboard + mouse input will be registered.



Furthermore, L.A Noire allows the player to "view notebook," essentially providing them access to precise details pertaining to the case details and the suspect themselves. By viewing and reading over this throughout the interrogation, the player can refresh their memory and more importantly, ask the correct questions.

It is also worth noting of the number of different characters present in the game. In comparison, my game will only feature two characters per level, and one will always be the interrogator/the player themselves.

Things I like from the game:

Option system: I like how the player steers several different possible outcomes in the interrogation by the individual choices they make at each point in the interrogation. Ultimately, the outcome would be success or failure, dependent on the player's own precision beforehand. This is exactly a feature I will be implementing in my game, the three choices proposed in the form of three different questions rather than 'truth,' 'doubt,' and 'lie.'

Graphics: The graphics used in L.A. Noire effectively capture real life realism as seen in the attention to detail of the various scenes and most definitely the characters featured in them as shown by the pictures above. However, implementing such realism in my game would defeat its simplistic purpose in the end. It would also demand a higher performance system and on top of that, lead to time constraints.

Camera: Although my game will also take on a third person view, here the game utilises an up-close shot on characters to immerse the user more into the gameplay. This simultaneously compliments the 3D models in the game. It wouldn't make sense for me to implement the camera angle used in L.A. Noire due to the 2D nature of my game.

Things I dislike from the game:

Complexity of the interrogations: The complexity of the game is emphasised through how the player must for example observe facial features and movement in the suspect to deduce what approach to take as a result. This, in itself, largely constricts the target audience age on the lower end, as they would most likely lack the ability to exercise such skill, which would make the game seem more tedious and less enjoyable and entertaining.

View notebook feature: The view notebook feature gives the player too much liberty throughout the interrogation to keep reviewing the case, which, if implemented in my simpler game, would make posing the most logical questions way too easy. However, it is understandable to see why L.A Noire has implemented this in their game to balance out the difficulty.

Number of interrogations: Numerous interrogations take place throughout the game, due to the fact that the game is more story-based. I would feature a much smaller number of interrogations to retain player's attention and allow them to reach an ending in one play-through in minimal time.

Extended LA Noire Research:

LA Noire, as highlighted before, makes use of high detail facial motion capture technology which was developed exclusively for the game. Not only facial expressions, but cues or tics that may be seen in the suspect during questioning can be picked up on by an analytical player. Along with this, the player has access to a handbook at any point in the interrogation which they can go through for clues and evidences they have collected at prior points in the game.

Comparison Point:

In my heavily simplified game, the player can also choose from three question choices, however these are based on a ranking system which have a direct impact on a hot bar progression.

The only way a player can gauge the best question to select in my game to progress the hot bar and win the interrogation is to ensure they have a clear understanding of the case by reading the details beforehand and using their logic heavily.

In total, regarding interrogations, there are 236 questions in the game.

Comparison Point:

My game only features a maximum of 15 questions.

Each question can be ranked as: Truth, Doubt, Lie

The player poses 'truth' if they believe the suspect is telling the truth.

The player poses 'lie' if they believe that the suspect is lying, and they have evidence to back it up.

The player poses 'doubt' if they believe the suspect is withholding information based on something they have mentioned or has said a statement slightly wrong.

Comparison Point:

In my game each question can be ranked as: Excellent, Neutral or Poor

An excellent question builds towards a level win and is a logically smart question ask relevant to details.

A neutral question doesn't really have any significant impact in the interrogation and may be seen as a general question.

A poor question is a logically foolish question to ask relevant to case details and allows the suspect to dig themselves out of the trouble they have landed themselves in!

Closing Comments: The game LA Noire compared to mine is much more complex and in depth. It favours a storyline approach, the interrogation aspect of the game only making up a portion of it. My game is based entirely on the interrogation aspect, and the mechanics within the interrogation is simplified, encouraging player logic.

INTERVIEW CONDUCTED WITH FARDIS RAJABI TO IDENTIFY USER REQUIREMENTS BETTER, AND POSSIBLY MAKE AMENDMENTS/INTRODUCE NEW FEATURES AFTER CONCLUDING AND GOING OVER END-USER OPINIONS.

Note: After brief communication with Fardis, I acknowledged that he had never played an interrogation style game before. Hence, to make up for this, I have explained the game thoroughly in its current proposed state now.

Me | Fardis

How challenging would you want an interrogation game to be?

High-medium intensity. Not too intense, but at a level where I would be surprised and have to think.

What would keep you playing, and allow you to finish the game in one sitting?

If the questions are not repetitive, otherwise I would get bored quickly. The user interface is also important.

How important are visuals? Is a highly realistic game which replicates reality closely more appealing to you?

I don't think a game which closely matches reality is always important. I would say something normal and not too flashy.

What are some subtle aspects of such a type of game which you appreciate when you notice?

Attention to detail to objects in the game. I like to focus on the small details in a game.

How important are sounds in a game? List some types of sound effects or such that you would like to hear throughout the game.

Good sound quality is important along with the sounds themselves in a game. I would want sound effects such as walking sound effects and ominous sounds, so I feel more spooked.

How long would you want the playthrough to be? Do you prefer longer or shorter sittings for games which are meant to be played in one sitting?

A perfect game to be played in one sitting in my opinion would be about 30 minutes to an hour.

What do you think about having the choice to propose multiple questions + the hot bar feature alongside it?

I would enjoy the game much more, and you could try play the game to achieve different endings if possible. The hot bar feature would keep me engaged and make the game much more enticing.

Lastly, how could I make my game as fun and immersive as possible to play?

The game could include some scary creative features such as displaying the user's IP address. In general, making the game more scarier would be fun. Too much reading would also be boring.

FEATURES OF THE PROPOSED SOLUTION: FEATURING THE MAIN PROPOSED FEATURES FOR THE GAME "HOT SEAT"

Hot bar	**The hot bar acts as an indicator throughout the interrogation to inform the player of their performance and how close they are to winning the interrogation**
Sound effects	Basic sound effects will be added for immersion. Examples may include (and are not limited to) pop-up sound effects, conversation sound effects etc.
A key to pause the game	A specific key bind pre-defined will be used to allow the user to momentarily pause the game at any moment.
Mouse control	Allows the user to use a "point and click" system to select buttons on the screen, such as questions in the interrogation gameplay.
Win/Lose ending screen	Two different endings which can be obtained depending on the success rate of the player with the interviews. This must be above a threshold limit to allow them to obtain a win.
Pixel art	Scenes displayed in the game along with the different suspects being interviewed along with the different screens

	will be created by pixel art. This ensures simplicity.
Question choices	Three different questions. Each ranked in a random order from a negative, to neutral, to positive choice selection which has direct impact on hot bar.
Menu screen	A basic menu screen allowing a player to Start and Exit a game and possibly toggle sound effects / in-game music on and off.
Minimum of 3-4 interrogation scenarios	This promotes the short and snappy and attention-grabbing aspect of the game, rather than being more long winded.
Description of interrogations beforehand starting	Displays all case and suspect details relevant to provide the player with enough info to select the right course of questions
Single-player game	The game will only be played by one person as it is deemed most suitable in this manner. Furthermore, it lies beyond my ability to program the game to facilitate for more than one player playing at the same time.
Pixel-like graphics	Less emphasis on realism but more on abstraction. More welcoming to a younger audience which is a large part of the target audience.

HARDWARE AND SOFTWARE REQUIREMENTS:

Hardware	Justification
Monitor	Allows the user to observe the visuals of the game and therefore be able to interact with the game

Mouse	Used to navigate through the menu and pause screen + select questions when playing.
Keyboard	Used to pause the game at any moment.
Speakers/headphones	A way of audio output, allowing the user to hear the sound effects and music in game.
Processor of speed 2Ghz +	This is the average speed of a processor required for low end game titles.
Memory 2GB + RAM	This is generally the amount of RAM required for less demanding games. The game will feature a limited number of sprites, scenes etc, hence could probably even need less RAM.
Integrated graphics	Minimum graphics will be required to run the game.

Software	Justification
Operating systems: windows, MacOS, Linux	Game is developed with python, and python is compatible with such major titles.
Python interpreter	Allows python code to execute on a system.
Game specific libraries	Game may utilise additional libraries which users may have to install on their systems to run the game properly.
Run-time environment	A place to run the game.
Processor of speed 2Ghz +	This is the average speed of a processor required for low end game titles.

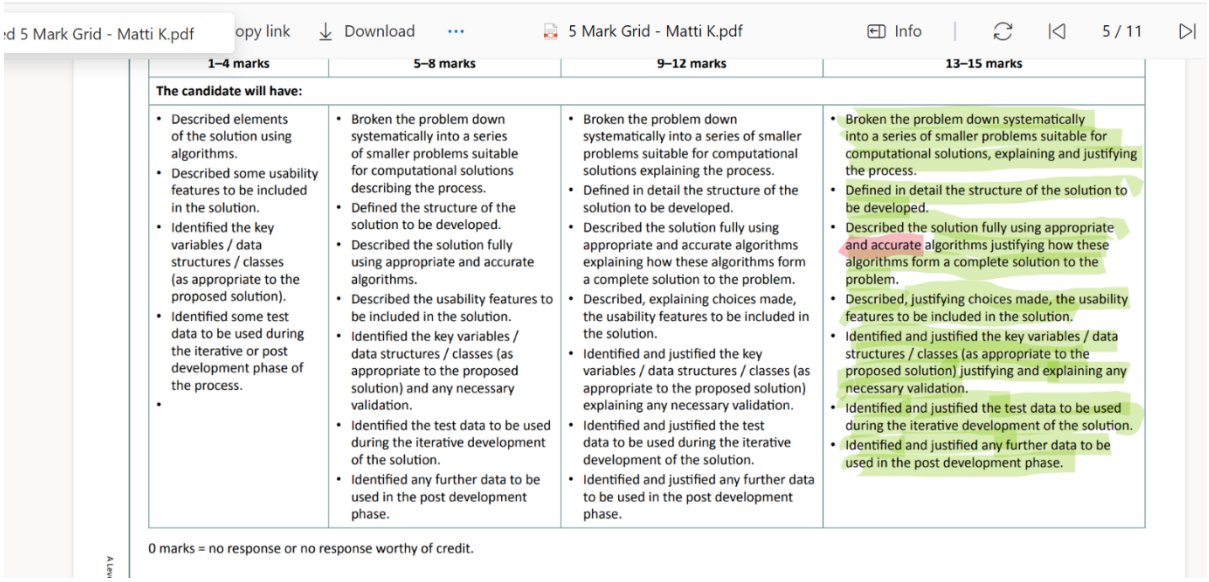
SUCCESS CRITERIA FOR "HOT SEAT."

Requirement	**Justification**

Minimum of four interrogation levels to be created, featuring different 'suspects' and their respective cases.	Gives the player more opportunities to achieve a win ending as they have a greater number of tries. Also makes the game more interesting without making it too time consuming by having double digit levels for example.
3rd person sky view	Seems most applicable to use as the game is two dimensional.
A starting menu with a minimum of three options	The three options that will be featured on a minimalistic scale will be the 'play' button to let the player start the game, the 'exit' button to quit the application and a settings button to access the settings menu so the player can modify the game to their likes.
Simple settings menu	Gives the player the freedom to toggle on and off the sound effects in the game and the music in the game.
Sound effects	Simple sound effects will be downloaded into files from external sources and added into the game, which will be used at points throughout the game to immerse the player slightly more. Will also be used when the player is clicking with their mouse.
Background music	A minimum of two different separate tracks will play in the background. The initial track will play on the menu screen when the game is first loaded up. The following track will play for the duration the game is played. May also use another two tracks for a win ending and a lose ending.
A dark theme implemented for the game.	To allude to the actual role of dealing with potential criminals and the nature of crime itself, a darker, more sinister theme, may be the most fitting for the player playing the role of the interrogator, in contrary to a bright and cheery atmosphere.
The interrogation room scene + winning scene + losing scene creation	These individual scenes must be created digitally through art in an external application and then transferred over to the

	game. This is an essential layer to be added to any game, as without a setting, the game would be very dull.
Hot bar mechanism	A visual hot bar placed on the side of the screen for the player to see in the interrogations which fills and depletes and remains neutral, in respect to the three types of questions that can be prompted by the player.
Fictional cases decided for fictional suspects in the game.	Must create fictional cases for suspects portrayed in the game. This will be displayed to the player and not be too long or difficult of a read. Allows the player to gain an understanding of the suspect and hence the best track of questions to ask to win the interrogation.
Question prompts	Deciding on what questions to provide the player for them to be able to choose from in the interrogations. This is an essential aspect of the game, as the game revolves around this multi-choice system.
Simple calculation system to calculate the one of the two endings the player obtains.	If only 4 interrogation scenarios are developed for the game, the player must have at least a 75 percent success rate, meaning they incriminate at a minimum of ¾ suspects.
Pause menu displayed through clicking 'm' character on the keyboard, and way of resuming game from this point.	If the 'm' button is clicked, a pause menu is displayed whilst the game is being played. This displays the elements of the settings menu + an exit button to leave the game. This is useful as the player can halt the game as needed and return to it. The player can then resume the game by clicking the enter button.
Creating a fixed character sprite for the interrogator (the role which the player plays) and different character sprites for the suspects in the different levels.	Allows the characters in the game to be visually displayed to the player in a simplified manner.

DESIGN SECTION



ed 5 Mark Grid - Matti K.pdf opy link Download ... 5 Mark Grid - Matti K.pdf Info 5 / 11

1–4 marks	5–8 marks	9–12 marks	13–15 marks
The candidate will have:			
• Described elements of the solution using algorithms. • Described some usability features to be included in the solution. • Identified the key variables / data structures / classes (as appropriate to the proposed solution). • Identified some test data to be used during the iterative or post development phase of the process. •	• Broken the problem down systematically into a series of smaller problems suitable for computational solutions describing the process. • Defined the structure of the solution to be developed. • Described the solution fully using appropriate and accurate algorithms. • Described the usability features to be included in the solution. • Identified the key variables / data structures / classes (as appropriate to the proposed solution) and any necessary validation. • Identified the test data to be used during the iterative development of the solution. • Identified any further data to be used in the post development phase.	• Broken the problem down systematically into a series of smaller problems suitable for computational solutions explaining the process. • Defined in detail the structure of the solution to be developed. • Described the solution fully using appropriate and accurate algorithms explaining how these algorithms form a complete solution to the problem. • Described, explaining choices made, the usability features to be included in the solution. • Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) explaining any necessary validation. • Identified and justified the test data to be used during the iterative development of the solution. • Identified and justified any further data to be used in the post development phase.	• Broken the problem down systematically into a series of smaller problems suitable for computational solutions, explaining and justifying the process. • Defined in detail the structure of the solution to be developed. • Described the solution fully using appropriate and accurate algorithms justifying how these algorithms form a complete solution to the problem. • Described, justifying choices made, the usability features to be included in the solution. • Identified and justified the key variables / data structures / classes (as appropriate to the proposed solution) justifying and explaining any necessary validation. • Identified and justified the test data to be used during the iterative development of the solution. • Identified and justified any further data to be used in the post development phase.

0 marks = no response or no response worthy of credit.

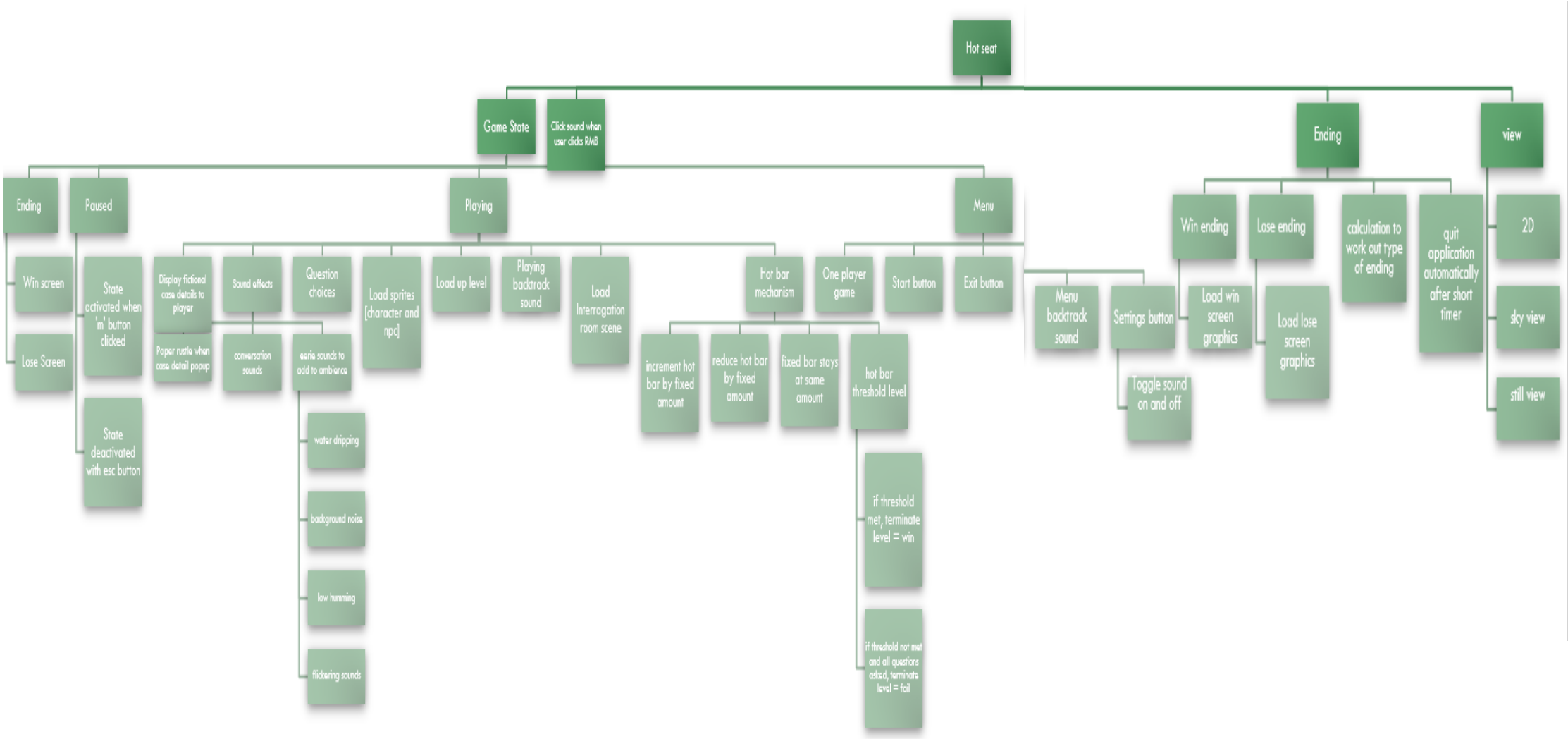
13/15

To warrant the marks awarded, you completed the allocated the following task:

- Breaking the problem down systematically into a series of smaller problems suitable for computational solutions, explaining and justifying the process.
- Defining in detail the structure of the solution to be developed.
- Describing the solution fully using appropriate algorithms, justifying how the algorithms form a complete solution in the final link algorithm.
- A clear description and justification of why the chosen usability features were included in the solution.
- The key variables / data structures were identified and described. In the iterative tests, some validations were included; post development test data was present too.
- However not all algorithms appear to be accurate (e.g. soundtrack) and furthermore, algorithms suggested in the final link algorithm are not present.

SYSTEMS DIAGRAM

The game is now be broken down into a systems diagram. The overall problem can be broken down into manageable sections and smaller processes which will need to be individually designed and then integrated together at the end to make up the solution.



Detailed explanations of different components

'Click sound when user click RMB' component: This component is a simple but satisfying addition to the game, playing a sound in response to whenever the user clicks the right mouse button on their mouse anywhere on the game screen.

'Game state' component contains many subcomponents, which feature every state the game could possibly be in from execution to termination. This would initially be the landing state the user would be forced to find themselves in when executing the game, this being the 'menu' state, which features the different buttons the user can interact with, this being to start, exit or display settings. The transitioning state would then go to the 'playing' state if the user interacts with the play button. At any time, the user may pause the game by clicking the 'm' button. One of two endings can be obtained, this being a win screen state or a lose screen state. I have also listed several subtle features which will be included in each state, this being soundtracks and sound effects for example to increase game immersion. I have also identified elements of each state, this being most concentrated in the 'playing' state, such as loading sprites, question choices etc. A very important aspect of this state which carries the game would be the hot bar mechanism I have identified. It's progression or depletion is directly affected by user input. Excellent choice questions posed by the player fill the bar, neutral questions which do not make much of a difference have no effect on the hot bar, and poor questions which harm the interrogation will deplete it.

'Ending' component is determined by a subroutine which can perform a simple logical operation to determine which of the two endings should be delivered to the user. For example, winning 2/3 levels would deliver a win screen. There is also a time limit as to which the screen is displayed for increased memory efficiency, where the application would be automatically terminated if the user manually chooses not to. My personal designs made for each respective screen will also be shown.

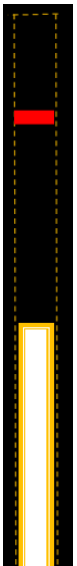
The 'view' component identifies how the game is visually displayed to the user. As we can see, the game is two dimensional, therefore flat. This was chosen due to time constraints and to ensure complexity is not too high. The view the player sees the game is from above, hence identified with 'sky view.' In addition, this camera view does not move and remains fixed, hence shown by 'still view.'

PROPOSED DESIGNS FOR THE GAME

Main Menu Design:

Highly simplified menu screen. No emphasis on complexity here as this is simply the start screen observed by the user when the game loads up. The more sinister/ominous theme is observed, with the use of darker colours, feeding into the game ambience before even beginning the game. The use of pixel art compliments the overall art design that the game will take.

Hot Bar Feature:



A simple hot bar, which fills up progressively in relation to the type of question choice proposed by the player, and the threshold limit indicated close to the top which represents a player has succeeded in the interrogation.

Win Screen Design:

Although the theme here is slightly dark, the contrast with the lighter white/grey colours links with the concept of winning here. The sprites displayed with their heads down aims to convey the message that the player has beaten the fictional suspects in the game.

Lose Screen Design:

The lose screen takes on an eerie design, emphasised with the hand blood print. This links to the reality of interrogators failing to incriminate genuine suspects who could be let out into the real world again and commit more crime. The importance of the role of putting criminals into jail is emphasised here, and the risk of not doing so.

SUMMARY OF THE PROCESSES

Start-up

- Initialise application when executed:
- Load up game
- Display main menu to user

- Menu:
- If user clicks on settings, then load settings menu.
- If user clicks on play then initialise the game. Game state = playing.
- If user clicks on exit then quit the application.

In-Game

- Initialise:
- Load up level design
- Load backing track sound
- Load up the sprites
- Load case details before displaying the setting
- Load question options
- Display hot bar

- Hot bar:
- If user selects excellent choice of question, then hot bar filled by a set increment
- If user selects neutral choice of question, then hot bar filled by no increment.

- If user selects a poor choice of question, then hot bar decreases by a set increment.

■ Interrogation outcome:
- If user is able to reach the hot bar threshold before all questions are posed by them, then the interrogation classes as a win.
- If the user is unable to reach the hot bar threshold before all questions are posed by them, then the interrogation classes as a fail.

■ Win and Lose ending:
- If interrogation success rate is greater than decided percentage, then display win screen ending.
- If interrogation success rate falls short than the decided percentage, then display lose screen ending.

Controls

■ Navigation:
- Point and click system with the use of a cursor alongside a clicking sound when LMB is pressed.

■ Pausing:
- Game can be paused during playing state with the 'm' button and resumed with the 'esc' button.

Sound effects:

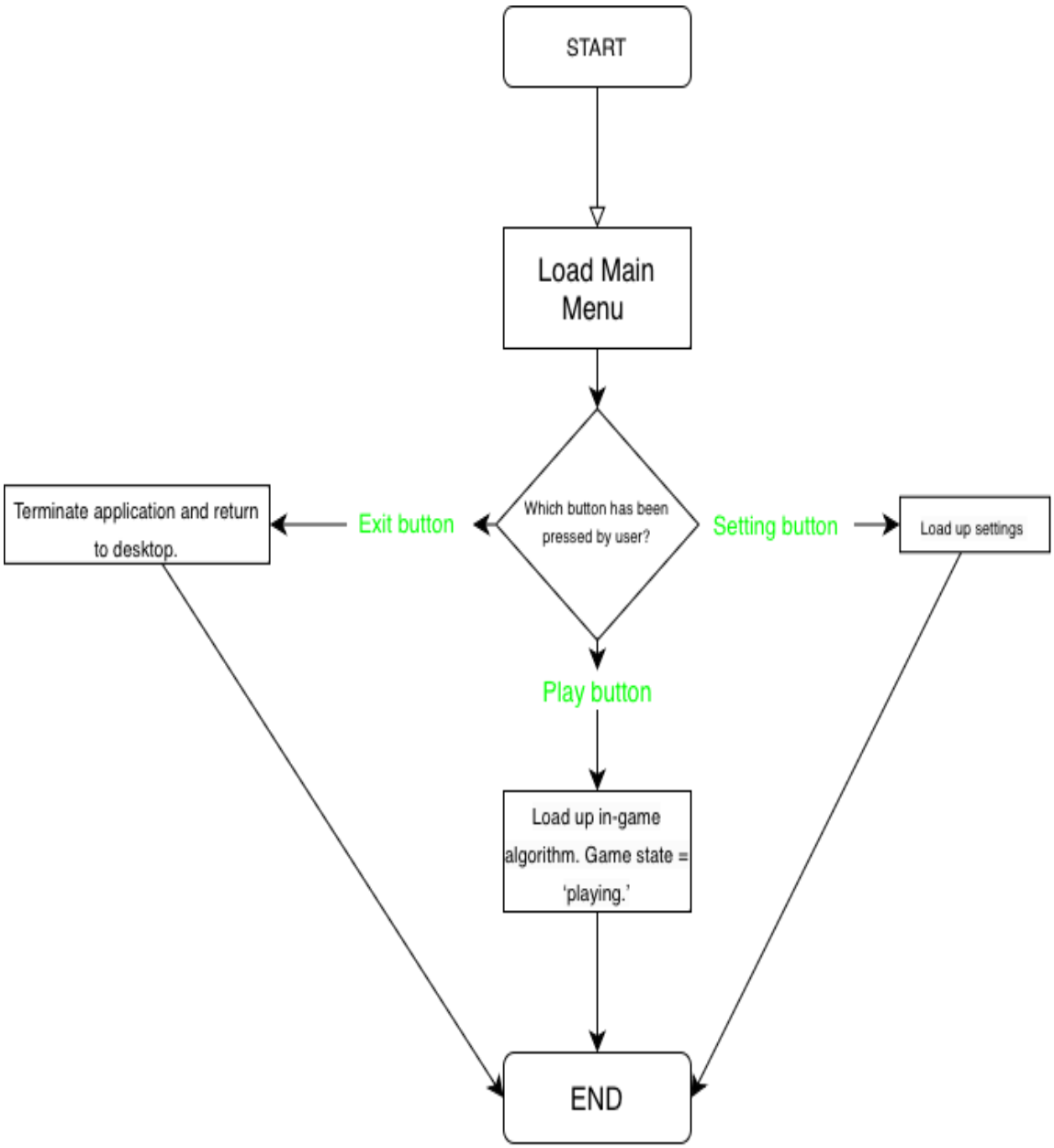
■ Main Menu:
- If user is in the main menu, then play menu backtrack sound

- Playing state:
- Ominous background sounds played at regular intervals.
- Conversational sounds to mimic dialogue between the player and npc.
- Backtrack sound

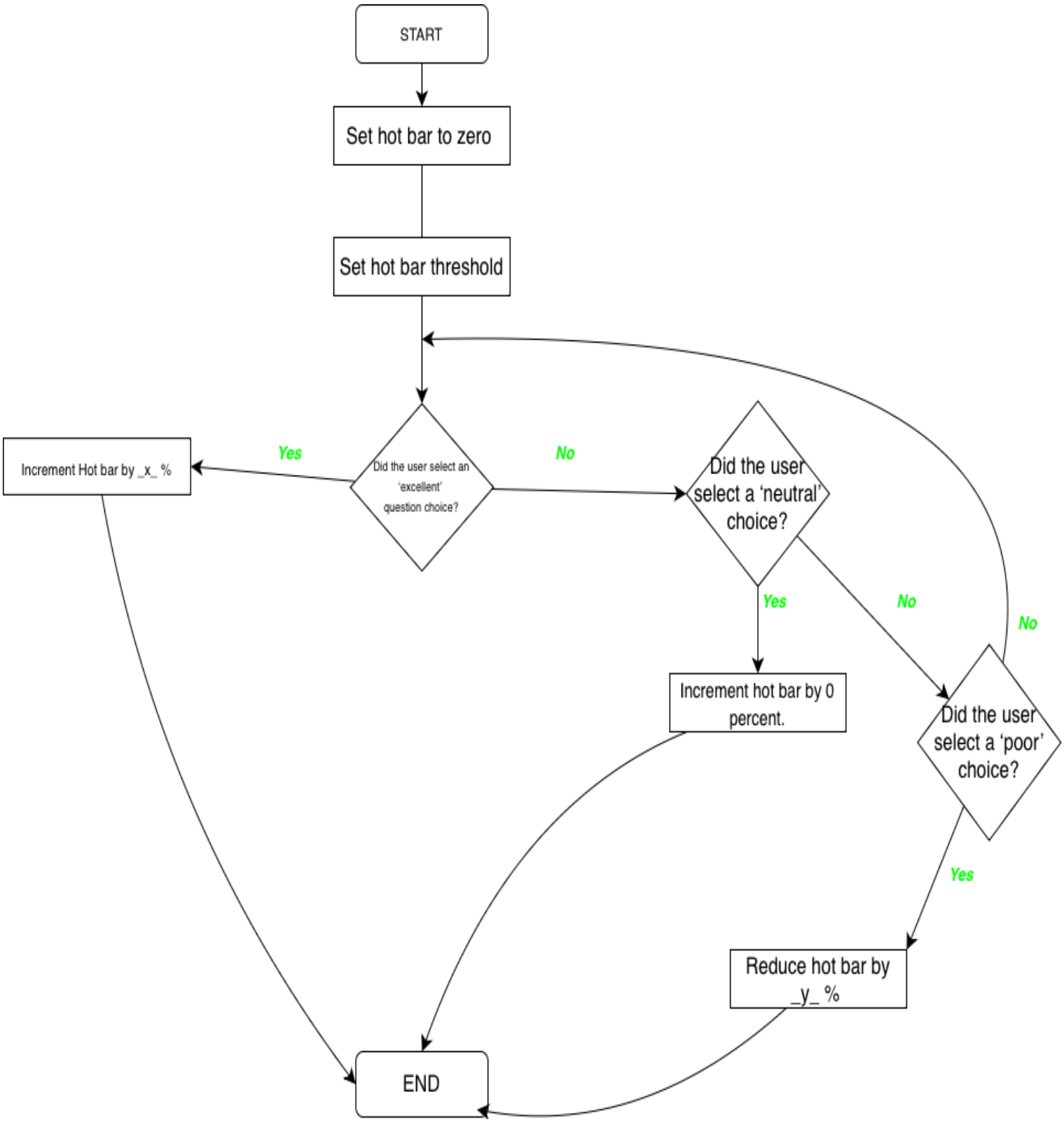
- Ending:
- Victory like music if ending = win
- Loss like music if ending = loss.

ALGORITHMS

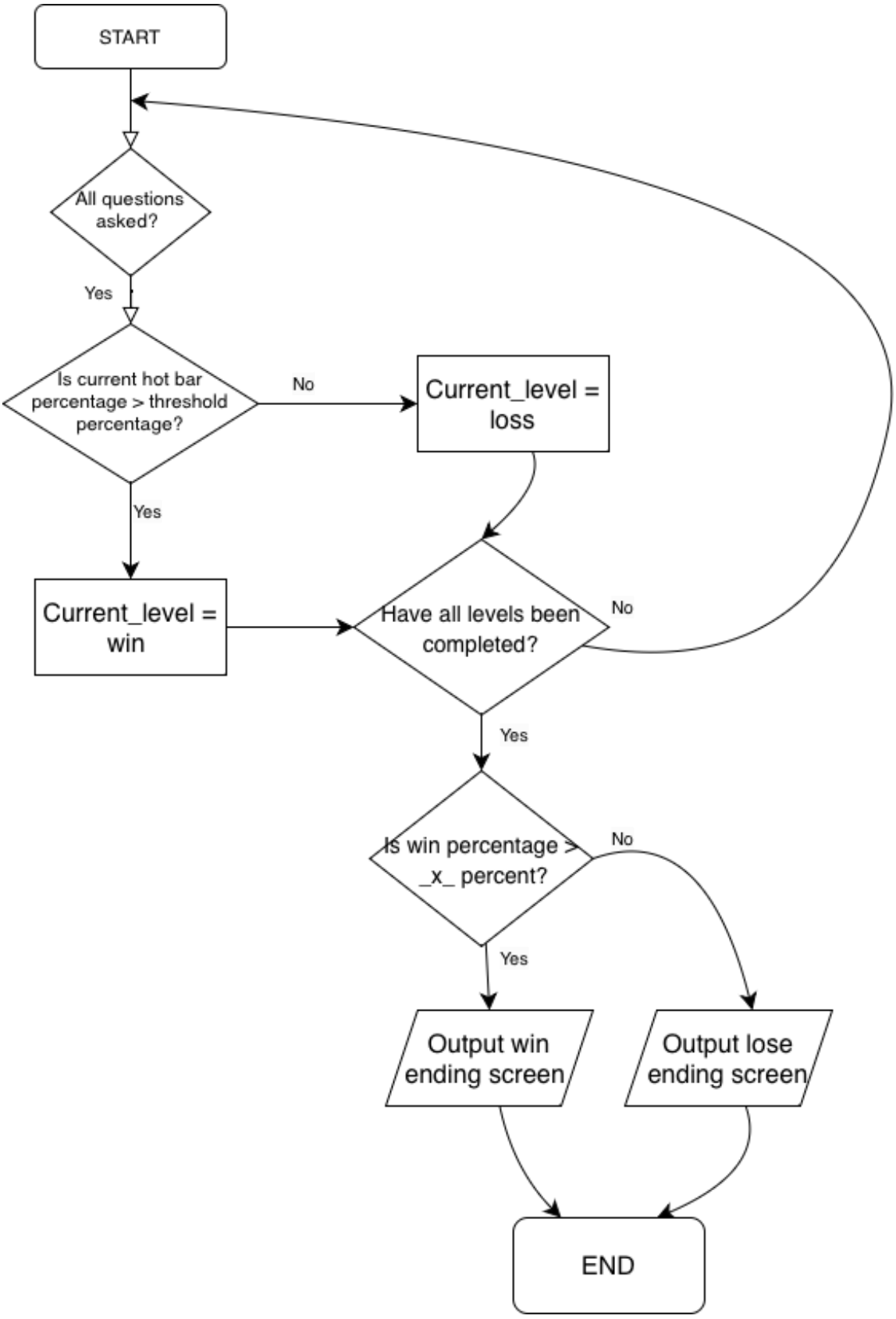
Main Menu System:



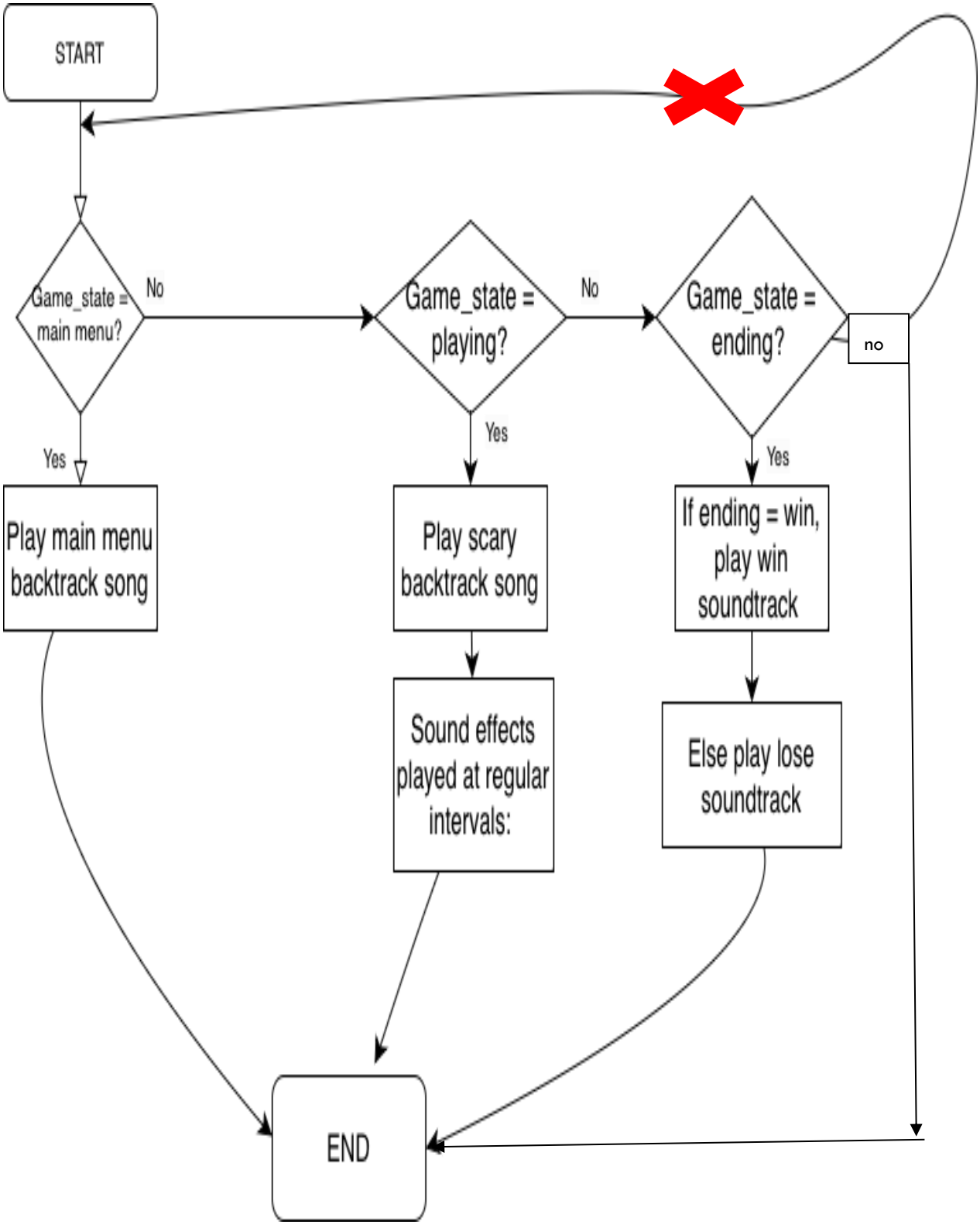
Hot Bar System:



Outcome / ending system:

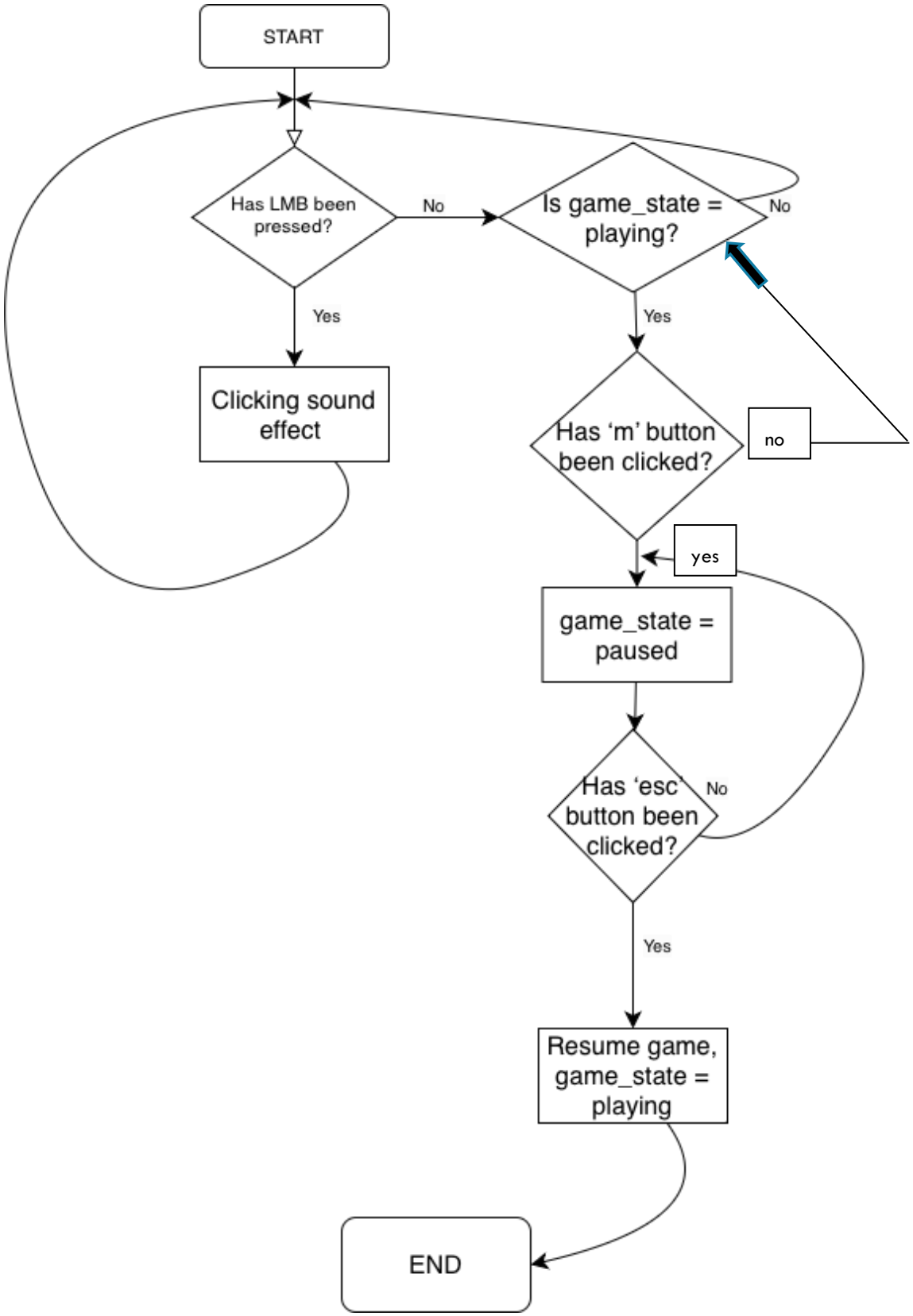


Soundtrack System

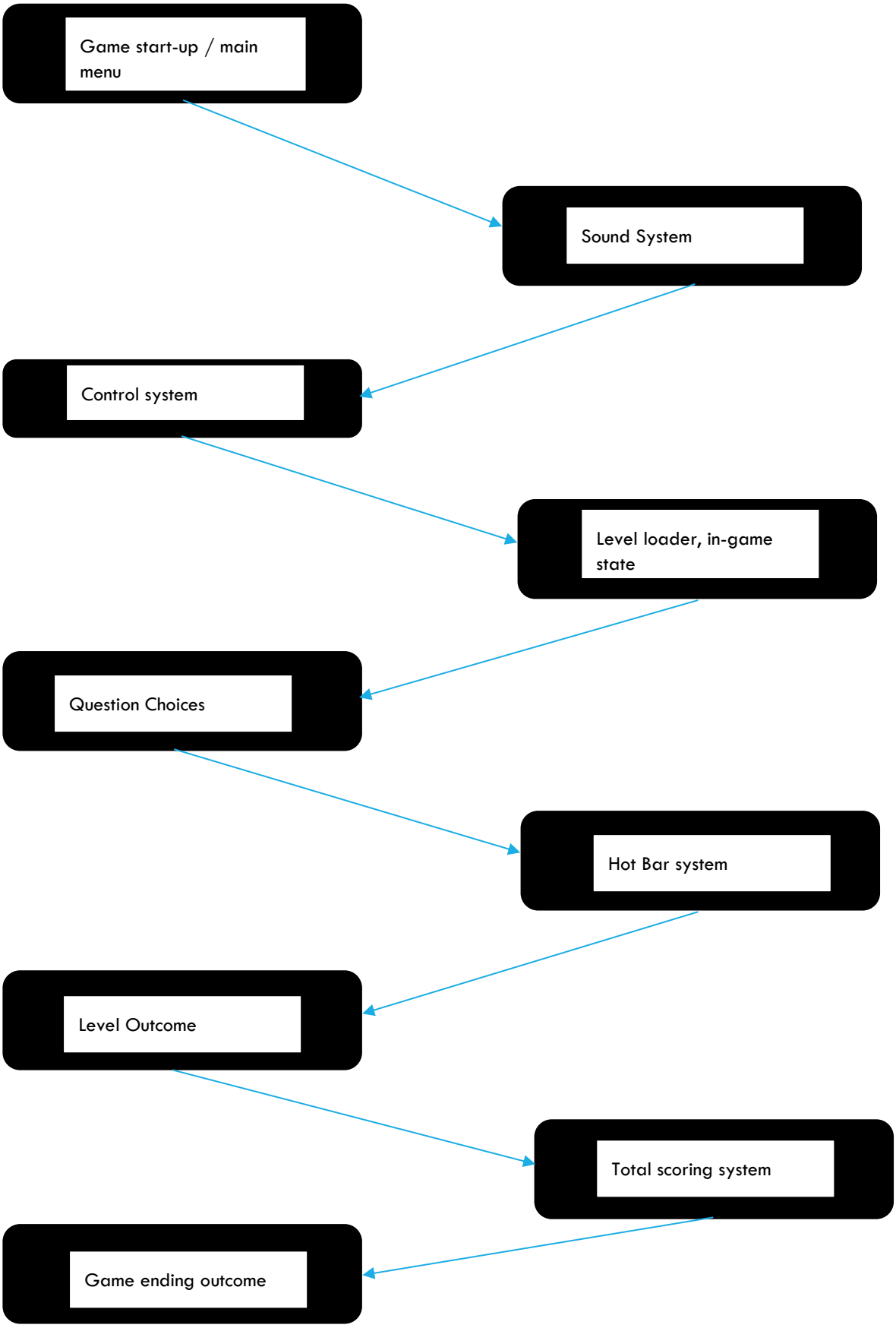


Controls System:

??



Algorithm Link:



TEST DATA TO BE USED DURING DEVELOPMENT

Menu screen

TEST DATA	TYPE
Left mouse click on a button: [play, settings, exit]	Valid
Left mouse click on region in the screen not occupied by a button	Invalid
Right mouse click anywhere on the screen	Invalid

Case detail screen

TEST DATA	TYPE
Space button	Valid
Clicking any other key other than the space button	Invalid
LMC on game exit	Valid

Pause Screen

TEST DATA	TYPE
Clicking 'm' key whilst game_state = running	Valid
Clicking any other key (other than m) whilst game_state = running e.g 'k'	Invalid
'esc' button clicked whilst game_state = paused	valid
Clicking any other button (other than esc) whilst game_state = paused e.g 'o'	Invalid

Playing

TEST DATA	TYPE
LMC on question boxes	Valid

Any other input, other than 'm' or 'LMC' on any other region on the screen, including the question boxes.	Invalid
'm' button click to pause	valid

Ending screen

TEST DATA	TYPE
All keyboard + mouse input e.g (RMC, LMC, 'd',)	Invalid
No user input (due to automatic ending timeout)	Valid

TEST DATA FOR BETA TESTING

Game states testing and what's included in them

Testing for?	Explanation	Hence expected outcome = ? [justification for my test]
Loading up the game	The game is executed and loads up when the user opens the application.	Game loads up without error when triggered to be executed, landing user on the menu screen.
Transitioning to a playing state	The game transitions without error from the menu state to the playing state when triggered to do so.	Game enters a playing state which is dependent on user input. They must select the 'play' button displayed at the main menu. All other input should be invalidated.
Initial popup at start of playing state	As the game enters a playing state and everything is initialised, the case details for the suspect should form a popup on the screen of the user.	Case details with text form a popup on the screen. Background is temporarily blurred for the time duration the user decides to keep the case details up. When user clicks the 'x' button displayed on the popup with LMC, the game continues as normal. All other input is invalidated.
Game termination and closes smoothly	The game closes at certain points in the game without error.	The game will shut down either at the end of the game when the win/lose screen is displayed for a pre-set period without user input, or at the menu screen if the user clicks the 'exit' button, and if the user decides to force close the game whilst playing it.

Transitioning to a paused state and out of it back to playing	The game enters a paused state when triggered to without error and gameplay resumes as normal when transitioning out of it	Game enters a paused state as long as the game state is playing and when the user clicks the 'm' button. Transitions back to a playing state when the user clicks the 'esc' button.
Transitioning to an end state when coming to the end of gameplay	A successful transition to one of two end screens dependent on player's prior performance.	If the player's performance is sufficient for a 'win,' then the win screen is displayed after the last 'interrogation' level. Vice versa, the 'lose' end screen is displayed if otherwise.
Game states only change at the right time	From the main menu, to the playing, to the paused, to the end screens, each game state is only put into action at the right time and the transition between each is successful.	Game states transitioning seamlessly as need be.
End Screen State	At the very end of the game, one of two end screens are displayed to the user. 'win' or 'lose'.	I expect the playing state to transition to the end screen at the right time, and for the end screen to be displayed to the user for a fixed period of time before an automatic termination of the game.

Sound Effects Testing

Testing for?	Explanation	Hence expected outcome = ?
Main menu correct soundtrack playing	The correct soundtrack associated with the main menu should play and for the right duration.	When the game is executed and fully loaded up, the main menu soundtrack plays for the duration the user is on the main menu.
Playing state correct soundtrack playing	As the user is playing the game, a different soundtrack plays in the background	When the user selects play and enters gameplay, the associated soundtrack plays for this state.

Sound effects	The sound effects in the playing state must be triggered at the correct times and play for the set duration it is meant to.	Events in the playing state may trigger particular sound effects. For example, I expect that when the interrogator or suspect is talking, two unique conversational sounds will play. Other sound effects may play at fixed time cycles for the entire gameplay such as the dripping sound effect.
Game end screens soundtrack	One of two endings – lose/win. Two distinct soundtracks for each outcome reflecting the ending.	At the end of the game, the correct soundtrack linked with either the win or lose end screen should play depending on the ending obtained by the player.
Interaction sound	LMC's by the user trigger a clicking sound effect.	Whenever the user clicks on anywhere on the screen, including buttons, a brief clicking sound should play.

Game Mechanics / Controls testing

Testing for?	Explanation	Hence expected outcome = ?
Hot Bar functionality	A hot bar is displayed on a side of the screen only when the level is undergoing. When I select one of three option choices, the hot bar is impacted in a particular way as a result. The hot bar dictates if the level is won or lost.	The hot bar must increment by a fixed amount when I select an excellent choice of question, must remain level if I choose a neutral question, and must decrease by a fixed amount when I select a poor choice of question. The hot bar threshold when reached should count the level as a win, and if not reached by the time all questions are posed by the player, then the level is counted as a loss.
Point and click system	The game heavily focuses on mouse usage, more so over keyboard usage to push the game forward.	The game should respond correctly when the user clicks on buttons with the LMC on the mouse.

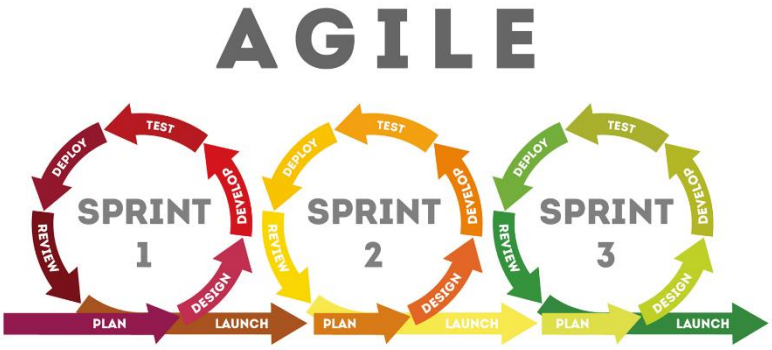
Extras

Testing for?	Explanation	Hence expected outcome = ?
Level Loading	At the commencement of each level, everything should load up correctly.	Case details should initially popup on the screen in the form of paper design along with associated sound effects. Upon closing, the scene should be established along with the sprites in their respective positions. Question choices / hotbar should load in.
Correct respective responses from NPC (suspect)	As the player selects a question choice to pose, the suspect responds with a unique response.	When the user selects a question to pose, the correct response should be delivered back by the suspect.

SOFTWARE DEVELOPMENT PROCESS

For the development of my game, I will be implementing a popular framework from the **Agile methodology**, which will be **KANBAN**. With the agile methodology, the program is developed through a series of short iterations known as sprints, which are short time-boxed periods where I will have focussed goals, outputting new features which would be developed. This cycle continuously repeats until I have built towards a complete solution by integrating all the features I have developed.

I have chosen the Kanban framework alongside the methodology due to its highly efficient nature. Kanban aims to maximise work output through progression in program development by completing tasks. These tasks are defined and the state of them (in terms of progress to their completion) is visually represented on a Kanban board, which displays to me the state of every piece of work at any time A Kanban project has a continuous flow of work as a result, as there is always a task to extract from the backlog and begin to work on.



A representation of the agile methodology. Sprints will take place upon a features/s chosen from the Kanban board, working to complete these features by the end of the sprint.

A representation of a Kanban board. Here I will add a list of tasks to the backlog, all of which when finally done will produce a whole game ready to play.

KEY VARIABLES AND DATA STRUCTURES

Procedure (section of code) , variable (a value stored in memory which can change), Game state (playing, paused, ending) , Sound file (link to sound effects or backtracks)

Names given below are just for reference, they will most likely not be the same in source code.

Method	Name	Data Type	Explanation	Justification

Procedure	Question_Choice	NA	This section of code will be used to store the different question choices the user can select from, and the associated outcome for each of these on the hot bar	The need for this is to allow the question choices to appear on the screen to the player, hence allowing them to make their selection.
Procedure	Hotbar_System	NA	This section of code will directly have an influence on the increments, decreases etc. visually seen in the hot bar displayed on the screen. This has a direct link to the 'Question_Choice' procedure as they are mutually dependent.	The need for this is to allow the hotbar system to determine a win or lose for the level the user is playing.
Variable	Question	String	This variable prompts the player to select one of three different question choices and also displays a mixture of three possible questions which will continue to change throughout the level.	The need for this is to allow the game to progress in the levels by acting as a means to allow the player to select a question choice.
Sound file	Button_click		Each time the player clicks with the left mouse button in an interactive manner, a click sound will play.	This is aimed to immerse the player more into the game. A simple feature.
Game sprite	Question button		A rectangular box which will hold and display each respective question.	Represents the actual questions for the player to see in a presentable manner.
Procedure	Win_ending		The game "win" ending is achieved by the player.	The need for this is to visually display the win end screen.

Procedure	Sound_system		This section of code will store all the backing tracks and sound effects to be used in the game and determine which particular one/s to play dependent on pre-defined conditions.	The need for this is as such. There will be several game states, and within each, a sound effect or backing track is associated to each.
Sound file	Win		Win ending backtrack.	A suitable backtrack of linked with the success of the player is to be played on the win end screen.
Sound file	Lose		Lose ending backtrack.	A suitable backtrack linked with the failure of the player is to be played on the lose end screen.
Sprite	Interrogator		The player's character.	The need for this is to have a fixed design character of the player.
Variable	Level_wins	Integer	This variable will keep track of how many levels the player has won.	The need for this is linked with the ending that will be achieved by the player, determined by how many wins they obtained prior from the total levels.
Sprite	Suspect		The suspect the player is interrogating, which will obviously differ for each level as different 'suspects' are interacted with.	The need for this is to display the suspect the player is aiming to overcome in the level. An important detail that immerses the player and would be nonsensical to not include.

Procedure	Game state	Boolean	This procedure will be utilised to determine what the current game state is, determined by the player's actions as a majority, excluding the ending state which will follow automatically after all levels are completed without requirement for user input.	The need for this is to track the current game state, which allows the appropriate screens, sound effects, options etc. to appear to the user.
Variable	Turns		Used to track when the player is posing a question or receiving a response from the suspect. Asks if the player has asked a question to determine if to proceed.	The need for this is as such. When the player is receiving a response, the question boxes would disappear and text pertaining to the suspect's response would show instead.
Sound_file	Playing_state_sound		A mixture of sounds that will be play at fixed time intervals when the game is in the playing state in the levels.	The need for this is mainly down to immersing the user and making up for some realism from the lack of focus on it in general.
Variable	Total_questions	Integer	how many questions have currently been asked by the player.	The need for this is to track how many questions out of the total possible questions have been asked by the user.
Variable	Case_details	String	At the beginning of each level, a brief case detail of the suspect is displayed on screen.	The need for this is to display the respective case detail for each suspect in each varying level.
Procedure	End_process		Responsible for quitting/terminating the application when prompted to by the user at expected points in the start screen via	The need for this is to close the application when required to do so through certain ways. The application may

			the quit button, or after a fixed time interval at the end screen.	be force closed by the user randomly at any point, however I believe the OS would be more responsible for this.

DEVELOPING THE CODED SOLUTION

Share Copy link Download ... 5 Mark Grid - Matti K.pdf Info 5 / 11

AO 3.2 Developing the coded solution (maximum 25 marks)			
Iterative development of a coded solution (maximum 15 marks)			
1–4 marks	**5–8 marks**	**9–12 marks**	**13–15 marks**
The candidate will have:			
• Provided evidence of some iterative development for a coded solution. • Solution may be linear. • Code may be inefficient. • Code may not be annotated appropriately. • Variable names may be inappropriate. • There will be little or no evidence of validation. • There will be little evidence of review during the development.	• Provided evidence for most stages of the iterative development process for a coded solution describing what they did at each stage. • Solution will have some structure. • Code will be briefly annotated to explain key components. • Some variable and/or structure names will be largely appropriate. • There will be evidence of some basic validation. • There will be evidence that the development was reviewed at some stage during the process.	• Provided evidence of each stage of the iterative development process for a coded solution relating this to the break down of the problem from the analysis stage and explaining what they did at each stage. • Provided evidence of some prototype versions of their solution. • The solution will be modular in nature. • Code will be annotated to explain all key components. • Most variables and structures will be appropriately named. • There will be evidence of validation for most key elements of the solution. • The development will show review at most key stages in the process.	• Provided evidence of each stage of the iterative development process for a coded solution relating this to the break down of the problem from the analysis stage and explaining what they did and justifying why. • Provided evidence of prototype versions of their solution for each stage of the process. • The solution will be well structured and modular in nature. • Code will be annotated to aid future maintenance of the system. • All variables and structures will be appropriately named. • There will be evidence of validation for all key elements of the solution. • The development will show review at all key stages in the process.
Testing to inform development (maximum 10 marks)			
1–2 marks	**3–5 marks**	**6–8 marks**	**9–10 marks**
The candidate will have:			
• Provided some evidence of testing during the iterative development process.	• Provided some evidence of testing during the iterative development process. • Provided evidence of some failed tests and the remedial actions taken.	• Provided evidence of testing at most stages of the iterative development process. • Provided evidence of some failed tests and the remedial actions taken with some explanation of the actions taken.	• Provided evidence of testing at each stage of the iterative development process. • Provided evidence of any failed tests and the remedial actions taken with full justification for any actions taken.

0 marks = no response or no response worthy of credit.

Iterative development: 12

The work is well presented and covers all key parts identified in the analysis phase; this accompanied by clear explanations and justifications of choices made. There is evidence of creating several prototypes at each stage of the development process. However not all codes are commented throughout the process and validation is not present in all parts of the solution.

Testing to inform development:8

Good evidence of testing is shown while failed tests ore only partially evidenced (only 2 failed tests) for this big solution

---

Basic Plan:

AO 3.2 Developing Coded Solution (Max 25 marks)

Iterative development of a coded solution: 15 marks

Provided evidence of each stage of the iterative development process for a coded solution relating this to the breakdown of the problem from the analysis stage and explaining what they did and justifying why.

Provided evidence of prototype versions of their solution for each stage of the process.

The solution will be well structured and modular in nature.

Code will be annotated to aid future maintenance of the system. All variables and structures will be appropriately named.

There will be evidence of validation for all key elements of the solution. The development will show review at all key stages in the process.

Testing to inform development 10 marks

Provided evidence of testing at each stage of the iterative development process. Provided evidence of any failed tests and the remedial actions taken with full justification for any actions taken

! – Please Note that source code image captures were from three different devices throughout my documentation of the development story: my iPad, my personal laptop and the computers provided by my school, hence the appearance of the IDE's may appear different.

23/02/2024: SAVING GAME TRANSCRIPTS TO .TXT FILES

Before I began my source, I decided to write out all the transcripts that will be used across the levels featured in my game, which briefly explain the fictional cases presented to the player and details of the suspects. In addition, I also had to write out the multiple-choice questions the player can pose and the unique responses that would be delivered by the suspect in return.

This was saved to my game file in the end.

Transcript Written Below:

! – APRIL EDIT: I have gone over this part of the documentation and have been advised to highlight the stream of questions which must be picked to obtain a perfect win by my teacher. Hence, below, I will highlight all the [excellent] question choices in green

Level One, case one:

Suspect Name: Liam Cox

Suspect Age: 43

Suspect Height: 6 foot 4

Crime that they may be convicted of if guilty: Aiding a crime and/or Grand Theft Auto (stealing a motorcycle)

Case details: [popup on screen]

On the night of February 7th, 2003, Liam and his friends were captured on CCTV leaving a local bar at around 00:00 and later returning with balaclavas and tools to steal a motorcycle valued at £435,000. A man resembling Liam's height and build was seen riding away on the bike. Without a doubt, it is under suspect that Liam was present at the scene initially but may have not return later to steal the bike, or he did return and was involved in the crime, or he did the latter and even drove away with the bike storing it somewhere.

Interrogation begins…

Question 1:

Excellent: "We have footage of you and your friends casing the motorcycle before the theft. What were you discussing at that time?"

Suspect Response: "I... uh, we were just talking about bikes in general. I swear I didn't know they were planning to steal it!"

Neutral: "Can you tell us more about your interest in motorcycles? Were you looking to buy one?"

Suspect Response: "Motorcycles? Oh, they're cool, but I wasn't planning on buying one. Just admiring them, that's all!."

Poor: "Do you often spend time in the parking lot where the motorcycle was stolen?"

Suspect Response: "Yeah, it's a common hangout spot after the bar. Nothing unusual about that."

Question 2:

Excellent: "Your friends have mentioned your expertise with motorcycles. Did you assist in hot-wiring the bike?"

Suspect Response: "What? No, I mean, I know about bikes, but I didn't help them with anything like that!"

Neutral: "How well do you know the friends you were with that night?"

Suspect Response: "We're just drinking buddies, really. I don't get involved in their personal business."

Poor: "Did you notice anything unusual about your friends' behaviour that night?"

Suspect Response: "Unusual? Nah, it was a typical night out. We were all just having fun."

Question 3:

Excellent: "The tools used in the theft match those found in your garage. How do you explain this?"

Suspect Response: "My tools? But I... I lend them out sometimes. They must've taken them without telling me."

Neutral: "Do you own any tools or equipment that could be used to steal a motorcycle?"

Suspect Response: "I have some tools, yeah, but they're for home repairs. I don't use them for anything else."

Poor: "Are you handy with repairs or mechanical work?"

Suspect Response: "Sure, I fix things around the house occasionally. It's just a hobby, though."

Question 4:

Excellent: "CCTV shows someone with your build wearing a balaclava. How do you explain that?"

Suspect Response: "That wasn't me! I mean, lots of people are my size, right?"

Neutral: "Were you wearing anything that could be mistaken for a balaclava that night?"

Suspect Response: "A balaclava? No, I don't even own one. It must've been someone else."

Poor: "What were you wearing on the night of the theft?"

Suspect Response: "Just my usual clothes. Jeans and a jacket, nothing special."

Question 5:

Excellent: "A witness identified you as the person speeding away on the motorcycle. How do you respond?"

Suspect Response: "They're wrong! I was nowhere near that bike when it happened."

Neutral: "Have you ever ridden a motorcycle similar to the one that was stolen?"

Suspect Response: "I've ridden bikes before, but not one like that. It's way out of my league."

Poor: "Do you enjoy riding motorcycles?"

Suspect Response: "Yeah, I ride occasionally. It's a great feeling, but I didn't steal that bike."

Level 2, Case two

Suspect Name: Elijah Kamara

Suspect Age: 18

Suspect Height: 5 foot 11

Crime that they may be convicted of if guilty: Selling counterfeit goods (replicas)

Case details: [popup on screen]

On the afternoon of July 19th, 2006, Max was spotted at a local market selling what appeared to be high-end designer bags. However, upon closer inspection by a sharp-eyed customer, the goods were identified as replicas and reported to the police. Max has been selling them at extremely high prices and claims to have not been aware they were fake.

Interrogation begins…

Question 1:

Excellent: "We have a statement from a customer saying you assured them these were genuine designer bags whilst showing fake receipts. How do you justify that?"

Suspect Response: "I… I thought they were real! My supplier told me they were, I swear!"

Neutral: "How did you acquire the products you were selling in such large numbers quickly?"

Suspect Response: "Got them from a guy online who said he had a surplus. Good deal, you know?"

Poor: "Do you often sell designer items at the market?"

Suspect Response: "Yeah, I find good deals and pass them on. People love a bargain."

Question 2:

Excellent: "Brand experts confirmed these are fakes. How do you explain the discrepancies?"

Suspect Response: "Discrepancies? I'm not an expert…I didn't notice anything off. And what's their proof anyways, they could just be saying that to stop my hustle."

Neutral: "Are you aware of the differences between genuine and counterfeit products?"

Suspect Response: "Well, I'm no expert, but I try to check the quality best I can."

Poor: "What do you know about the brands you're selling?"

Suspect Response: "Just what I see in ads. They're popular and people love them, so I stock them."

Question 3:

Excellent: "The supplier you mentioned online is known for distributing replicas. Were you aware of their reputation?"

Suspect Response: "No way… I must've not checked properly."

Neutral: "How do you choose your suppliers?"

Suspect Response: "I look for the lowest prices, but I guess I need to be more careful."

Poor: "If people were not going to find out, would you choose a replica supplier to lower costs?"

Suspect Response: "No, never. I try to be as honest as possible."

Question 4:

Excellent: "Do you store this merchandise anywhere else?"

Suspect Response: "My storage? But I… I keep lots of things there, not just bags."

Neutral: "How many products do you order per month from this supplier?"

Suspect Response: "Hmm, I would say about 20 units."

Poor: "What's the money like for your products you are selling?"

Suspect Response: "Pretty good, things move fast. That's why I need the storage."

Question 5:

Excellent: "There's evidence in your bank transactions of you ordering large quantities of replica receipts. Why do you need them?"

Suspect Response: "Labels? I… I use them to just make the product seem more real- I mean, I'm not sure. It's not to fake anything!"

Neutral: "We realised from informants over the past weeks who visited your store that you are very protective over taking pictures of the products. Why is that?

Suspect Response: "Oh that? Yeah, it's just my own marketing tactic to encourage them to buy the product on the spot. Nothing harmful sir."

Poor: "Are you involved in customising the products you sell?"

Suspect Response: "A bit of personal touch here and there, nothing major."

Level 3, Case 3

Suspect Name: Tyler Bennett

Suspect Age: 19

Suspect Height: 5 foot 10

Crime that they may be convicted of if guilty: Vandalism (Graffiti)

Case details: [popup on screen]

In the late hours of September 5th, 2006, a series of graffiti tags were sprayed across the walls of the newly renovated downtown shopping district. The tags match the unique style of 'T-Bone,' a nickname known to be used by Tyler on social media in the past.

Paint cans matching the colours used in the graffiti were found in Tyler's backpack. Tyler insists he was out tagging but claims the shopping district wasn't his canvas for the night.

Interrogation begins…

Question 1:

Excellent: "The tags found downtown match your signature style seen in your past social media accounts. How do you explain that?"

Suspect Response: "Look, I won't lie, 'T-Bone' is me. But I didn't hit the shopping district. That's not my work."

Neutral: "What's the significance of the 'T-Bone' tag? We've seen it around a lot lately."

Suspect Response: "It's just a tag, you know? It's how I express myself. But many others may use it too."

Poor: "Do you know anyone else who uses the 'T-Bone' tag?"

Suspect Response: "Yeah! Plenty of people! I invented that junk but others have been copying it for time. So, I'm telling you, I didn't tag the district."

Question 2:

Excellent: "We found spray paint in your backpack. Were you out tagging last night?"

Suspect Response: "Yeah… I was out, but not there. I stick to the abandoned old factory district, away from the crowds."

Neutral: "Why do you carry spray paint around? Is it always for tagging?"

Suspect Response: "Mostly, yeah. Sometimes I do commissioned pieces, legal ones, you know? That's like asking a smoker why he carries cigs, you feel me?"

Poor: "Is spray painting something you do often?"

Suspect Response: "It's my thing, but I'm not dumb enough to hit a spot like that on opening week."

Question 3:

Excellent: "CCTV caught someone with your build in the area and no one was seen near the old factory similar to your build throughout the entire night. Can you account for your whereabouts?"

Suspect Response: "Um, I think I entered and left the factory through alleyways and stuff… not really on the road so maybe that's why."

Neutral: "Have you visited the shopping district recently?"

Suspect Response: "Not recently, no. I avoid places with too much heat."

Poor: “Do you have any issues with the shopping district or its owners?”

Suspect Response: “Issues? Nah, it’s just another part of the city to me.”

Question 4:

Excellent: “A witness in their home near your address described someone matching your description fleeing the scene. What were you running from?”

Suspect Response: “Running? I wasn’t running from anything. Maybe they saw me… jogging. I live nearby.”

Neutral: “Do you usually work alone, or do you have a crew?”

Suspect Response: “I’m a lone wolf. Crews bring drama and heat, not my style.”

Poor: “What time did you get home last night?”

Suspect Response: “Late, like 3 AM. But I was nowhere near downtown, I swear.”

Question 5:

Excellent: “Your fingerprints were found on a can at the scene. Still saying you weren’t there?”

Suspect Response: “That’s… that’s impossible. I mean, I touch a lot of cans, but I didn’t drop any at the district.”

Neutral: “How do you choose the locations for your tags?”

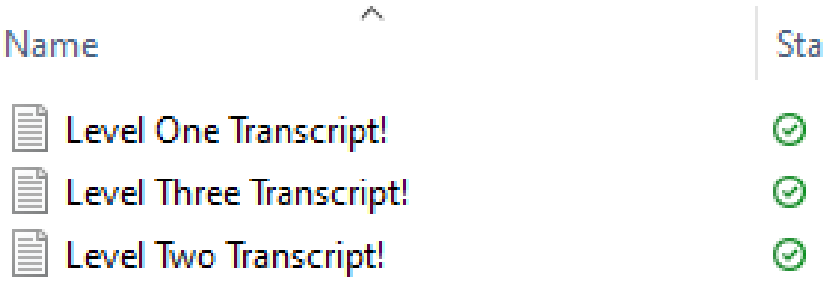
Suspect Response: “I look for spots that speak to me, you know? Places that aren’t owned and usually abandoned that could use some care.”

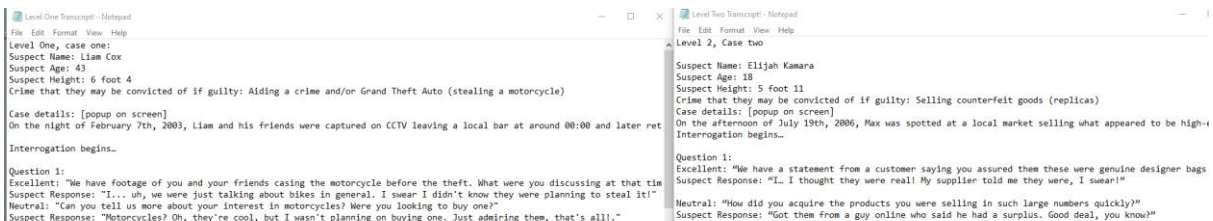
Poor: “What do you aim to communicate with your graffiti?”

Suspect Response: “It’s about making a mark, showing the city there’s art in unexpected places. That’s why I stick strictly to abandoned places you see?”

END TRANSCRIPT

I then copied all the transcripts to a .txt file which was saved in a location within my game’s folder so I could copy the elements of it at a later stage to the source code:





04/03/23: CREATING THE MENU OF MY GAME

The best place to start with the development was creating the menu screen and assets for it.

Inserts the necessary libraries.

Defining Game Class

Sets game display screen size

Initialises lost and won state to false and level =1.

```
import sys
import pygame
import random

class Ga
    def
        (parameter) self: Self@Game
        self.size = (900,650)
        self.screen = pygame.display.set_mode(self.size)
        pygame.display.set_caption('Hot Seat')
        self.clock = pygame.time.Clock()
        self.lost = False
        self.won = False
        self.level = 1
        self.assets = {
            'menu' : pygame.transform.scale(pygame.image.load('images/menu.png'),self.size),
        }
        self.rects = {
            'menu' : [
                pygame.Rect(10,185,250,70),
                pygame.Rect(10,270,250,70),
                pygame.Rect(730,0,150,70),
```

Executing the program gave me my desired output as shown below:

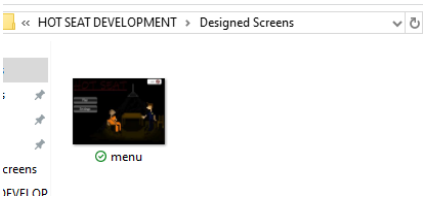
Defines the rectangle interactive elements.

E.g.: 'Play' button

Loading and scaling game assets: menu screen image I created

APRIL UPDATE:

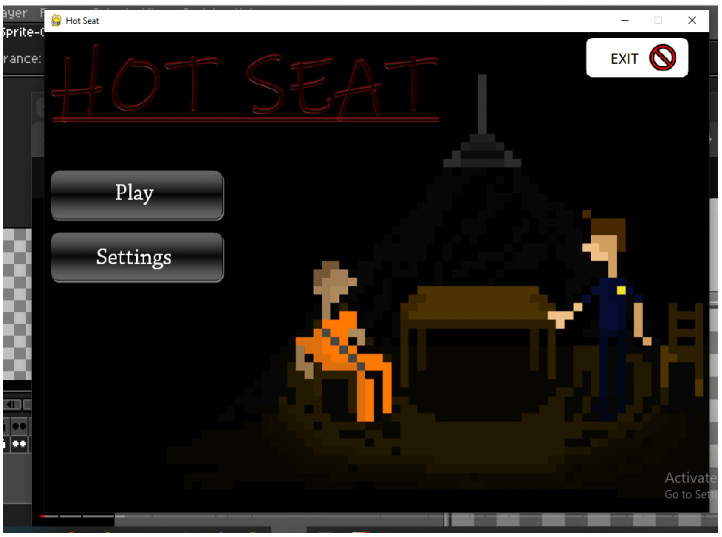
I realised that I have not specified comments in the image above, however I can assure that comments were added. Below are some update screenshots of the updated version of the same sections of source code:




```
class Game():
    def __init__(self) -> None:
        # Initialize the game
        pygame.mixer.pre_init()
        pygame.init()
        self.size = (900,650)
        self.screen = pygame.display.set_mode(self.size, flags=pygame.SRCALPHA)  # Create game window
        pygame.display.set_caption('Hot Seat')  # Set window title
        self.music = True# Music setting
        self.sfx = True  # Sound effects setting
        self.clock = pygame.time.Clock()  # Game clock
```

```
self.lost = False  # Flag for player losing the game
self.won = False  # Flag for player winning the game
self.wins = 0
self.time = 14
self.losses = 0
self.level = 0  # Current level of the game
self.max_level = 2
# Load game assets
self.assets = {

   # Define clickable regions on the menu screen
   self.rects = {
       'menu' : [
           pygame.Rect(10,185,250,70),  # Rectangle for "Play" button
           pygame.Rect(10,270,250,70),  # Rectangle for "Settings" button
           pygame.Rect(730,0,150,70),   # Rectangle for "Quit" button
       ]
   `
```

The menu screen has been successfully setup. All interactive elements are present + the menu screen design is displayed appropriately.



Game loading upon execution:

What is being tested?	Input	Justification of my input	Outcome (as a result of input)	How to solve (if error occurs)
Does the main menu load as the first thing along with all associated assets	Executing the game	To allow the game to start	The menu loads as expected with the correct, and interactive assets	NA

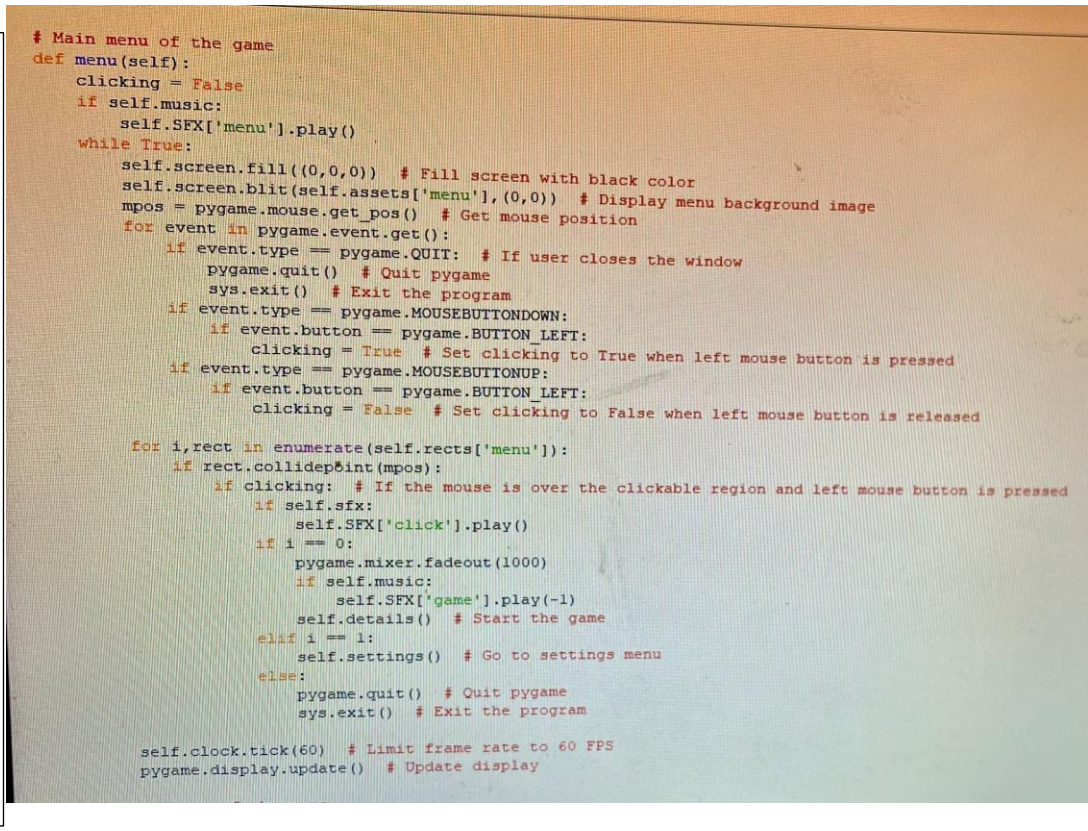
Coding the functionality of the main menu

The menu function to the right has within it several key functionalities and assets regarding the initial landing point of the user when they execute the application – the main menu.

A Boolean variable is used to track if the mouse button is clicked.

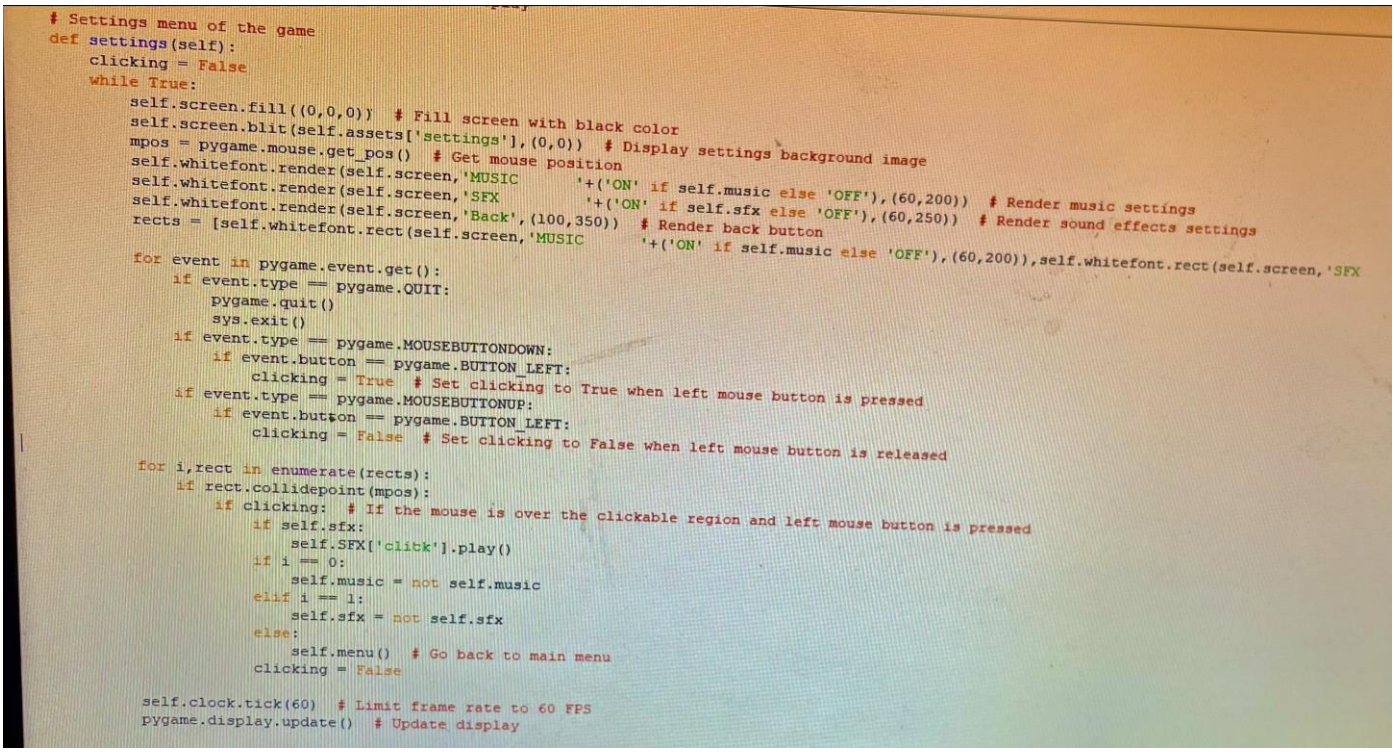
'self.SFX,' allows menu music to play, along with a click sound effect I also decided to integrate when a menu item is pressed. When the user clicks play, the soundtrack for the playing state seen as 'game' begins to play.

Obtaining the mouse position and tracking whether it has been pressed over a significant region - the collision points over the rectangle assets: play, settings, exit.



```
# Main menu of the game
def menu(self):
    clicking = False
    if self.music:
        self.SFX['menu'].play()
    while True:
        self.screen.fill((0,0,0))  # Fill screen with black color
        self.screen.blit(self.assets['menu'],(0,0))  # Display menu background image
        mpos = pygame.mouse.get_pos()  # Get mouse position
        for event in pygame.event.get():
            if event.type == pygame.QUIT:  # If user closes the window
                pygame.quit()  # Quit pygame
                sys.exit()  # Exit the program
            if event.type == pygame.MOUSEBUTTONDOWN:
                if event.button == pygame.BUTTON_LEFT:
                    clicking = True  # Set clicking to True when left mouse button is pressed
            if event.type == pygame.MOUSEBUTTONUP:
                if event.button == pygame.BUTTON_LEFT:
                    clicking = False  # Set clicking to False when left mouse button is released

        for i,rect in enumerate(self.rects['menu']):
            if rect.collidepoint(mpos):
                if clicking:  # If the mouse is over the clickable region and left mouse button is pressed
                    if self.sfx:
                        self.SFX['click'].play()
                    if i == 0:
                        pygame.mixer.fadeout(1000)
                        if self.music:
                            self.SFX['game'].play(-1)
                        self.details()  # Start the game
                    elif i == 1:
                        self.settings()  # Go to settings menu
                    else:
                        pygame.quit()  # Quit pygame
                        sys.exit()  # Exit the program

        self.clock.tick(60)  # Limit frame rate to 60 FPS
        pygame.display.update()  # Update display
```

Settings functionality which can be opened via the main menu rect asset coded:



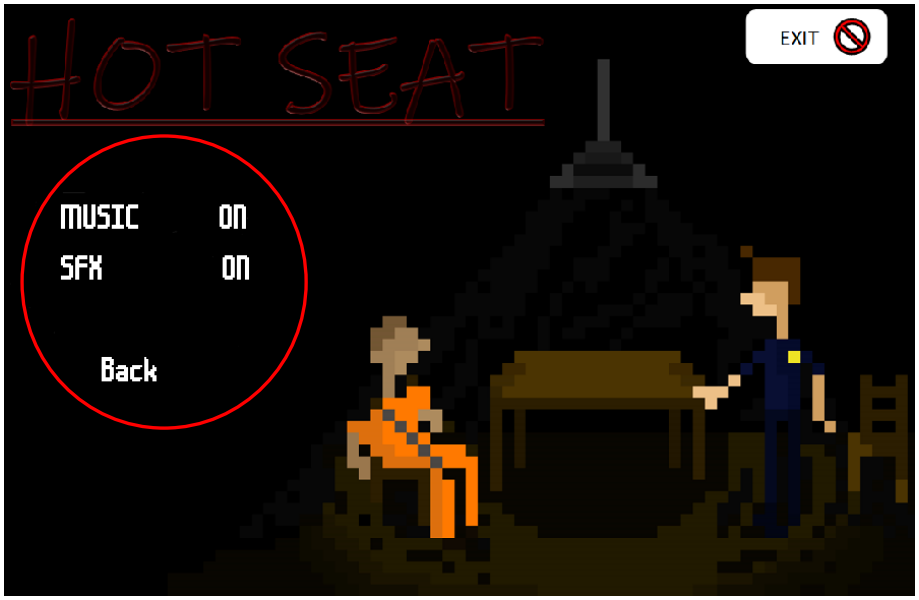
```
# Settings menu of the game
def settings(self):
    clicking = False
    while True:
        self.screen.fill((0,0,0))  # Fill screen with black color
        self.screen.blit(self.assets['settings'],(0,0))  # Display settings background image
        mpos = pygame.mouse.get_pos()  # Get mouse position
        self.whitefont.render(self.screen,'MUSIC       '+('ON' if self.music else 'OFF'),(60,200))  # Render music settings
        self.whitefont.render(self.screen,'SFX         '+('ON' if self.sfx else 'OFF'),(60,250))  # Render sound effects settings
        self.whitefont.render(self.screen,'Back',(100,350))  # Render back button
        rects = [self.whitefont.rect(self.screen,'MUSIC       '+('ON' if self.music else 'OFF'),(60,200)),self.whitefont.rect(self.screen,'SFX

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.MOUSEBUTTONDOWN:
                if event.button == pygame.BUTTON_LEFT:
                    clicking = True  # Set clicking to True when left mouse button is pressed
            if event.type == pygame.MOUSEBUTTONUP:
                if event.button == pygame.BUTTON_LEFT:
                    clicking = False  # Set clicking to False when left mouse button is released

        for i,rect in enumerate(rects):
            if rect.collidepoint(mpos):
                if clicking:  # If the mouse is over the clickable region and left mouse button is pressed
                    if self.sfx:
                        self.SFX['click'].play()
                    if i == 0:
                        self.music = not self.music
                    elif i == 1:
                        self.sfx = not self.sfx
                    else:
                        self.menu()  # Go back to main menu
                    clicking = False

        self.clock.tick(60)  # Limit frame rate to 60 FPS
        pygame.display.update()  # Update display
```

As known, within the main menu, there are three interactive assets on the screen. The exit button if interacted with leads to a system exit. The play button triggers a new state of the game to start, the playing state. However, I had to code the setting button, and what the user could access within it.

For simplicity, I allowed the user to tweak two settings within the settings aspect of the game. This being to toggle between ON and OFF for music and sfx (sound effects) in the game.

'enumerate(rects)' as seen in the source code is essential to iterate over the rectangle regions over which the user may interact with the mouse pointer.

To ensure a smooth exit back to the main menu, 'self.menu' calls the main menu function when the back button is clicked.



Testing the main menu:

What is being tested?	Input	Justification of my input	Outcome (as a result of input)	How to solve (if error occurs)
Is there a correct game state change	Clicking the exit button	By clicking the exit button, this will cause the game to exit	The game exits	NA
Is there a correct game state change	Clicking the settings button	By clicking the settings button, the settings menu should load	The setting menu loads up	NA
Is there a correct game state change	Click the play button	By clicking the play button, the game should start and display the case details	Case details displayed for first level	NA

Menu screen

TEST DATA	TYPE	
Left mouse click on a button: [play, settings, exit]	Valid	✓
Left mouse click on region in the screen not occupied by a button	Invalid	✓
Right mouse click anywhere on the screen	Invalid	✓

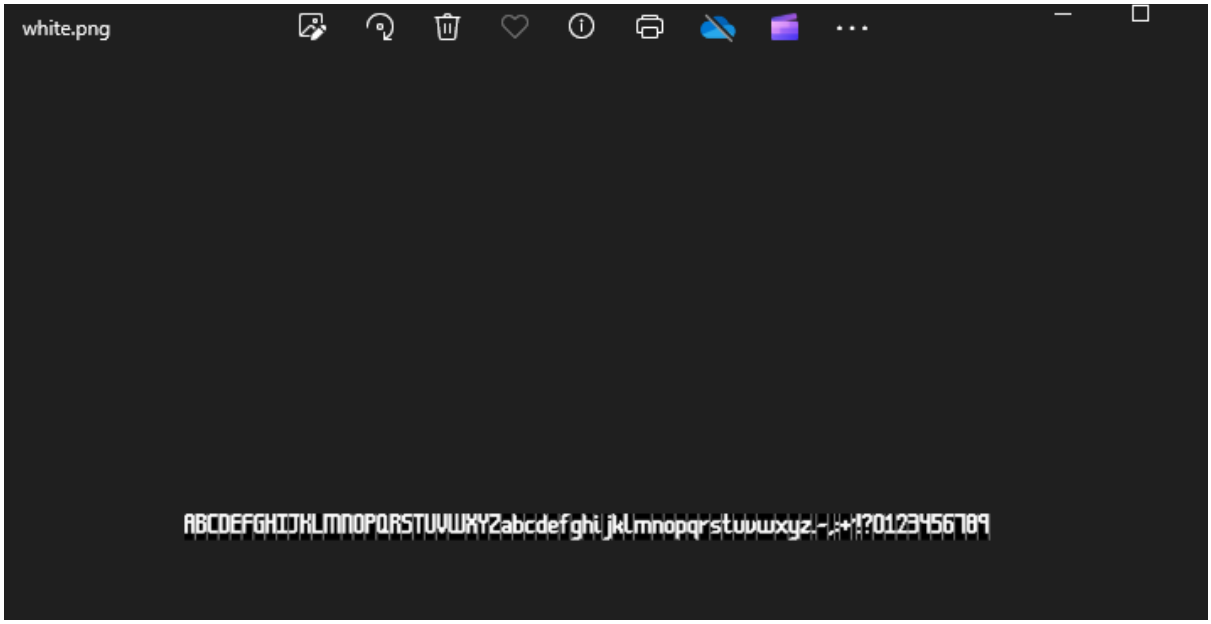
05/03/24: CREATING CUSTOM FONTS FOR THE GAME

The game is going to have a lot of dialogue and the system fonts presented are not very appealing to me, and I believe may not be of large appeal therefore to the end-user. Hence, here I create custom fonts to enhance text quality.

```python
import pygame

# Function to clip a portion of an image
def clip(img, x, y, x_size, y_size):
    # Copy the image
    handle_img = img.copy()
    # Create a rectangular clip region
    clipR = pygame.Rect(x, y, x_size, y_size)
    # Set the clip region
    handle_img.set_clip(clipR)
    # Extract the clipped portion of the image
    image = img.subsurface(handle_img.get_clip())
    # Return the clipped portion as a separate image
    return image.copy()

# Class for handling fonts
class Font():
    def __init__(self, path, size):
        # Space between letters
        self.spacing = 1
        # List of characters in the image ordered in the order they appear in the image
        self.character_order = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W
        # Load font image
        font_img = pygame.image.load(path).convert()
        current_char_width = 0
        self.characters = {}
        character_count = 0
```

The overall purpose of the code can be broken down into two parts. The function ‘clip’ clips a portion of an image, which can be used to extract the images of each character from the font image file:

Example of one of the font image files used for the game:



The font class allows me to create different fonts/styles/effects which are not available in the default pygame fonts. The _init_ method intialises the font object with the path and the size of the image file I am using. The spacing attribute is used to set the space between the letters; the character_order attribute stores the order of the characters in the font image file.

In addition, there are also methods to display the text on the screen to the end-user.

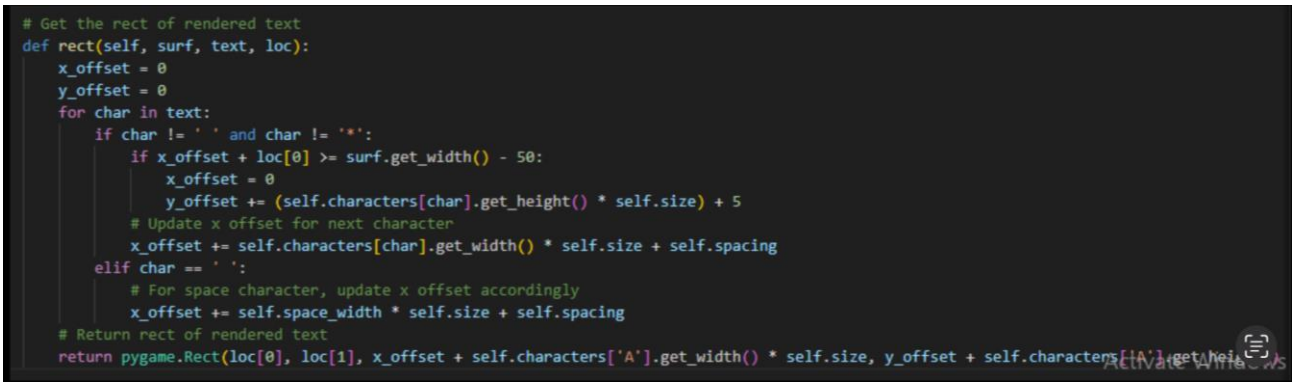
```
# Check for letter borders which are grey and have a red value of 127
for x in range(font_img.get_width()):
    c = font_img.get_at((x, 0))
    if c[0] == 127:
        # Clip the image and add it to the dictionary
        char_img = clip(font_img, x - current_char_width, 0, current_char_width, font_img.get_height())
        self.characters[self.character_order[character_count]] = char_img.copy()
        character_count += 1
        current_char_width = 0
    else:
        current_char_width += 1
# Set the space character's width to the width of the letter A
self.space_width = self.characters['A'].get_width()
self.size = size
```

The code above allows me to utilise a dictionary in correspondence to the font characters. This allows me to efficiently store the letter images extracted from the font image so they can be used along the game. Dictionaries utilise a key:value pair, and in my case, my key for each letter image would be the letter itself such as: ‘A’, ‘B’, ‘C’ etc.

The code loops through each pixel column of the font image, starting from the leftmost one, checking if its red component is 127. This is a way of detecting the grey border that separates each letter in the image. If the pixel is grey, it means a letter has ended, so an image is clipped from the previous border to the current one and adds it to the dictionary as a value with a corresponding character as the key as mentioned above.

If the pixel is not grey, it means the letter is continuing, hence increments the width of the current letter by one.

The space character’s width (separation between characters) is set to the same of that as the letter A.



```
# Get the rect of rendered text
def rect(self, surf, text, loc):
    x_offset = 0
    y_offset = 0
    for char in text:
        if char != ' ' and char != '*':
            if x_offset + loc[0] >= surf.get_width() - 50:
                x_offset = 0
                y_offset += (self.characters[char].get_height() * self.size) + 5
            # Update x offset for next character
            x_offset += self.characters[char].get_width() * self.size + self.spacing
        elif char == ' ':
            # For space character, update x offset accordingly
            x_offset += self.space_width * self.size + self.spacing
    # Return rect of rendered text
    return pygame.Rect(loc[0], loc[1], x_offset + self.characters['A'].get_width() * self.size, y_offset + self.characters['A'].get_Hei
```

Finally, in order to get the dimensions and position of text on the surface that will be shown, I decided to use a rectangle for rendered text. By using a 'rect' object that python provides, I am able to access the x, y, width and height of the rendered text.

The surface object being the text is represented with the 'surf' parameter.

The x_offset and y_offest variables keep track of the horizontal and vertical offsets of each character from the location.

If the character is not a dot or asterisk, then we have a normal letter. The method inside the code checks if the x-co-ordinate of the location of the character exceeds the width of the surface of the screen, and hence if it needs to wrap to the next line. The y_offset is used to move to the text to the next line with a 5-pixel gap between the lines.

If the text does not need to wrap, then the method updates the x_offset by the width of the character + spacing between characters which moves the text to the right as normal.

If the character is a space, then the x_offset is updated such that the text is moved to the right by the size of the space.

The letter A is used as a reference as we assume it to have the same size as other characters.

08/03/24: MAKING A UTILITY SCRIPT

Creating this script was beneficial to store two functions which both had two slightly different purposes linked to loading images in the game.

```
import os
import pygame

BASE_IMG_PATH = 'assets/images/'



def load_image(path,size=None):
    if size:
        img = pygame.transform.scale(pygame.image.load(BASE_IMG_PATH + path).convert_alpha(),size)
    else:
        img = pygame.image.load(BASE_IMG_PATH + path).convert_alpha()
    return img

def load_images(path,size=None):
    images = []
    for img_name in sorted(os.listdir(BASE_IMG_PATH + path)):
        images.append(load_image(path + '/' + img_name,size=size))
    return images
```

The function load_image is used to load an *individual* image.

1. A full path is set to the image file

2. The image is loaded and converted to a suitable format for pygame (transparent backgrounds).

The function load_images is used to load several images form a dictionary.

1. Construct a full path to the dictionary

2. Iterate over the sorted image files in the dictionary

3. Each image is then loaded

The reason for these two seperate functions is as follows.

There will be several points in the game where several images will need to be loaded and implemented on the screen at the same time, such as in an interrogation level which includes several sprites, a background, question box assets etc.

There can also be points in the game where a single image needs to be loaded. An example is the win and lose end screen.

My reasoning behind this choice of introducing a utility script:

- modularity: this is something key I am trying to achieve for the entire development of this program. By creating separate scripts for the overall game I can keep my code organised.

- re-usability: the functions defined can be used throughout the game as will be seen in the next stage of my development below, instead of writing the same code again and again multiple times.

09/03/24: UPDATED GAME CLASS

As the game development progresses, I need to add more detail and make some changes to the game class so it in encompass the entire development of the game.

```
class Game():
    def __init__(self) -> None:
        # Initialize the game
        pygame.mixer.pre_init()
        pygame.init()
        self.size = (900,650)
        self.screen = pygame.display.set_mode(self.size, flags=pygame.SRCALPHA)  # Create game window
        pygame.display.set_caption('Hot Seat')  # Set window title
        self.music = True# Music setting
        self.sfx = True  # Sound effects setting
        self.clock = pygame.time.Clock()  # Game clock
        self.whitefont =  Font('assets/images/fonts/white.png',2)  # Initialize font for text rendering
        self.bigwhitefont =  Font('assets/images/fonts/white.png',3)
        self.smallwhitefont =  Font('assets/images/fonts/white.png',1.2)
        self.Greyfont =  Font('assets/images/fonts/grey.png',2)  # Initialize font for text rendering
        self.blackfont =  Font('assets/images/fonts/black.png',1.5)
        self.smallblackfont =  Font('assets/images/fonts/black.png',1.2)
        self.lost = False  # Flag for player losing the game
        self.won = False  # Flag for player winning the game
        self.wins = 0
        self.time = 14
        self.losses = 0
        self.level = 0  # Current level of the game
        self.max_level = 2
        # Load game assets
        self.assets = {
            'menu' : pygame.transform.scale(load_image('menu.png'),self.size),
            'settings' : pygame.transform.scale(load_image('settings.png'),self.size),
            'board' : pygame.transform.scale(load_image('board.png'),self.size),
            'won' : load_image('won.png',size=self.size),
            'lost' : load_image('lost.png',size=self.size),
            'mugshots' : load_images('mugshots',size=(45,45)),#contains a list of images ordered by name
            'level_0' :load_images('levels/0',size=(self.size[0]-150,self.size[1]-150)),
            'level_1' :load_images('levels/1',size=(self.size[0]-150,self.size[1]-150)),
            'level_2' :load_images('levels/2',size=(self.size[0]-150,self.size[1]-150)),
```

As can be seen in the updated version of the game class, it now encompasses the different fonts which can be used for text throughout the game.

Wins and losses for the 3 playable levels are set to zero, and can increment throughout the game, which will be covered later.

```
}
self.SFX = {
    'menu' : pygame.mixer.Sound('assets/music/menu.mp3'),
    'won' : pygame.mixer.Sound('assets/music/win.mp3'),
    'lost' : pygame.mixer.Sound('assets/music/lose.mp3'),
    'game' : pygame.mixer.Sound('assets/music/game.mp3'),
    'pause' : pygame.mixer.Sound('assets/music/Pause.mp3'),
    'click' : pygame.mixer.Sound('assets/music/click.mp3'),
}
self.SFX['game'].set_volume(0.5)#adjust the volume
self.SFX['menu'].set_volume(0.5)#adjust the volume
self.bar = Bar((30,400),(30,300))
# Define clickable regions on the menu screen
self.rects = {
    'menu' : [
        pygame.Rect(10,185,250,70),  # Rectangle for "Play" button
        pygame.Rect(10,270,250,70),  # Rectangle for "Settings" button
        pygame.Rect(730,0,150,70),   # Rectangle for "Quit" button
    ]
}
# Details for each level of the game
self.casedetails = [
    '''Suspect Name: Liam Cox*Suspect Age: 43*Suspect Height: 6 foot 4*Crime that they may be convicted of if guilty: Aiding a crime and Grand Theft Auto *On t
    '''Suspect Name: Elijah Kamara*Suspect Age: 18*Suspect Height: 5 foot 11*Crime that they may be convicted of if guilty: Selling counterfeit goods aka repli
    '''Suspect Name: Tyler Bennett*Suspect Age: 19*Suspect Height: 5 foot 10*Crime that they may be convicted of if guilty: Vandalism aka Graffiti
```

I also introduce here the music tracks/effects which will be used throughout the game, all defined here in the game class. These, along with all the assets and images are stored in a local game file and accessed via that.

The different music tracks/effects were obtained from YouTube copyright free sources and downloaded in the format of mp3 files.

The rectangles representing the main menu buttons are nothing new and are seen as before in the prior early version of the game class.

I also stored the details for each of the 3 levels in the game. This will include details from the ***transcript*** I designed at the beginning of my development.

Includes the questions of each level.
Snippet shown below

```
]
# Questions and answers for each level
self.levelquestions = {
    0 : {
        1 : {
            "We have footage of you and your friends casing the motorcycle before the theft. *What were you discussing at that time?":"I... uh, we were just
            "Can you tell us more about your interest in motorcycles? Were you looking to buy one?":"Motorcycles? Oh, they're cool, but I wasn't planning on
            "Do you often spend time in the parking lot where the motorcycle was stolen?":"Yeah, it's a common hangout spot after the bar. Nothing unusual a
        },
        2 : {
            "Your friends have mentioned your expertise with motorcycles. Did you assist in hot-wiring the*bike?  ":"What? No, I mean, I know about bikes, b
            "How well do you know the friends you were with that night?":"We're just drinking buddies, really. I don't get involved in their personal busine
            "Did you notice anything unusual about your friends' behaviour that night?":"Unusual? Nah, it was a typical night out. We were all just having f
        },
        3 : {
            "The tools used in the theft match those found in your garage. How do you explain this?":  "My tools? But I... I lend them out sometimes. They m
            "Do you own any tools or equipment that could be used to steal a motorcycle?": "I have some tools, yeah, but they're for home repairs. I don't u
            "Are you handy with repairs or mechanical work?": "Sure, I fix things around the house occasionally. It's just a hobby, though."
        },
        4 : {
            "CCTV shows someone with your build wearing a balaclava. How do you explain that?": "That wasn't me! I mean, lots of people are my size, right?"
            "Were you wearing anything that could be mistaken for a balaclava that night?": "A balaclava? No, I don't even own one. It must've been someone
            "What were you wearing on the night of the theft?": "Just my usual clothes. Jeans and a jacket, nothing special."
        },
        5 : {
            "A witness identified you as the person speeding away on the motorcycle. How do you respond?": "They're wrong! I was nowhere near that bike when
            "Have you ever ridden a motorcycle similar to the one that was stolen?": "I've ridden bikes before, but not one like that. It's way out of my le
            "Do you enjoy riding motorcycles?": "Yeah, I ride occasionally. It's a great feeling, but I didn't steal that bike."
        }
    },

    1 :{
        1 :{
            'We have a statement from a customer saying you assured them these were genuine designer bags whilst showing fake receipts. How do you justify t
            'How did you acquire the products you were selling in such large numbers quickly?' : 'Got them from a guy online who said he had a surplus. Good
            "Do you often sell designer items at the market?": "Yeah, I find good deals and pass them on. People love a bargain.",
        },
        2: {
            'Brand experts confirmed these are fakes. How do you explain the discrepancies?': 'Discrepancies? Im not an expert...I didnt notice anything off
```

10/03/24: IMPLEMENTING THE DETAILS METHOD.

The purpose of the "details method" would consequently be responsible for rendering the fictional case details pop-up before the user begins playing the level or chooses to exit the game. Not only would this display the case detail popup, but many other key functionalities as described before in the design documentation come into action here.

```
def details(self):
    if self.music:#placeholder for when I implement music
        pass
        #random.choice(list(self.sfx)).play()
    while True:
        self.screen.fill((0,0,0))  # Fill screen with black color
        self.screen.blit(self.assets['board'],(0,0))  # Display a detective board to blit the details on top of
        self.screen.blit(self.assets['mugshots'][self.level],(200,250))


        self.blackfont.render(self.screen,'case details',(360,220))
        self.blackfont.render(self.screen,self.casedetails[self.level],(280,270),650)  # Render case details settings
        self.whitefont.render(self.screen,'press space to continue',(260,520))

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_SPACE:
                    self.run()
        self.clock.tick(60)  # Limit frame rate to 60 FPS
        pygame.display.update()  # Update display
```

Breaking down key components of the "details" method:

```
def details(self):
    if self.music:#placeholder for when I implement music
        pass
        #random.choice(list(self.sfx)).play()
```

The first component of the method will be responsible for the playing of a music track and other sound effects. implemented a pass statement as a placeholder for future code I will implement here to run the music.

```
while True:
    self.screen.fill((0,0,0))  # Fill screen with black color
    self.screen.blit(self.assets['board'],(0,0))  # Display a detective board to blit the details on top of
    self.screen.blit(self.assets['mugshots'][self.level],(200,250))


    self.blackfont.render(self.screen,'case details',(360,220))
    self.blackfont.render(self.screen,self.casedetails[self.level],(280,270),650)  # Render case details settings
    self.whitefont.render(self.screen,'press space to continue',(260,520))
```

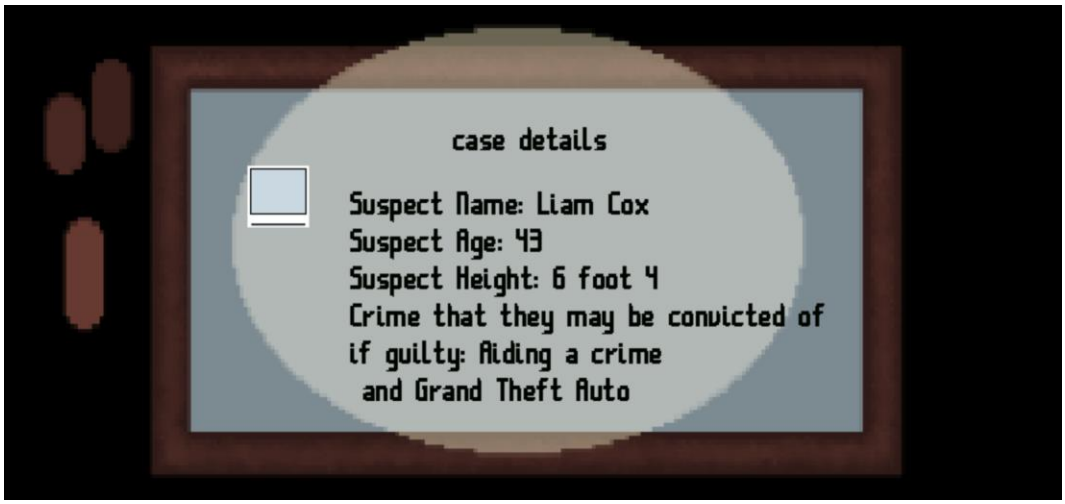
The second component here initially runs a while loop to run an infinite loop to keep displaying the case details until some form of input is detected by the user which is covered later on below.

The following code visually displays assets I created; these being used in the case detail pop-up section before the interrogation level officially starts.

To begin, the screen is filled with a black colour, over which the code will blit (draw) the detective board asset. An interesting feature I added which will have its respective position indicated by the co-ordinate values will be a mugshot of the suspect.

And then, in black font, I render my prior written case details from the transcript written earlier.

This is the result:



And for the third component of this method, a very important part, is responsible for key input functionalities from the player at this current state in the game.

```
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_SPACE:
                    self.run()
        self.clock.tick(60)  # Limit frame rate to 60 FPS
        pygame.display.update()  # Update display
```

Here I utilise an event handling loop.

1. If the window's close button is clicked, the game is terminated and quit.

2. If the user clicks the 'space' button, then the game continues on to the level

3. With some brief research, I found that capping the fps can be particularly benefit for lower performance systems. This can reduce potential heat generation and also free up CPU usage for other tasks to prevent any game issues such as stuttering or lagging.

Here is a snippet of a dictionary I used to store some of the game assets which have been used above in development so far:

```
        self.assets = {
            'menu' : pygame.transform.scale(load_image('menu.png'),self.size),
            'settings' : pygame.transform.scale(load_image('settings.png'),self.size),
            'board' : pygame.transform.scale(load_image('board.png'),self.size),
            'mugshots' : load_images('mugshots'),#contains a list of images ordered by name
        }
```

load_image allows the image stored in the game files to be displayed.

self.size ensures only the desired size of these assets are loaded onto the game screen.

The load_images return a list of images ordered by name.

What is being tested?	Input	Justification of my input	Outcome (as a result of input)	How to solve (if error occurs)
Case detail pop-up	Clicking the play button	The case detail screen should	The case detail screen renders as intended	NA

		popup after the play button is hit		
Game state transition from case detail screen to level	Clicking the space button	The game should proceed onto the level smoothly after the space button is pressed	The level starts as a result	NA

Case detail screen

TEST DATA	TYPE
Space button	Valid ✓
Clicking any other key other than the space button	Invalid ✓
LMC on game exit	Valid ✓

11/03/24: CREATING THE ELEMENTS REQUIRED FOR LEVELS TO RUN

I decided to start by creating a class for the hot bar which will allow me to create a visual representation of it, and the mechanics that will be required behind it, responsible for filling and depleting etc.

```
class Bar:
    def __init__(self, pos, size):
        self.pos = list(pos)  # Convert position to a list
        self.size = list(size)  # Convert size to a list

    def draw(self, surface, ratio):
        # Adjust the size of the bar based on the provided ratio
        size = list(self.size)
        size[1] *= ratio

        # Draw the filled rectangle representing the bar
        pygame.draw.rect(surface, (255, 255, 255), (self.pos[0], self.pos[1] - size[1], *size))

        # Draw the outline of the bar
        pygame.draw.rect(surface, (255, 255, 255), (self.pos[0], self.pos[1] - self.size[1], *self.size), 2)
```

This section of code within the class is responsible for creating the hotbar using pygame's drawing functionality provided, which I found to be much more useful compared to importing a sprite which would make coding much harder. Output displayed on the right

list(pos) and list(size) are vital to store the position of the hotbar on the screen when it is displayed and how big the hotbar should be in terms of the filled rectangle within it.

Randomisation of texts, system to check if answer is correct, system to dynamically adjust spacing between questions to ensure they don't overlap.

```
# Main gameplay loop
while True:
    ratio = questions_answered / total_questions  # Calculate the ratio of correct answers to total questions
    counter += 1  # Increment the counter

    # Fill the screen with black color
    self.screen.fill((0, 0, 0))

    # Display the level background image
    self.screen.blit(self.assets['level_' + str(self.level)][0], (self.screen.get_width() / 2 -
                                                                   self.assets['level_' + str(self.level)][
                                                                       0].get_width() / 2, 0))
```

In this section of code, I incorporate the ratio calculation which is used to update the hot bar display whilst the level is played.

'questions_answered / total_questions' calculates the ratio of questions the player has answered correctly to the total number of questions that can be posed in each level.

I also load in the level background image. In this case, due to ease, I used the same basic image in the main menu, however I will change the suspect sprite in the chair for each of the three levels.

```
# Display instructions if counter is less than 10
if counter < 10:
    self.whitefont.render(self.screen, 'ask the write question to fill the progress bar up', (260, 520))
elif choice == None and not gameover:
    # Display randomized questions and their choices
    for i, question in enumerate(randomlist[question_number]):
        if i == 0:
            self.smallwhitefont.render(self.screen, str(i + 1) + '. ' + question, (30, 520 + 520 * (i / 10)))
        else:
            offsets = {}
            for j in range(i):
                global questionslist
                questionslist = list(randomlist[question_number])

                rect = self.smallwhitefont.rect(self.screen, questionslist[j], (0, 0))
                offsets[j] = rect[3] + sum(list(offsets.values()))
            self.smallwhitefont.render(self.screen, str(i + 1) + '. ' + question,
                                       (30, 520 + offsets[i - 1] + i * 5))
```

Now, finally, I developed a section of code which will provide a basic instruction to the user as to how to proceed the game, this by 'asking the right question' and they can expect a positive output in regard to their decision on the hot bar which they will be able to see.

If the user is idle and makes no choice by not providing the correct input to ask a question, a loop is entered.

The next section of code displays the three questions the player can pose in a randomised manner.

The first question is always rendered to a specific position on the screen. The subsequent questions must also be displayed. The final part of the code stores subsequent questions below the previous one.

This part was quite tough, and required a lot of help, as the text would not render properly.

Testing Hot – Bar mechanics alongside level mechanics:

What is being tested?	Input	Justification of my input	Outcome (as a result of input)	How to solve (if error occurs)
Hot Bar functionality increment	Excellent question selection	By selecting an excellent question, the hot bar should increment by a suitable amount	The hot bar fill increases by a suitable amount	NA

Hot Bar functionality decrease	Poor question selection	By selecting a poor question, the hot bar should decrease by a suitable amount	The hot bar fill decreases by a suitable amount	NA
Hot Bar stationary functionality	Neutral question selection	By selecting a neutral question, the hot bar fill should remain in a fixed position	The hot bar fill remains stationary, not moving up or down	NA

12/03/24: LEVEL SCREEN AND 'MUGSHOT' PLACEHOLDER FILL FOR CASE DETAIL

For my level screen which proceeds after the case detail screen, I decided the most convenient thing to do regarding time constraints was to implement the menu screen design but alternate the 'suspect' seated on the other end with the use of 3 sprites for the 3 levels.

In addition, I also added a mugshot holder on the case detail screen which will contain a basic headshot of the suspect.

For each of these features, I had to add images into my game file as seen below:

Mugshot pictures ranging from level 1 to level 3

Level screens 1 – 3

0

```
# Fill the screen with black color
self.screen.fill((0, 0, 0))
mpos = pygame.mouse.get_pos()
# Display the level background image
self.screen.blit(self.assets['level_' + str(self.level)][0], (self.screen.get_width() / 2 -self.assets['level_' + str(self.level)][0].get_width() / 2, 0))
```

This section of code is responsible for rendering the level image.

```
#Displays case details
def details(self):

    while True:
        self.screen.fill((0,0,0))  # Fill screen with black color
        self.screen.blit(self.assets['board'],(0,0))  # Display a detective board to blit the details on top of
        self.screen.blit(self.assets['mugshots'][self.level],(650,180))
```

This section of code is responsible for placing the mugshot images in the respective co-ordinate position where this designed placeholder for the image is:

Practical demonstration

Now that I had coded what was relevant to display the questions, the game scene, and the progress bar which would be implemented, I had to focus on making a solution which could process the player's input choice in terms of the question they select to ask the suspect. Then I need to develop a part of the solution which checks to see if the player has won or lost the level and then update the game state. Finally, I need to code event handling.

```
# Draw the progress bar
self.bar.draw(self.screen, ratio)

# Process player's choice and update game state accordingly
if choice != None:
    question_chosen = questionslist[choice]
    self.smallwhitefont.render(self.screen, self.levelquestions[self.level][question_number][question_chosen],
                               (30, 520))
    pygame.display.update()
    temp = list(self.levelquestions[self.level][question_number].keys())
    answer = 0
    for i, item in enumerate(temp):
        if item == question_chosen:
            answer = i - 1
    questions_answered += answer * -1
    question_number += 1

    time.sleep(3)  # Pause for 3 seconds after displaying answer
    choice = None
```

First, the hot bar is drawn, and to do this the draw method developed at prior stages is called. At the beginning of each level, the hot bar will be filled a fix amount.

If the choice does not equal to none, this means the player has made their question choice, hence their chosen question is rendered onto the screen and the display is updated to display the rendered text.

To update the game state:

- A temporary list temp is created from the keys of the current level questions from a dictionary. Keys represent possible choices for player.
- A for loop goes through each item in temp and if the current item matches the question chosen then the index for the answer is obtained, and the number of questions to ask reduces by 1.

- Then, question_number is incremented to move onto the next question.

The game is paused for three seconds, allowing the player to read the answer from the supect for that specific question choice.

Choice is set to 'None' again to indicate the player now needs to make a new choice for the next set of questions.

```
# Check if the player wins or loses the level and update game state
if ratio == 1:
    self.wins += 1
    self.level += 1
elif question_number > 5:
    self.losses += 1
    self.level += 1
```

At the end of the level, the program needs to determine if the player has won or lost the level as this will determine the final ending of the game: 'win' or 'lose'.

If ratio ==1 suggests the hot bar is full, and hence the player has won the game, and number of wins increment by 1. As a result, the player moves onto the next level.

However, if all 5 questions have been asked and the hot bar has not been filled, then the level is counted as a loss, and the number of losses increments by one.

```
# Event handling loop for user input and window events
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        pygame.quit()
        sys.exit()
    if event.type == pygame.KEYDOWN:
        # Process keyboard input for choosing answers
        if event.key == pygame.K_1:
            choice = 0
        if event.key == pygame.K_2:
            choice = 1
        if event.key == pygame.K_3:
            choice = 2
        if event.key == pygame.K_4:
            choice = None

self.clock.tick(60)  # Limit frame rate to 60 FPS
pygame.display.update()  # Update display
```

Above is the event handling loop which is crucial to allow the game to respond correctly to particular user input.

- the game is exited cleanly if the user closes the game window.

- if event.type — pygame.KEYDOWN checks if a key has been pressed down. The user may select the keyboard numbers [1,2 or 3] to select a question to pose in the level.

NOTE: initially, I did intend the game to just be a point and click system throughout its entirety. However, I found it much simpler in terms of coding the solution to register a keyboard input for the question choice selected by the user rather than having to a mouse click, because:

1. Collision detection: the game needs to detect whether the user has clicked within the boundaries of an interactive element I need to create on top. Having to do this in the main menu, I thought it would be simpler later on and more convenient to implement a keyboard input system instead, as I am aware of time constraints.

Final Result:

Testing the level mechanics when game is in playing state:

What is being tested?	Input	Justification of my input	Outcome (as a result of input)	How to solve (if error occurs)
Suspect response	KEY '1'	This selects the first question the player can pose	The suspect responds in the intended manner	NA
Suspect response	KEY '2'	This selects the second question the player can pose	The suspect responds in the intended manner	NA
Suspect response	KEY '3'	This selects the third question the player can pose	The suspect responds in the intended manner	NA
Pause menu	M button	This button should trigger the pause menu to popup in a playing state	The game does pause, and the correct sound associated to the pause menu plays, however the screen goes dark	Finding solution:
Soundtrack	Clicking play button	The play button triggers the new playing game state which has its own respective sound track	The correct soundtrack plays as intended.	NA
Case pop-up	Clicking play button	The play button along with triggering the new game track also triggers an automatic screen of the case details of a suspect	The case detail screen pops up	
Case detail to level progression	Clicking space button	The space button as instructed continues to the level	The level starts as intended	NA

Playing

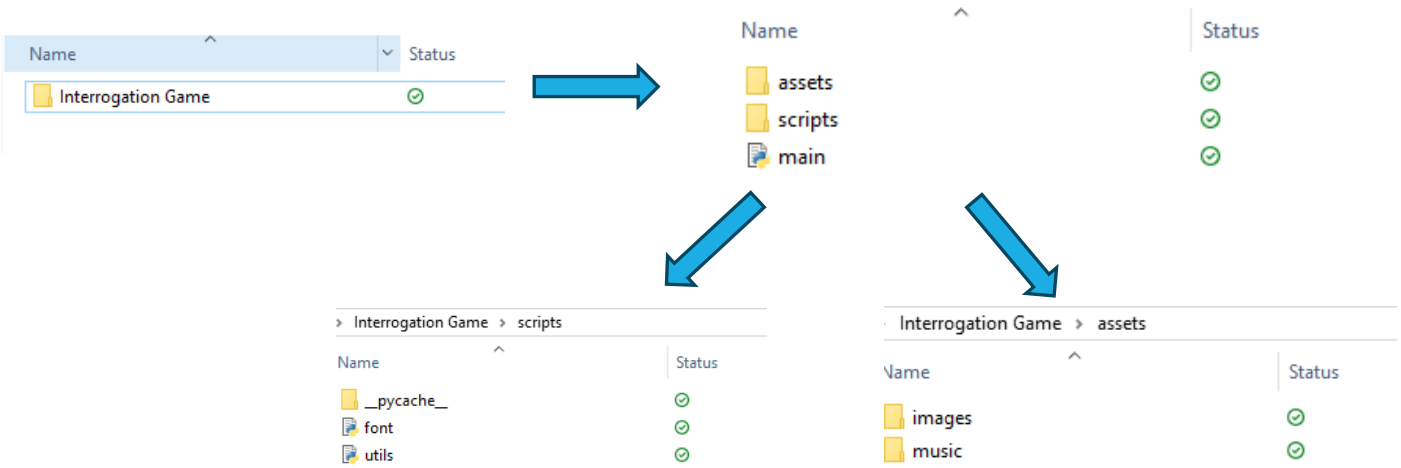
TEST DATA	TYPE		Explanation
LMC on question boxes	Invalid		The LMC on question boxes is now invalid. Use NUM keys 1,2,3 to trigger question.

'm' button click to pause	valid		
Clicking 'm' key whilst game_state = running	Valid		
Clicking any other key (other than m) whilst game_state = running e.g 'k'	Valid		Player clicks 1,2,3 to ask question
'esc' button clicked whilst game_state = paused	valid		
Clicking any other button (other than esc) whilst game_state = paused e.g 'o'	Invalid		

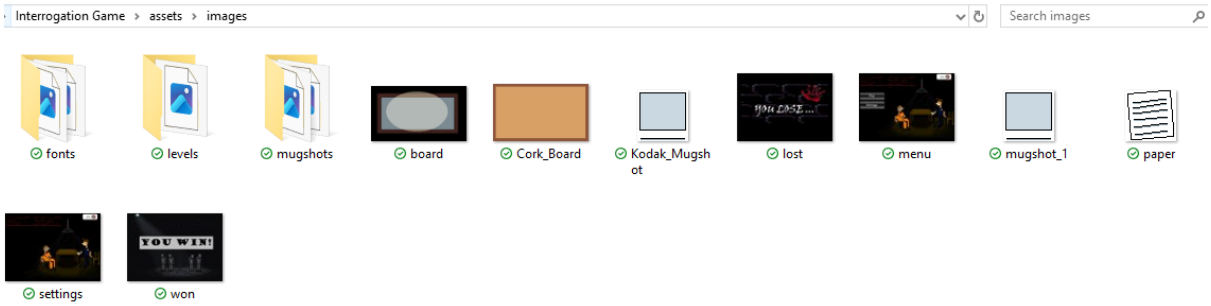
Featured above is my original test table. Some changes did have to be made, as I did avert away from the original way posed in earlier parts of documentation of how the questions will be triggered.

15/03/24: SHOWING MY UPDATED GAME FOLDER

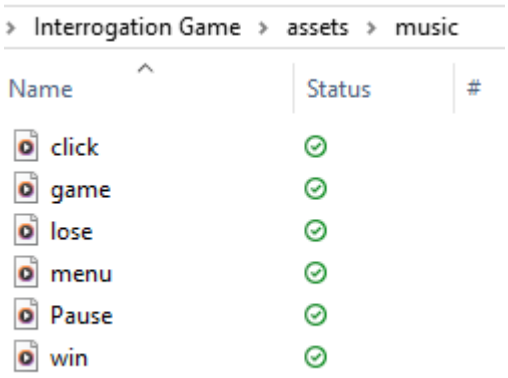
Throughout the development I added different files with different purposes to my game folder. However, I decided to make it better structured. Below are images of the game folder.



Images folder:



Music folder:



16/03/24: OVERCOMING KEY PROBLEMS -> TRANSITION BETWEEN LEVELS, WIN/LOSE ENDING SCREEN, TRYING TO FIND A SOLUTION

1. I had to solve how I would transition between the different levels in the game, something I did not consider before. From there, I had to incorporate a feature which will lead to a win or lose end screen, and the mechanics regarding this.

I decided to implement a fade out feature, along with displaying a message to the user if they have won or lost the level. In addition, within this section I also code the two endings the user can obtain 'win' or 'lose' and also an automatic system exit if no user input.

This required the coding of a new section of code:

```
    def eventHandler(self,ratio,action):
        if self.music:
            if self.won or self.lost:
                pygame.mixer.fadeout(100)
                self.SFX[action].play()
        counter = 0

        s = pygame.Surface(self.size, pygame.SRCALPHA)
        while True:
            # Fill the screen with black color
            self.screen.fill((0, 0, 0))
            # Draw the progress bar
            self.bar.draw(self.screen, ratio)
            if not self.won and not self.lost:
                self.screen.blit(self.assets['level_' + str(self.level)][0], (self.screen.get_width() / 2 -self.assets['level_' +
                str(self.level)][0].get_width() / 2, 0))
                rect = self.bigwhitefont.rect(self.screen,'You '+ action,(0,0))

                s.fill((0,0,0,counter if counter < 256 else 255-counter%255 if counter > 500 else 255))
                if counter > 255:
                    self.bigwhitefont.render(self.screen,'You '+ action,(self.size[0]/2 - rect[2]/2,50))
                if counter > 255*2:
                    self.level +=1
                    self.details()
                self.screen.blit(s, (0,0))
                counter += 1
            else:
                self.screen.blit(self.assets['level_' + str(self.level)][0], (self.screen.get_width() / 2 -self.assets['level_' +
                str(self.level)][0].get_width() / 2, 0))
                rect = self.bigwhitefont.rect(self.screen,'You '+ action,(0,0))

                s.fill((0,0,0,counter if counter < 256 else 255-counter%255))
                if counter > 255:
                    self.bigwhitefont.render(self.screen,'You '+ action,(self.size[0]/2 - rect[2]/2,50))
                if counter > 255:
                    self.screen.blit(self.assets[action],(0,0))
                if counter//60 > self.time:
                    pygame.quit()
                    sys.exit()
                self.screen.blit(s, (0,0))
                counter += 1
```

```
def eventHandler(self,ratio,action):
    if self.music:
        if self.won or self.lost:
            pygame.mixer.fadeout(100)
            self.SFX[action].play()
```

This section is responsible for the music handling between the transition of levels. Regardless if the user has won or lost the level, the current music track playing fades out.

```
        counter = 0

        s = pygame.Surface(self.size, pygame.SRCALPHA)
```

I refer to this part of the code as the 'initialisation' necessary to set up the appropriate variable and surface.

The counter is necessary to track the timing of events, and in this context for tracking the timing for certain actions regarding the transition.

's = ….' Is necessary to allow me to create a new surface with the same dimensions as the game window (self.size) and (pygame.SRCALPHA) allows for transparency effects used in the fade.

```
while True:
    # Fill the screen with black color
    self.screen.fill((0, 0, 0))
    # Draw the progress bar
    self.bar.draw(self.screen, ratio)
    if not self.won and not self.lost:
        self.screen.blit(self.assets['level_' + str(self.level)][0], (self.screen.get_width() / 2 -self.assets['level_' +
        str(self.level)][0].get_width() / 2, 0))
        rect = self.bigwhitefont.rect(self.screen,'You '+ action,(0,0))

        s.fill((0,0,0,counter if counter < 256 else 255-counter%255 if counter > 500 else 255))
        if counter > 255:
            self.bigwhitefont.render(self.screen,'You '+ action,(self.size[0]/2 - rect[2]/2,50))
        if counter > 255*2:
            self.level +=1
            self.details()
        self.screen.blit(s, (0,0))
        counter += 1
```

'if not self.won and not self.lost' indicates that the game is still ongoing:

There are 5 questions in the level. If 3/5 of the questions are answered in a way such that the hot bar is decreased to 0 it is no longer possible for the player to win the level as 60 percent of the questions must be answered as 'excellent' for a win. Hence, the game fades out as such, and fades back in to display the 'You have lost' message on the screen. The opposite is as such: If the player has reached the 4th question whilst the past three did not cause the hot bar to decrease completely then the game poses the 4th and 5th question as normal. After the 5th question, if the hot bar is filled then the won message is displayed, and if the hot bar is only filled partially, then the lost message is displayed.

The level is incremented, and the case details for the next level pop up on the screen if it wasn't the last level that had just finished.

```
    else:
        self.screen.blit(self.assets['level_' + str(self.level)][0], (self.screen.get_width() / 2 -self.assets['level_' +
        str(self.level)][0].get_width() / 2, 0))
        rect = self.bigwhitefont.rect(self.screen,'You '+ action,(0,0))

        s.fill((0,0,0,counter if counter < 256 else 255-counter%255))
        if counter > 255:
            self.bigwhitefont.render(self.screen,'You '+ action,(self.size[0]/2 - rect[2]/2,50))
        if counter > 255:
            self.screen.blit(self.assets[action],(0,0))
        if counter//60 > self.time:
            pygame.quit()
            sys.exit()
        self.screen.blit(s, (0,0))
        counter += 1
```

This final section is about the win/loss ending screen and the automatic game closure:

The respective image for win/loss is displayed on the screen. There is another fade in effect here leading on to the ending screen.

My game displays the ending for 10 seconds, and then if the user has not already manually closed the game by this point, then there is an automatic [pygame.quit() and sys.exit()] which closes the game automatically.

What is being tested?	Input	Justification of my input	Outcome (as a result of input)	How to solve (if error occurs)
Fade transition between levels	Finishing the level	The fade should occur at the end of a level and before the start of the next	A successful fade is implemented, fading out and then fading back in to display if user has won or lost	NA
Fade transition to ending screen after last level	Finishing the last level	Ending screen follows the last level	The game fades smoothly into a WIN/LOSE ending screen	NA

Proof of 'You have won' and 'You have lost' message at the end of each level:

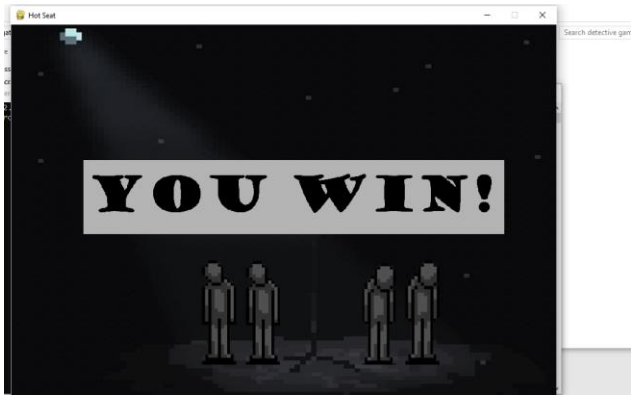
Hot bar filled = level won = 'You won' message displayed

Hot bar empty = level lost = 'You lost' message displayed

All 5 questions asked, but hot bar partially filled = level lost = 'You lost' message displayed

Proof of the win screen and lose screen at the end of the game:

Lose Ending if 2/3 levels have been lost.



Win Ending if at least 2/3 levels have been won.

What is being tested?	Input	Justification of my input	Outcome (as a result of input)	How to solve (if error has occured)
Win ending screen	Following question path in a manner such that a minimum of two out of three levels are won	The game requires a minimum of 2/3 levels won to obtain the win ending	Win ending screen loaded	NA
Lose ending screen	Following question path in such a manner that at least 2/3 levels are lost	The game requires a minimum of 2/3 levels lost to trigger the lose screen ending.	Lose ending screen loaded	NA
Automatic system exit	No user input	The system should perform an automatic exit, the game should close on its own without any user input	The game successfully exits on its own after about 7-10 seconds.	NA

Ending screen

TEST DATA	TYPE
All keyboard + mouse input e.g (RMC, LMC, 'd',)	Invalid ✓
No user input (due to automatic ending timeout)	Valid ✓

19/03/24: FIRST BETA TEST ON FIRST PROTOTYPE, REFERRING TO CLIENT

A prototype for the entire game had been created and I referred to my end-user, F.R, to play the game. I observed game play and noted down any issues.

Errors / Issues [what I observed]
1. **Text replies from suspect in response to questions posed is sometimes too quick to read**
2. **Majority of game sounds I implemented do not turn out to be suitable, reducing game quality. The ending screen sounds play for longer than the time for the system exit to occur.**

Remedial Action I took to resolve problems:

1. I observed that when the suspect answers were slightly longer, the player didn't have adequate time to read the entire reply. I decided to take on a simple fix:

```
# Process player's choice and update game state accordingly
if choice != None:
    question_chosen = questionslist[choice]
    self.smallwhitefont.render(self.screen, self.levelquestions[self.level][question_number][question_chosen],
                               (30, 520))
    pygame.display.update()
    temp = list(self.levelquestions[self.level][question_number].keys())
    answer = 0
    for i, item in enumerate(temp):
        if item == question_chosen:
            answer = i - 1
    questions_answered += answer * -1
    question_number += 1

    time.sleep(3)  # Pause for 3 seconds after displaying answer
    choice = None
```

Increasing the value stored within 'time.sleep()' leads the reply to remain on the screen for a longer time. By increasing the value, and trying the game again, I deduced that the replies were on the screen for long enough for the majority of players to read comfortably.

2. For my second problem, I decided to find a better resource for my game sounds. I decided to implement sound effects of games of similar ambience/setting uploaded to the YouTube platform. I spent more time listening to different sounds, and compiled a basic note on my device:

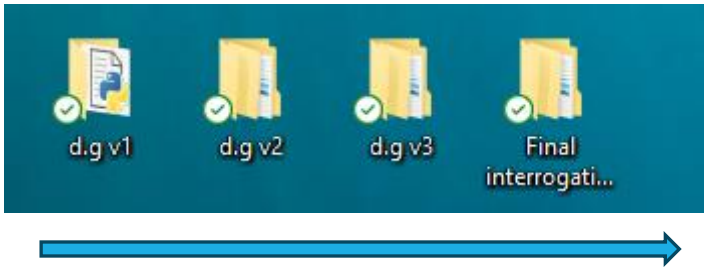
My new win screen and lose screen sounds derived from a game called 'Outlast' with a tense and horror ambience. Not only did this relate to my game, but the average length of the sound effects varied around the ten second outro mark for my two endings, hence I deduced it was best to use this as a resource.

Used an online YouTube to MP3 converter and replaced the original music files to change. Upon testing again, the new sounds complemented the system exit timing and ambience much better

What is being tested?	Input	Justification of my input	Outcome (as a result of input)	How to solve (if error occurs)
Suspect answer response time	Posing a question with NUM keys 1,2,3	This allows the player to pose a question and the suspect replies	The suspect's response stays slightly longer on the screen	NA
Correct new sound track integration	Completing all three levels	After the three levels the ending screen is displayed	New sound tracks play correctly	NA

21/03/24: REFLECTING ON THE 4 PROTOTYPES

Over the short sprint of development for my game, subtle changes were made to the main source code behind the game. Subtle changes had to be made to avert errors/bugs which arose. On the leftmost side of the attachment below represents the initial prototype made and on the rightmost side represents the final game. I will be reflecting over the whole course of development, hence may repeat some solutions already mentioned prior so these will be referenced.



The game does require the pygame module to be installed on the device it is being run on.

Prototype 1:

Errors/problems I found:

Error/Problem Identified	Brief Explanation
Default music state is set to 'off'	In the settings menu, the music state is set to off in a natural state which opposes the natural game setting which is to be turned on.
Too quick text replies from suspect	The text replies from the suspect occur too quickly for lengthy answers
Win/Lose screen game state transition is poor	The win/lose screen suddenly cuts in instead of a smooth transition.
Click sound effect not incorporated properly	There was no click sound working on the menu screen when the user hits LMB
Issue with level progression to game ending	If the user lost 2 out of the three levels, then the game would jump to the ending instead of allowing the player to play the final level.

What was solved and the action taken to do so:

Level issue was fixed – allows player to progress through ALL the three levels regardless of the fact that even if they have lost the first two.

How?

Text replies from suspect were made slightly longer.

Prototype 2:

Error/Problem identified	Brief Explanation
Music state problem remains	As described above
Still need to add a click sound effect and add code for a click sound effect	Still had to import a click sound and code it into the game
Ending screen sound was not suitable	The sounds had to be switched out as they didn't match the game well
Ending screen was displayed to the user for an infinite period	The ending screen was up for an infinite amount of time. Such to the point that the sound associated with it would eventually run out.
Jumping to ending suddenly after last level	Had to incorporate a fade.

What was solved and the action taken to do so: 

Sound effect / background music tracks were easily fixed by importing new sound files from a better resource.

Ending screen infinite time was fixed by adding a simple section of code which performed an automatic system exit after a fixed period of short time, similar to that of the win/lose sound effect.

Fade added to the ending, however as seen below this did have problems.

Music state was now set to 'on' by default.

Prototype 3:

Error/Problem identified	Brief Explanation
Buggy settings menu	Interacting with the ON and OFF settings for the music and sfx was very buggy, the option flickering between the two states if interacted with
Fade problem	The fade incorporated in the playing state was not working as intended. It would very briefly display

	'You have won' or 'You have lost' before jumping to the next level. The fade in for the ending screen worked as intended, however it would continue fading in and out several times.

What was solved and the action taken to do so:

The settings menu was reformed to its original state in earlier prototypes which allows for smooth interaction.

The fade problem was fixed such that a smooth single fade played leading on to the ending screen.

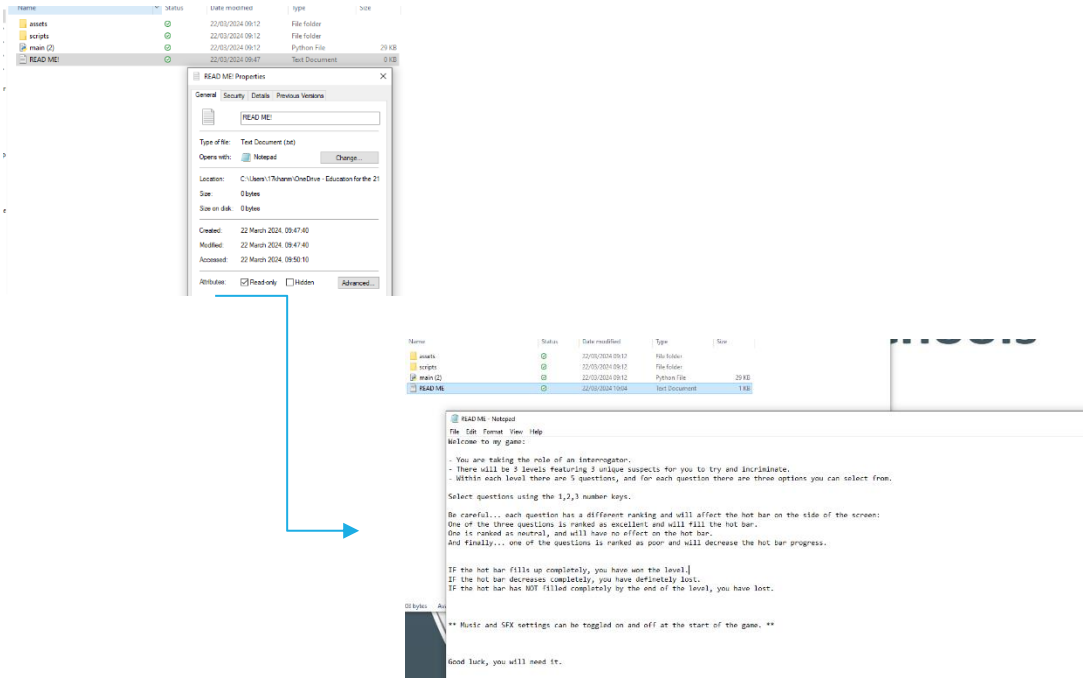
Prototype 4: All problems had been resolved and I was content with this version.

22/03/24: MAKING THE GAME READY FOR DELIVERY

In many indie developed games – games created by single developers or smaller teams - a note is often added within the game file to go over general instructions / controls the player may need to be aware of that are not explicitly explained.

I decided to implement this:

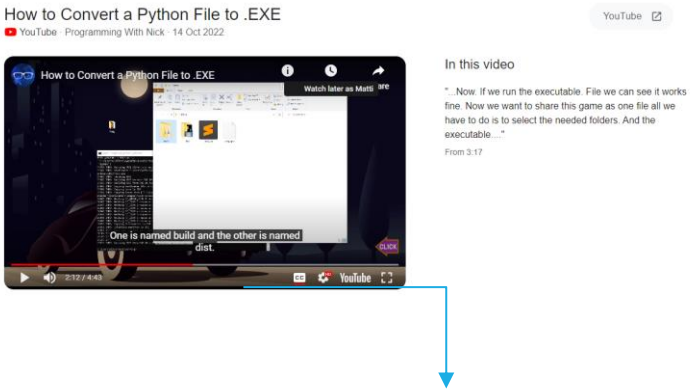
1. First, I added a .txt file to the main game folder, making it easy to see and attention capturing through the title it is saved as, encouraging the player to read it first.
2. I had to ensure the .txt file was set to read-only so it could not be changed by anyone other than me.

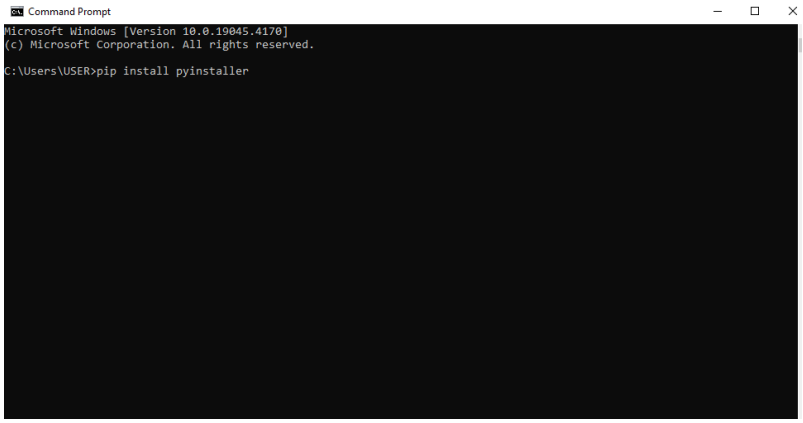


As the game file responsible for executing the game is a '.py' file, it means certain modules must already be installed on the system which are trying to run the game.

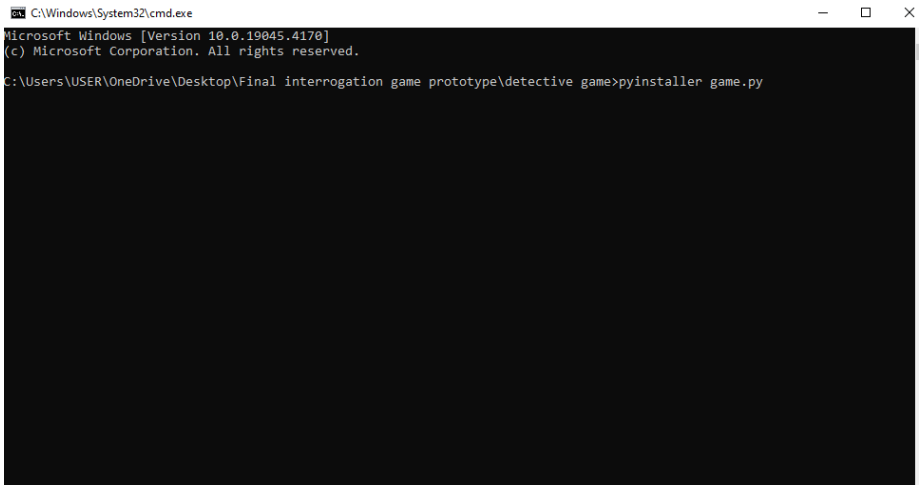
It would be largely inconvenient to distribute the game file to end users and have them require installing pygame and such to run the game.

Hence, I had to convert the .py file to a .exe file. I didn't know how to do this, hence used the following YouTube video. As a result, I used pyInstaller.





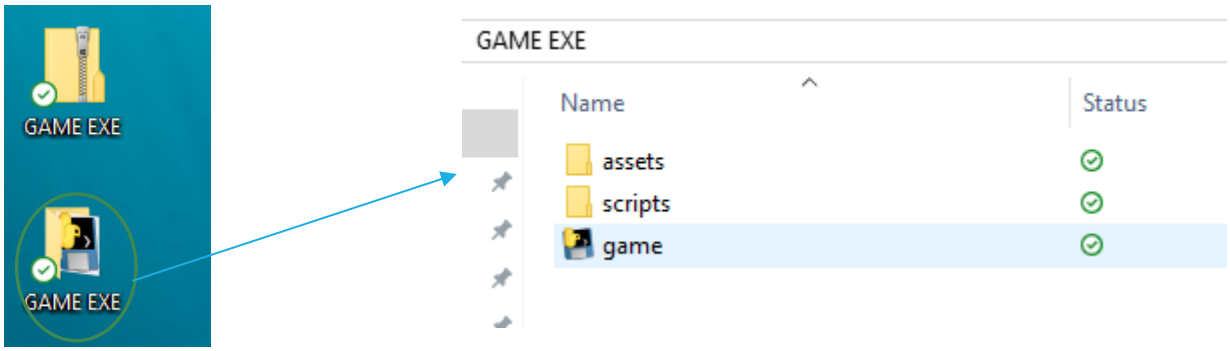
I had to first install 'pyinstaller,' a handy python package for my case, and I done this via the cmd terminal on my laptop. This was the command I ran. After successful installation, I then moved onto the next step.



I then set the cmd terminal to point to the file which contained my .py file. I ran the next command 'pyinstaller game.py.'

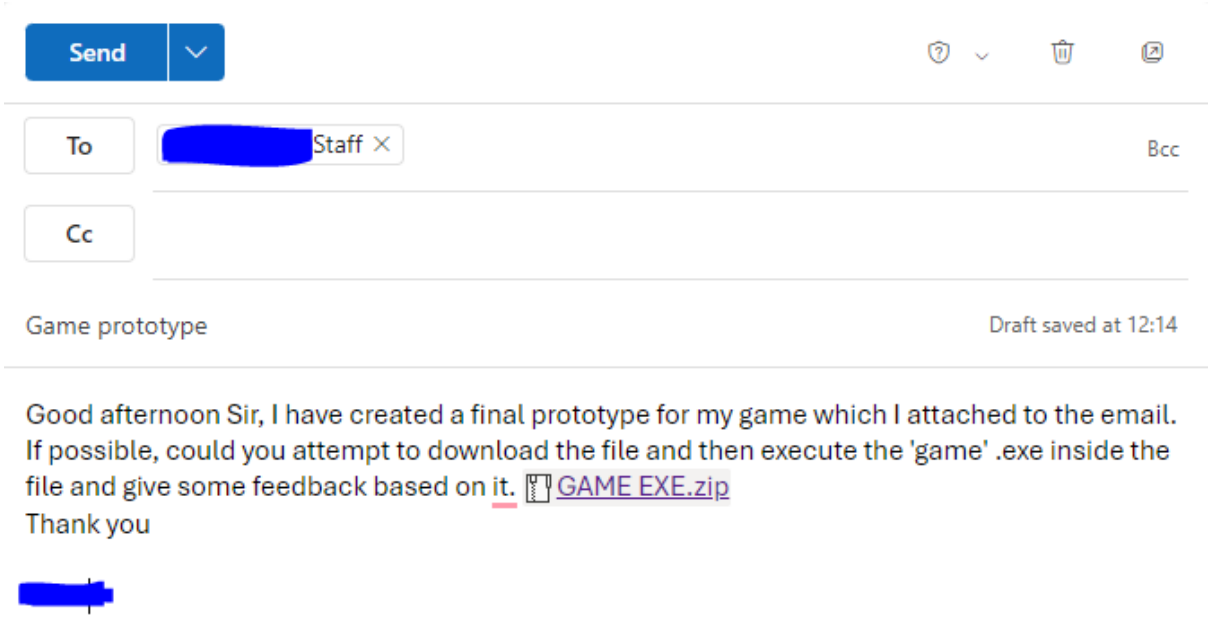
This in turn will allow the package to read my python scripts, and then create folders and an executable that users can run without any other extra installation.

Within this 'dist' file was the executable to run my script. I had to copy the executable into the exact location my .py script was located. I then deleted all extra folders created by the pyInstaller module which were not necessary and tested if the .exe would start my game. It indeed did, hence I created a compressed folder including the .exe file for my game, which I could send to users. This is how it looks below:

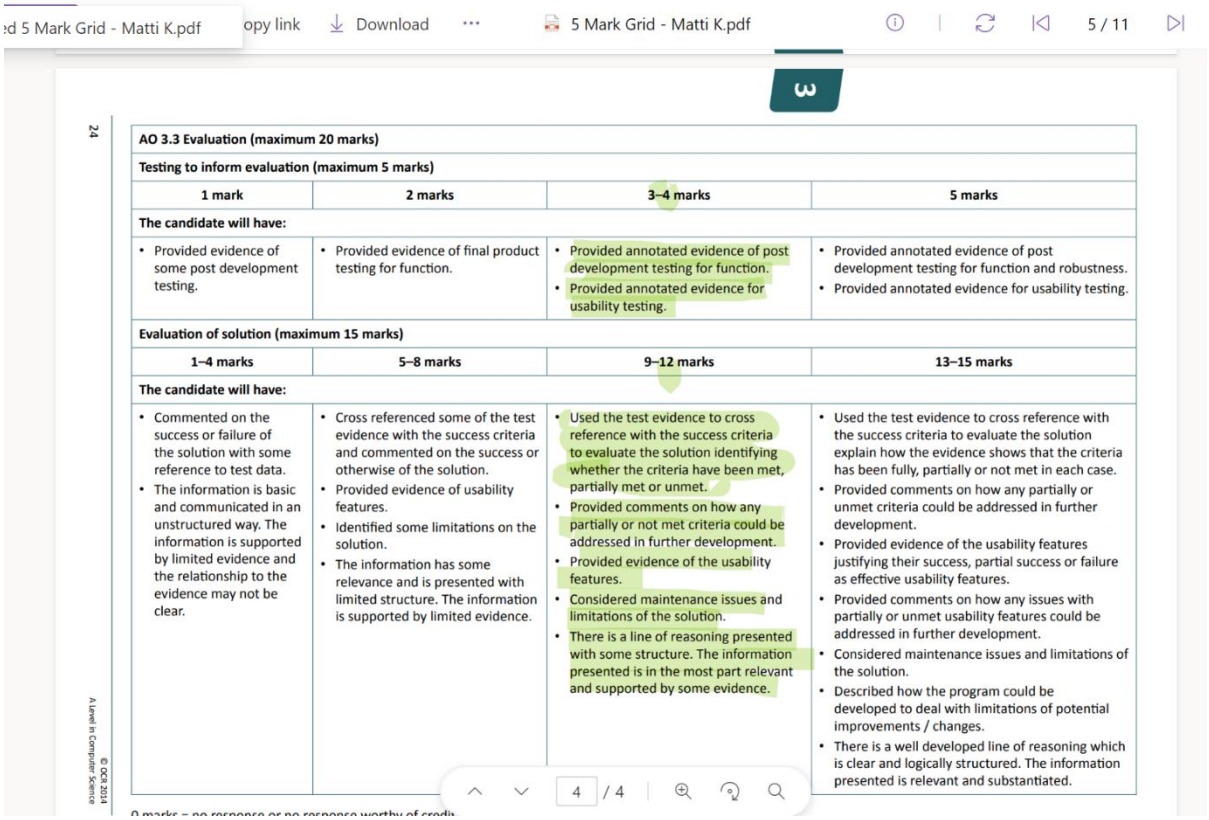


As my end-user/client had already played the game, I decided to send the executable to my teacher to see if it would run as intended and collect some feedback.

What is being tested?	Input	Justification of my input	Outcome (as a result of input)	How to solve (if error occurs)
Game Execution with .exe file	Double click on the game.exe application	This should trigger my game to start and operate just like how it does as when I execute the game in its .py file format.	The game successfully executes without requiring modules such as pygame to run.	NA


Send
To
[redacted] Staff
Bcc
Cc
Game prototype
Draft saved at 12:14
Good afternoon Sir, I have created a final prototype for my game which I attached to the email. If possible, could you attempt to download the file and then execute the 'game' .exe inside the file and give some feedback based on it. GAME EXE.zip
Thank you
[redacted]

EVALUATION



ed 5 Mark Grid - Matti K.pdf opy link Download ... 5 Mark Grid - Matti K.pdf 5 / 11

3

24

AO 3.3 Evaluation (maximum 20 marks)			
Testing to inform evaluation (maximum 5 marks)			
1 mark	**2 marks**	**3–4 marks**	**5 marks**
The candidate will have:			
• Provided evidence of some post development testing.	• Provided evidence of final product testing for function.	• Provided annotated evidence of post development testing for function. • Provided annotated evidence for usability testing.	• Provided annotated evidence of post development testing for function and robustness. • Provided annotated evidence for usability testing.
Evaluation of solution (maximum 15 marks)			
1–4 marks	**5–8 marks**	**9–12 marks**	**13–15 marks**
The candidate will have:			
• Commented on the success or failure of the solution with some reference to test data. • The information is basic and communicated in an unstructured way. The information is supported by limited evidence and the relationship to the evidence may not be clear.	• Cross referenced some of the test evidence with the success criteria and commented on the success or otherwise of the solution. • Provided evidence of usability features. • Identified some limitations on the solution. • The information has some relevance and is presented with limited structure. The information is supported by limited evidence.	• Used the test evidence to cross reference with the success criteria to evaluate the solution identifying whether the criteria have been met, partially met or unmet. • Provided comments on how any partially or not met criteria could be addressed in further development. • Provided evidence of the usability features. • Considered maintenance issues and limitations of the solution. • There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence.	• Used the test evidence to cross reference with the success criteria to evaluate the solution explain how the evidence shows that the criteria has been fully, partially or not met in each case. • Provided comments on how any partially or unmet criteria could be addressed in further development. • Provided evidence of the usability features justifying their success, partial success or failure as effective usability features. • Provided comments on how any issues with partially or unmet usability features could be addressed in further development. • Considered maintenance issues and limitations of the solution. • Described how the program could be developed to deal with limitations of potential improvements / changes. • There is a well developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated.

A Level in Computer Science © OCR 2024

4 / 4

0 marks = no response or no response worthy of credit

NAVIGATION

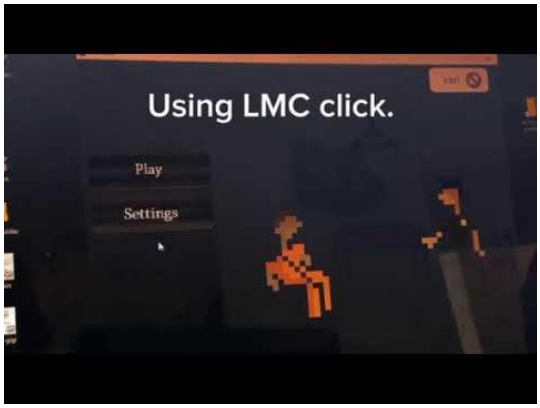
This section of test is about the navigation implemented through the game.

Test Number	What is being tested?	Input/Action	Expected Outcome	Actual outcome	Has the requirement been reached?	Video evidence reference
1	Navigation in the main menu screen	VALID: The left mouse button click will be pressed on all the buttons in the menu.	VALID: The button selected allows the game to proceed in the intended manner	VALID: The game responds in the correct manner to each respective button.	YES	1.1
		INVALID: Left mouse button click where there are no buttons	INVALID: No change in game state should occur.	INVALID: No change in game state occurs.	YES	1.2
		BORDERLINE: Left mouse button click on the very edge of the menu buttons	BORDERLINE: The game should still recognise the input if within the collision area of the	BORDERLINE: Although the input is borderline, the game still recognises	YES	1.3

			button and proceed in the intended manner.	the interaction with the menu button and responds.		
2	Pause feature in playing state	VALID: Whilst the game is in a playing state, the 'p' button is clicked and the 'esc' button is clicked to return to playing state	VALID: The game should enter the paused state	VALID: The game enters the paused state with the correct sfx, however a black screen is loaded and the method of returning to playing state is by clicking the same button to pause. The esc button does not work to go back to playing state. In addition, when unpausing, the level restarts.	NO	1.4
		INVALID: Pressing the 'k' key as the game is playing	INVALID: The game should continue playing	INVALID: The game continues playing	YES	1.5
3	Selecting questions to progress through level	VALID: Clicking the 1, 2, or 3 NUM key	VALID: The question is selected and posed to the suspect	VALID: The question is successfully selected and posed to the suspect, in which the respective response for that particular question is given.	YES	1.6
		INVALID: Left mouse click on the questions on the screen	INVALID: Although this feature was originally meant to be implemented, there should be no effect.	INVALID: No effect, player must selected the NUM keys	YES	1.7

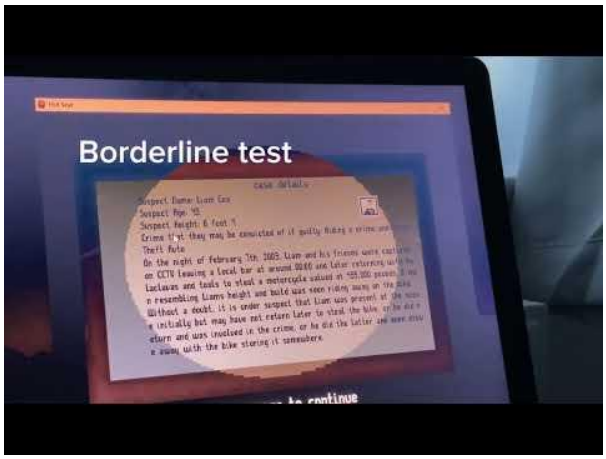
Video Reference hyperlinks for Navigation testing: Covers 1.1 -> 1.2

Navigation Testing.mp4



Video Reference hyperlinks for Navigation testing: Covers 1.1 -> 1.3

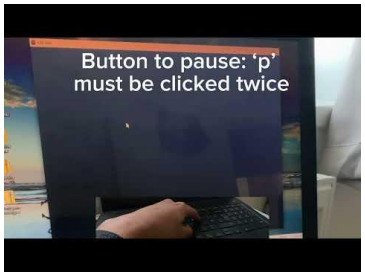
Navigation borderline test.mp4



Video Reference hyperlinks for Navigation testing: Covers 1.4 -> 1.5

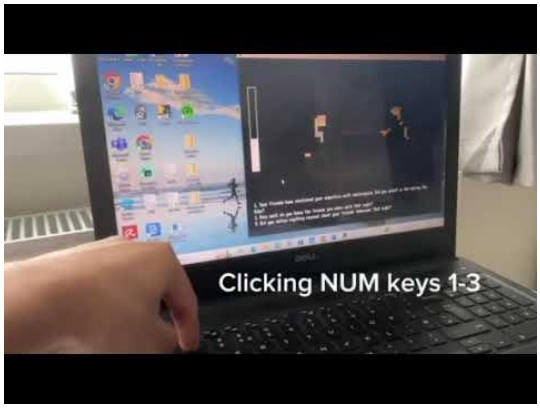
Navigation Testing 2.mp4

Navigation testing 3.mp4



Video Reference hyperlinks for Navigation testing: Covers 1.6 -> 1.7

Navigation testing 4.mp4



GAME STATE CHANGES

This section of testing is conducted on the transition between game states

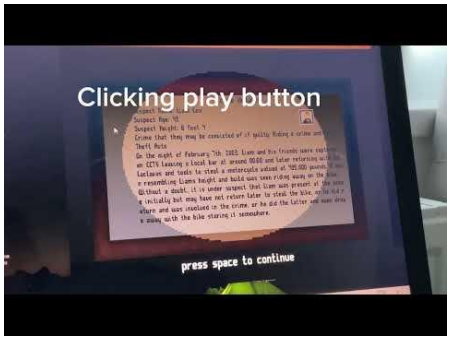
Test Number	What is being tested?	Input/Action	Expected Outcome	Actual outcome	Has the requirement been reached?	Video evidence reference
3	Transition from menu to playing state	VALID: Clicking the play button at the main menu	VALID: The game state changes to playing and the case detail screen appears.	VALID: The game state changes to playing, and the game transitions to the playing	YES	1.5

				state, displaying the case detail screen.		
		INVALID: Pressing the space button at main menu	INVALID: No effect, the game remains on the main menu.	INVALID: The game remains on the main menu with no effect.	YES	1.6
4	Transition of game state from the main menu to the settings screen.	VALID: The settings button is clicked with LMC on the main menu.	VALID: The setting screen is presented, allowing the user to tweak settings.	VALID: The setting screen is shown, and the user can successfully toggle two displayed settings.	YES	1.7
		INVALID: Clicking the 'm' button which triggers a similar pause menu feature in a playing state.	INVALID: No change should occur, and the main menu screen is still displayed.	INVALID: No change occurs	YES	1.8
5	Closing the game down via main menu	VALID: Clicking the exit button at main menu	VALID: The game should close	VALID: The game closes successfully.	YES	1.9
		INVALID: Clicking the 'esc' button at the main menu	INVALID: The game should not close, no change.	INVALID: The game does not close down.	YES	2.0
6	Transition of game from the case detail screen to the playing the levels	VALID: Clicking the 'space' button as informed	VALID: The game transitions to the beginning of the first level.	VALID: Game successfully transitions to the first level.	YES	2.1
		INVALID: Clicking the 'enter' button	INVALID: The case detail screen should remain on display.	INVALID: The case detail does indeed remain on display infinitely until the space	YES	2.2

				button is clicked.		
7	Transition of game to the WIN/LOSE ending screen.	VALID: Complete the last level	VALID: Depending on number of levels won, the win or lose screen fades in.	VALID: The win screen loads in if 2/3 levels won and lose screen loads in if 2/3 levels lost.	YES	2.3
		INVALID: Winning the first two levels or losing the first two levels	INVALID: The game should not display the ending screen even if the first two levels have been won already, or the first two levels have been lost already	INVALID: Regardless of if the first two levels are won or lost, the game proceeds on to the last level, allowing the user to play all three levels.	YES	2.4

Video Reference hyperlinks for Game State changes testing: Covers 1.5 -> 1.6

Game state testing 1.mp4

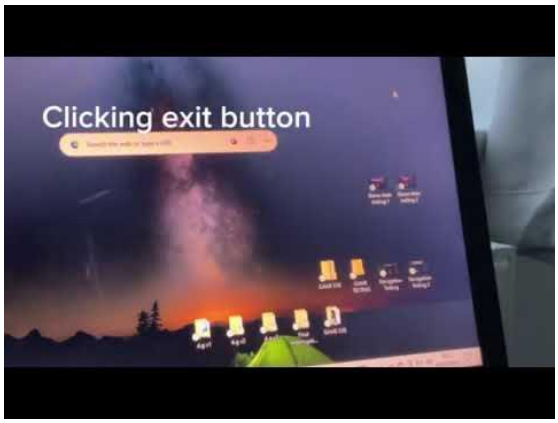


Video Reference hyperlinks for Game State changes testing: Covers 1.7 -> 1.8

Game state testing 2.mp4

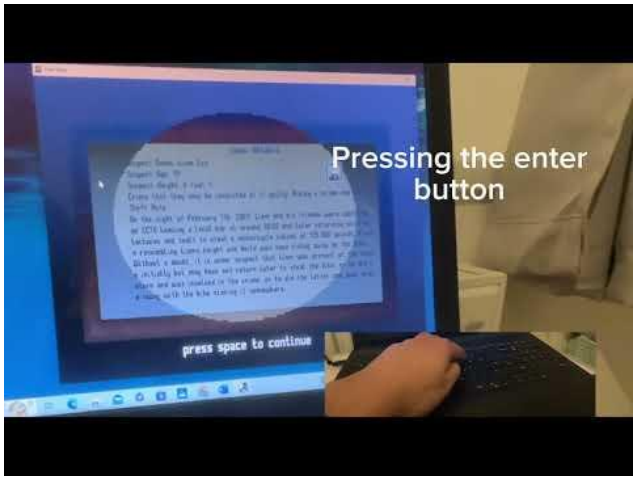
Video Reference hyperlinks for Game State changes testing: Covers 1.9 -> 2.0

Game state testing 3.mp4



Video Reference hyperlinks for Game State changes testing: Covers 2.1 –> 2.2

Game testing 4.mp4



Video Reference hyperlinks for Game State changes testing: Covers 2.3 -> 2.4

Game testing 5.mp4

Game state testing 6.mp4

Note: The test above represents how even though I purposely won the first two levels, the game did not end and successfully proceeded to the last level, hence I've shown both requirements are satisfied.

The first video shows the win screen scenario, and the second video shows the lose screen scenario.

SCORING SYSTEM TESTING

The test ran below will be in regard to the scoring system applied to my game.

Test Number	What is being tested?	Input/Action	Expected Outcome	Actual outcome	Has the requirement been reached?	Video evidence reference
8	Hot Bar increment and associated level outcome	VALID: Selection of an excellent ranked question	VALID: The hot bar fill should increment by a fixed portion from its current position. Level is won by 3 excellent choices.	VALID: The hot bar does indeed increment by a fixed amount, such that if I was to pose 3/5 excellent questions, I would win the level, which is what is intended.	YES	2.5
		INVALID: Selection of a poor ranked question	INVALID: The hot bar should not increment, but rather the fill should decrease by a fixed amount, such that if two poor questions are posed in succession the level is lost.	INVALID: The hot bar does indeed decrease by a fixed amount, such that if I was to pose two poor questions the level is lost.	YES	2.6
		BORDRLINE: Selection of a neutral ranked question	BORDERLINE: This input is still recognised,	BORDERLINE: No effect registered on hot bar,	YES	2.7

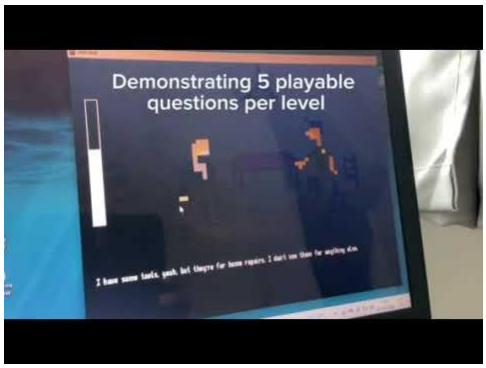
			however should have no effect on the hot bar.	but input is processed and proceeds on to next question.		
9	Number of available questions	VALID: Value assigned to variable total_questions variable should equal 5.	VALID: Should be 5 available questions at max for each level	VALID: As expected	YES	2.75
		INVALID: Any other value assigned to this variable	INVALID: When the game is initialised, the variable assigned to number of questions would be set to value other than 3	INVALID: As expected, this would not cause the game to function as intended.	NA	-

Video Reference hyperlinks for Scoring System testing: Covers 2.5 -> 2.7

Scoring System testing.mp4

Video Reference hyperlinks for Scoring System testing: Covers 2.75

Scoring System 2.mp4



SOUND TESTING

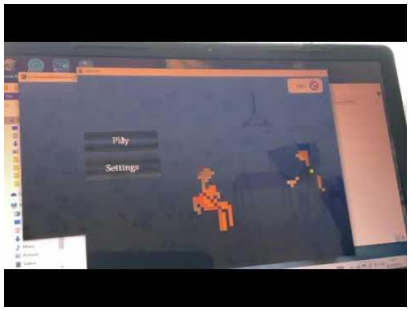
Test Number	What is being tested?	Input/Action	Expected Outcome	Actual outcome	Has the requirement been reached?	Video evidence reference
10	Main Menu screen soundtrack	VALID: The game is executed	VALID: When the game initialises upon execution, the main menu track should automatically play	VALID: As expected, the main menu track is automatically set to an 'on' state and begins playing.	YES	2.8
		INVALID: No game execution	INVALID: The game does not initialise, hence no sound track can play at the main menu.	INVALID: As expected	YES	NA
11	Main menu button interaction sound effect	VALID: Clicking on a menu button	VALID: The beep sound effect should be triggered.	VALID: The sound effect plays successfully	YES	2.9
		INVALID: Idling on the main menu, no button interaction.	INVALID: The sound effect should not play.	INVALID: As expected, the sound effect is only triggered by user input as Left mouse	YES	NA

				click on a menu button.		
		BORDERLINE: Clicking on the 'quit' menu button	BORDERLINE: Although this is a valid menu button, the interaction should not trigger the sound effect.	BORDERLINE: As expected, the sfx does not play.	YES	3.0
12	Playing State soundtrack	VALID: Selecting the play button to start the game and enter playing state	VALID: The soundtrack should play associated with playing state	VALID: As expected, the correct soundtrack plays straight away	YES	3.1
		INVALID: Idling on the main menu or selecting any menu button other than the play button	INVALID: As the game has not entered playing state, the soundtrack should not play	INVALID: As expected, the soundtrack does not play.	YES	NA
13	Pause menu soundtrack	VALID: Clicking the 'm' button whilst game is in playing state	VALID: Soundtrack associated with pause menu should play up until the user exits the pause menu	VALID: As expected, the soundtrack plays.	YES	3.15
		INVALID: Not accessing the pause menu via the assigned button whilst in a playing state	INVALID: The soundtrack does not play until user accesses pause menu.	VALID: As expected.	YES	NA
14	Win screen soundtrack	VALID: Obtaining the win ending = winning 2 levels minimum.	VALID: The win screen soundtrack should play when it loads up	VALID: As expected.	YES	3.2
		INVALID: Not obtaining the win ending via means of not completing all levels, or obtaining lose ending.	INVALID: The win sound track does not play	INVALID: As expected.	YES	NA

15	Lose screen soundtrack	VALID: Obtaining the lose ending = losing two levels minimum	VALID: The lose screen soundtrack should play when it loads up	VALID: As expected.	YES	3.3
		INVALID: Not obtaining the lose ending via means of not completing all levels or obtaining win ending.	INVALID: The lose screen soundtrack should not play	INVALID: As expected.	YES	NA
16	Turning off SFX sound track in main menu	VALID: Accessing the settings in the main menu and toggling the SFX to 'off'	VALID: The sound effect only associated at the main menu – this being the interaction sound – should not play.	VALID: As expected.	YES	3.4
		INVALID: Not toggling SFX to off in settings	INVALID: The sound effect continues to play at main menu	INVALID: As expected	YES	3.5
17	Turning music off in main menu	VALID: Toggling the music option to 'off' in the settings at main menu	VALID: Music should cease to play when the user starts the game.	VALID: As expected	YES	3.6
		INVALID: Not toggling the music option off	INVALID: The music should continue to play	INVALID: As expected	YES	NA

Video Reference hyperlinks for Sound testing: Covers 2.8 -> 3.0

Sound Testing 1.mp4

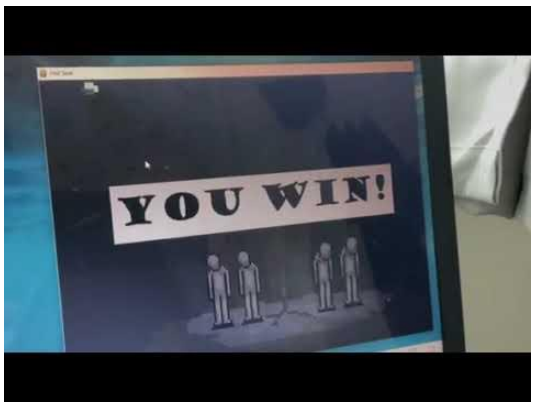


Video Reference hyperlinks for Sound testing: Covers 3.1 -> 3.15

Sound Testing 2.mp4

Video Reference hyperlinks for Sound testing: Covers 3.2 -> 3.3

Sound Testing.mp4



Video Reference hyperlinks for Sound testing: Covers 3.4 -> 3.6

sound testing 3.mp4

SCREEN ELEMENTS / ASSETS

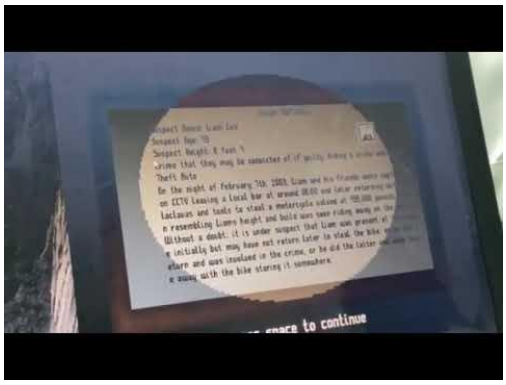
This section of testing will aim to test the presentation of visual screen elements / assets for my game.

Test Number	What is being tested?	Input/Action	Expected Outcome	Actual outcome	Has the requirement been reached?	Video evidence reference
18	Hot Bar presentation	VALID: Starting a level via pressing 'space' on the case detail screen	VALID: Hot Bar should appear on the left side of the screen with some fill.	VALID: Hot Bar does appear on the left side of the screen with fixed fill at the start of level.	YES	3.7
		INVALID: User is not playing level.	INVALID: No hot bar to display	INVALID: As expected, no hot bar displays until level is playing.	YES	NA
19	Question presentation	VALID: Starting a level via pressing 'space' on case detail screen	VALID: Questions should appear in randomised order at the bottom of the screen.	VALID: As expected, the questions appear in randomised order at the bottom of the screen. Randomised essentially	YES	3.8

				means that if the game is played on two different occasions, the order of the questions in ranking '1, 2, and 3' should be different.		
		INVALID: User does not start playing level	INVALID: In that case, no questions expected to appear on screen	INVALID: As expected, no questions appear on screen until level starts.	YES	NA
20	Automatic close feature at ending	VALID: No user input at the ending screen	VALID: The game should automatically close after a period of roughly ten seconds without any user input.	VALID: When the ending screen is displayed, as expected, the game automatically shuts down after about ten seconds.	YES	3.9
		BORDERLINE: There is a user input, however the user clicks on the exit button for the game window.	BORDERLINE: Although the game would automatically shutdown with no user input after a short period of time, the user could technically also interact with the game window exit feature which is presented in any application, which should close the game	BORDERLINE: As an alternative, the game does indeed also close down in a viable way through this option as well.	YES	4.0
		INVALID: clicking 'esc' button at ending screen	INVALID: This input would not cause the game to close	INVALID: The game does not close as expected.	YES	4.1

Video Reference hyperlinks for screen elements / assets testing: Covers 3.7

Screen Element testing 1.mp4

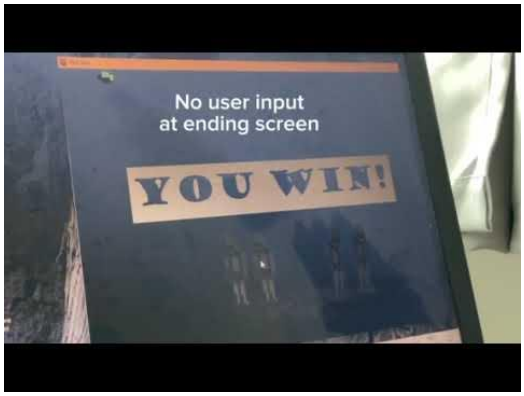


Video Reference hyperlinks for screen elements / assets testing: Covers 3.8

Screen Element testing 2.mp4

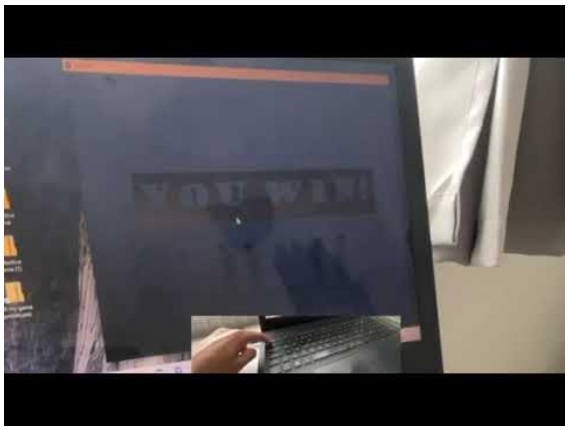
Video Reference hyperlinks for screen elements / assets testing: Covers 3.9

Screen Element Testing 3.mp4



Video Reference hyperlinks for screen elements / assets testing: Covers 4.0 -> 4.1

Screen Element testing 4.mp4



USABILITY FEATURES

Here, I will go over, along with my 'end-user,' Fardis Rajabi, the usability features. The constant question I kept in mind here was: "are the features useful to my end user when they are playing my game?"

MENU

Me: For my main menu, I have made it such that you can interact with the limited options [play, settings and exit] via the Left mouse click on your mouse. What are your thoughts on this?

F.R: I appreciate the simplicity behind the main menu. It is straightforward and accessible to all.

Me: I understand that you may have also acknowledged that my menu screen doesn't pose a lot of options for the user to select. What are your thoughts on this?

F.R: Honestly, I don't see what else you may've included for the main menu! It's a good starting point to the game and I like the settings option too.

GAME CONTROLS

Me: Now, I did highlight in early planning for the game that I will only incorporate a 'point and cl throughout the game, however I did decide to switch it up slightly when it comes to playing the levels. Here you are required to use the NUM keys. How useful is this for you?

F.R: Simple and easy to use. However, I will say that when playing the level initially it wasn't straight away clear as to how I was meant to select my desired question.

Me: Yeah exactly, that's a good point, I didn't incorporate a feature such that it informs the player in the game how to select the question, so I decided to combat this by introducing a 'READ ME' note before they play the game.

F.R: Ah okay I see, in that case it shouldn't be a problem if you were to distribute the game to other users. Although we cannot guarantee every end-user will actually read it before playing however this is upon them.

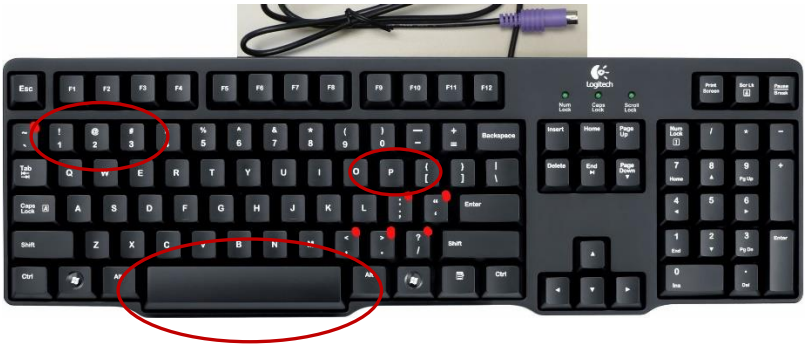
Me: Exactly, it's kind of like players choosing between doing a tutorial at the beginning of a game or skipping it. but I think I've done my part here. Anyways, moving on, along with the NUM keys for selecting questions, and the LMC on the mouse for selecting buttons, I also use the space bar which needs to be used to begin a level after reading the case details. What did you think of this?

F.R: I appreciated how a message does appear at the bottom of the case popup screen which tells you to click the space button to move on. No problems on my end.

Me: Great, and also the 'p' button can be used to pause the game whilst you are playing the level. What were your thoughts on this?

F.R: I feel like this feature should be removed, particularly as the pause screen is unfinished and not properly developed, and also, just like the NUM keys for questions selection, there is nothing indicating in game how to pause the game if need be. I also found that by pausing the level resets, so it is a bit of a buggy feature.

Me: You have a great point, it was unfortunately I feature I didn't finish and it does cause some troubles.



SCORING SYSTEM

Me: The scoring system as we discussed several times would be the hot bar system I implemented in my game. I know we both mutually mentioned how the only design to change was that of the hot bar. What did you think of this?

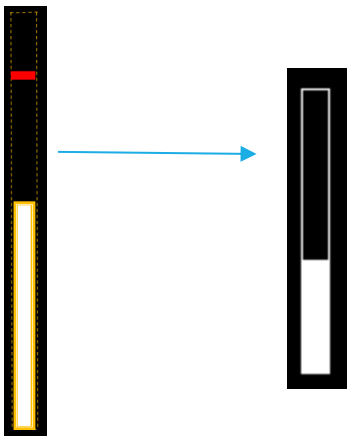
F.R: The design is simple and easy on the eye. I think the hot bar is also the perfect size relative to the screen size. Some cool features you could've added alongside the hot bar might've been an animation as it fills up or depletes, but I know that development had to occur quickly.

Me: Right, these are definitely ideas I could carry forward for later versions of the game. I did initially say that the hot bar would have a marked point, where if reached, the level would be won. However, in the game, I made it such that the hot bar has to fill completely to win the level. Any problems with that?

F.R: I didn't even notice it to be honest. They are both basically the same thing and seeing the hot bar fill completely is much more satisfying as a player in my opinion.

Me: Perfect, lastly what do you think about the progression of the hot bar in terms of winning and losing a level?

F.R: To be honest, it wasn't too difficult or too easy to win a level. I really think you have nailed a perfect balance. I did have to think before selecting a question but not to a point where it was tedious. It was really fun actually, and the hot bar didn't fill too quick, but how quick it dropped from a poorly selected question was a good warning to stay focussed.



Hot bar evolution from proposed design to final design.

HOW WELL DOES MY SOLUTION MATCH THE SUCCESS CRITERIA PROPOSED?

I will now go over my success criteria table which I made in the research stage. I will compare the success criteria to the solution proposed and see how well the solution met the criteria.

Changes made throughout the development stage in reference to the original criteria will be highlighted, and any unmet or partially completed requirements will be highlighted, and I will explain how I would've completed them fully.

1. Level Criteria

Requirement	**Justification**
Minimum of four interrogation levels to be created, featuring different 'suspects' and their respective cases.	Gives the player more opportunities to achieve a win ending as they have a greater number of tries. Also makes the game more interesting without making it too time consuming by having double digit levels for example.

This requirement was partially met. Instead of creating four interrogation levels, I ended up creating three. I did ensure to present different fictional suspects along with their cases for each level to keep the player entertained, so I did meet this part of the requirement. There would have been no problem in creating an extra playable level, however I was aware of time constraints.

In order to fully meet the requirement, I would have developed an extra level, writing out the required transcript for it etc.

2. Player POV (point of view) criteria

Requirement	Justification
3rd person sky view	Seems most applicable to use as the game is two dimensional.

The way I met this requirement also changed, as I decided to implement the same design used for my menu screen over to the levels, which does ensure a two-dimensional game, however the perspective of the player is from a side angle, rather than from above.

I didn't see the need in ensuring a 'sky view,' as it does not affect game quality or playability so much, hence was content with my proposed design.

3. Main Menu Design criteria

Requirement	Justification
A starting menu with a minimum of three options	The three options that will be featured on a minimalistic scale will be the 'play' button to let the player start the game, the 'exit' button to quit the application and a settings button to access the settings menu so the player can modify the game to their likes.
Simple settings menu	Gives the player the freedom to toggle on and off the sound effects in the game and the music in the game.

This requirement was completely met! My final solution does feature a starting menu with the mentioned selectable options. The requirement does mention 'minimum,' however, there was not anything else that would've been logical to add to the main menu.

On deeper thought however, having a controls option to allow the user to see the different controls they can use through the game would've been better than informing them via the readme document in the game file.

The settings menu was developed to allow the player to alter two things, this being to turn the game music ON or OFF and turning the sound effects associated with the main menu ON and OFF.

4. Sound effect/tracks criteria

Requirement	Justification
Sound effects	Simple sound effects will be downloaded into files from external sources and added into the game, which will be used at points throughout the game to immerse the player slightly more. Will also be used when the player is clicking with their mouse.
Background music	A minimum of two different separate tracks will play in the background. The initial track will play on the menu screen when the game is first loaded up. The following track will play for the duration the game is played. May also use another two tracks for a win ending and a lose ending.

These requirements were also successfully met. I utilised YouTube as a great resource to obtain my sound effects and soundtracks for the game.

These were all the sound effects / tracks I used summarised below:

- button interaction sound effect at main menu

- main menu backing track sound

- level backing track sound

- win screen backing track sound

- lose screen backing track sound

5. Game Ambience criteria

Requirement	Justification
A dark theme implemented for the game.	To allude to the actual role of dealing with potential criminals and the nature of crime itself, a darker, more sinister theme, may be the most fitting for the player playing the role of the

	interrogator, in contrary to a bright and cheery atmosphere.

I would confidently say that I also met this requirement. To ensure the overall theme/ambience for my game was darker and more sinister, I made great use of the soundtracks, and also ensured that colours used throughout the game were darker and linked to the themes behind the game. This darker theme can also be seen in some proposed designs for the game, such as the ending screen designs, especially for the LOSE screen design.

6. Scenes design criteria

Requirement	**Justification**
The interrogation room scene + winning scene + losing scene creation	These individual scenes must be created digitally through art in an external application and then transferred over to the game. This is an essential layer to be added to any game, as without a setting, the game would be very dull.

I used the publisher application to create all my proposed designs, including that used for the winning ending screen, losing ending screen, and what was eventually used for the interrogation room scene during the levels.

Being transparent, I intended to create a unique setting for the 'interrogation room,' however as described earlier I decided to copy the design used in my main menu screen and might slight altercations. Had I more time, I would've created a new environment for the interrogation room, which I believe would be a small but significant addition for the player that would increase the game quality.

7. Hot Bar Criteria

Requirement	**Justification**
Hot bar mechanism	A visual hot bar placed on the side of the screen for the player to see in the interrogations which fills and depletes and remains neutral, in respect to the three types of

	questions that can be prompted by the player.

The hot bar mechanism requirement was successful in the end:

The dynamic nature behind it was implemented, in terms of the way it can either remain neutral, fill up or decrease depending on the question the user selects in a multiple-choice decision. The hot bar was placed appropriately on the left side of the screen for the player to easily see and keep track of throughout the level. If there was something I would change, it would be more so to do with the design of the hot bar, making it slightly more complex and appealing to the eye and including some animations for its respective movements.

8. Level Detail criteria – case script writing

Requirement	**Justification**
Fictional cases decided for fictional suspects in the game.	Must create fictional cases for suspects portrayed in the game. This will be displayed to the player and not be too long or difficult of a read. Allows the player to gain an understanding of the suspect and hence the best track of questions to ask to win the interrogation.

Another successfully implemented requirement. I dedicated time at the beginning of my development story to write out a script for each level, which included the fictional case detail for each fictional suspect in each of the three levels. I took into account the target audience, 16 and above, hence ensured that the cases were based on low level to medium crimes, trying my best to stay away from more "hard core" criminal activities which would render my initial target audience aim useless.

I would argue that a more logical player with a more logical approach can easily overcome and win each level in the game without having to refer so much back to the case details presented at the start of the level.

I observed several test users playing my game and found that they showed difficulty on average from the first level into the early parts of the second level, however gauged an understanding by then. The third level in addition, I believe, was poorly scripted in terms of the questions to follow, making it too easy. This however balances out the game, as each user on average lost the first level and was able to win the next two in order to have a better chance of winning.

9. Multi-question prompt script criteria

Requirement	**Justification**

Question prompts	Deciding on what questions to provide the player for them to be able to choose from in the interrogations. This is an essential aspect of the game, as the game revolves around this multi-choice system.

Within the creation of my game script highlighted in the documentation, I also set out the script for the 3 questions which can be asked at 5 different points in all three levels. Like mentioned in the prior explanation of requirement 8, I believe I could've made the questions slightly more challenging for the last level, however overall, I am satisfied.

I stored the large script for my question prompts and case details in the game class, which I show screenshots of in the development stage.

10. Level Outcome criteria [win,lose]

Requirement	**Justification**
Simple calculation system to calculate the one of the two endings the player obtains.	If only 4 interrogation scenarios are developed for the game, the player must have at least a 75 percent success rate, meaning they incriminate at a minimum of ¾ suspects.

This requirement was successfully implemented in a slightly different way, and very important as it determined if the player had done enough to win a level, or lose on the contrary, and overall determined a win or lose ending.

As my final solution only made use of three interrogation levels, these were the basics of the calculation system I established which I found most suitable:

- 3/5 questions answered excellently leads to a level win

- 2/5 questions answered poorly leads to a level loss

- 2/3 levels won should lead to a level win as a minimum value

- 2/3 levels lost should lead to a level loss as a minimum value

11. Pause System criteria

Requirement	**Justification**

Pause menu displayed through clicking 'm' character on the keyboard, and way of resuming game from this point.	If the 'm' button is clicked, a pause menu is displayed whilst the game is being played. This displays the elements of the settings menu + an exit button to leave the game. This is useful as the player can halt the game as needed and return to it. The player can then resume the game by clicking the enter button.

This requirement was not met completely and was developed faulty. My final solution provides a method to allow the user to pause the game whilst they are playing a level, by clicking the specified button. No pause menu is rendered however, apart from a complete black screen, hence the elements of the settings menu I planned to portray and an exit button for them to quit the game at that point are not present. There is a way for the player to return back to the game, this being however by clicking the same button they used to pause the game, and hence not the enter button. In addition, the level restarts, all progress wiped away.

Overall, if I had to complete this requirement, I would develop the pause feature to ensure a menu renders of resemblance to that of the settings screen in the main menu, and also a way for the user to exit the menu, informing them how to do so, and to ensure it does not corrupt level progress.

12. Game Character sprite criteria

Requirement	**Justification**
Creating a fixed character sprite for the interrogator (the role which the player plays) and different character sprites for the suspects in the different levels.	Allows the characters in the game to be visually displayed to the player in a simplified manner.

This requirement was successfully met. I exceeded the requirement by also introducing a 'mugshot' feature for each suspect which is displayed on the case detail screen before the start of each level. A cool, but simple feature.

Although I use the menu screen design for the level design, I ensured to alternate the 'suspect' sprite with three different sprites which resembled the suspect description as best as possible for good attention to detail, which my test users appreciated.

MAINTENANCE | LIMITATIONS

Maintenance for a game is essential to update the game, fix any bugs or to introduce new features to improve the game over time. In my case, this would be taken up by introducing any requirements I had stated prior to development which were not implemented as intended or at all for that matter.

In addition, as I can only test the game so much, if the game was to be distributed to a plethora of players, they may across bugs which I had not found, hence introducing a report feature which re-directs them to a basic form to explain the bug would be smart.

MAINTENANCE AND LIMITATIONS

Pause menu displayed through clicking 'm' character on the keyboard, and way of resuming game from this point.

The pause menu doesn't affect gameplay so much. None of my test players actually had to pause the game as it has such a short duration, of roughly 5-minute play time max. However, if I was to update the game in the future and add more levels, this would increase average game time, hence implementing this feature would improve the game.

Going over my source code, I realised that I had provided two pathways to allow the user to pause the game, this being through both the 'm' and the 'p' keyboard button. This would be an easy fix, as I can remove the latter to match the game requirements.

In addition, the actual module pertaining to the pause feature isn't finished to give me my desired outcome. Only a fadeout is provided to a black screen, and the sfx linked to the pause menu plays. In order to undergo maintenance, I would have to add some assets found in the main menu such as settings and a quit button. Furthermore, the bug which restarts the entire level also remains when exiting the main menu.

```
def pause(self):
    if self.music:
        pygame.mixer.fadeout(1000)
        self.SFX['pause'].play()
    while True:
        self.screen.fill((0,0,0))  # Fill screen with black color

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_m or event.key == pygame.K_p:
                    pygame.mixer.fadeout(100)
                    self.SFX['game'].play(-1)
                    self.run()
        self.clock.tick(60)  # Limit frame rate to 60 FPS
        pygame.display.update()  # Update display
```

Minimum of four interrogation levels to be created.

Going forward with the game in the future, this requirement would be completed by the creation of additional script of new suspects, case details and questions, and then adding elements of the script into the appropriate location without the main.py source code and making tweaks to facilitate for an extra level.

Hot bar mechanism

Although I did complete the requirement, I believe that it could be made to be more visually appealing by importing designs rather than using pygame's default drawing features. In addition, animations could be applied to the hot bar, such as when it fills up or depletes, by looping through an animation sheet, something I am current unfamiliar with and would have to learn.

The interrogation room scene

Due to time constraints, I did decide to copy over the main menu design for the interrogation room scene. However, to truly implement this requirement as it was intended before, this to create separate designs for the interrogation room scene during level gameplay, I would need to create new designs for the interrogation room, possibly in the publisher software as well or in another design software.

3rd person sky view

To complete this requirement, I could approach it in two ways. Number one will require perspective transformation to a 2D top-down representation of the scene. With some brief research, if I was to initially design the image for the interrogation room scene in a manner which was not from a 'bird eye view' this would appear to be quite complex, as I would have to use the OpenCV library and undergo some steps such as:

- identify co-ordinate points of all vertices of the image

- specifying the width and height of the transformed image

- apply a perspective transform method

…to obtain a bird's eye view of the design.

However, a much simpler approach would be to design the interrogation room scene in such a way that it is already from a 'sky view / bird's eye view' perspective.

PROJECT APPENDIXES

CODE LISTING