

SOFTWARE DESIGN AND DEVELOPMENT (SDD)

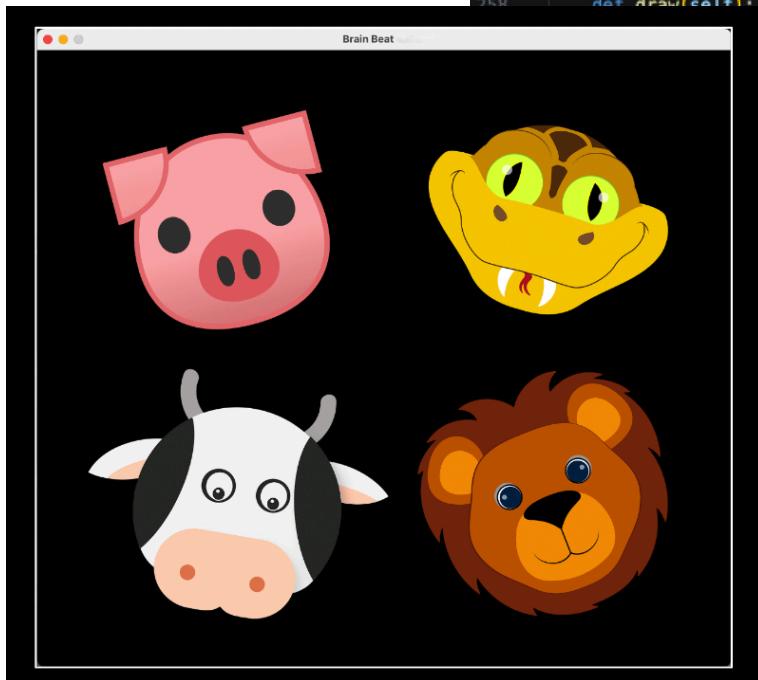
HSC COURSE - TASK 3 - *Brain Beat*

DR. ZANG

KELLYVILLE HIGH SCHOOL

ARAV ZAD 2024

```
241
242     def game_over_animation(self):
243         original_surface = self.screen.copy()
244         flash_surface = pygame.Surface((self.screen.get_size()))
245         flash_surface = flash_surface.convert_alpha()
246         for beep in self.beeps:
247             beep.play()
248         r, g, b = WHITE
249         for _ in range(3):
250             for start, end, step in ((0, 255, 1), (255, 0, -1)):
251                 for alpha in range(start, end, ANIMATION_SPEED * step):
252                     self.screen.blit(original_surface, (0, 0))
253                     flash_surface.fill((r, g, b, alpha))
254                     self.screen.blit(flash_surface, (0, 0))
255                     pygame.display.update()
256                     self.clock.tick(FPS)
257
258     def draw(self):
259         self.screen.fill("black")
260         0, 20, f"Score: {str(self.score)}").draw(self.screen)
261         0, 20, f"High score: {str(self.high_score)}").draw(self.screen)
262         n self.buttons:
```



PREFACE.....	3
DEFINING AND UNDERSTANDING THE PROBLEM.....	4
PLANNING AND DESIGNING THE SOFTWARE SOLUTIONS.....	12
IMPLEMENTATION OF SOFTWARE SOLUTIONS.....	20
TESTING AND EVALUATING SOFTWARE SOLUTIONS.....	43
MAINTAINING SOFTWARE SOLUTIONS.....	44
REFERENCES.....	45

PREFACE

This document is created to meet the previous software proposal requirements as portrayed in **Task 1 - Programming Project Report**. The software product that will be developed is 'Brain Beat' - which aims to provide entertainment to users of all ages. However in disguise it's real intentions are to promote cognitive thinking with classic memory games suited to resolve today's day and age related problems. **Please read the following document leading to this URL** before acknowledging this Software Report. Minor changes have been made from V.01 to V.02.

URLs:

 V.01

<https://docs.google.com/document/d/1jZt2f-IEGI3JdBwRfvqsPnh6o2pAWAyeQGmuEz2xcE/edit?usp=sharing>

DEFINING AND UNDERSTANDING THE PROBLEM

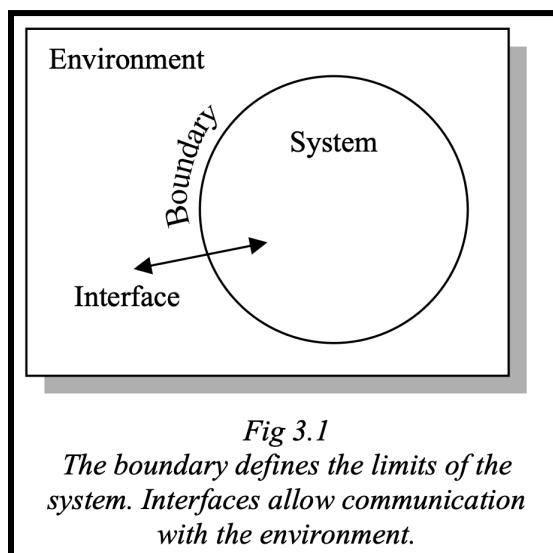
Defining the problem

A problem that has been arising a lot nowadays in society are mental health issues such as dementia and Alzheimer's disease. This happens due to excessive stress levels, and other problems such as unhealthy addictions. This is relevant to all ages.

Therefore, a solution to this problem has been identified through extensive research such as: Healthy Memory Games. Whilst healthy memory games can include playing cards and board games, a simpler way to distribute the solution would be through making a software application which can be used by everyone, anywhere, and at any time.

By making the application caterable for all age groups, it must be visually appealing, and exciting; the complete opposite of mundane. Due to Software Developing issues of compatibility (Windows and MacOS) this program will only be available for MacOS users. This is still a large attribute as 20% of the world uses MacOS. In the future, depending on the success of the application, given time and money, a more compatible version for Windows users may also be established.

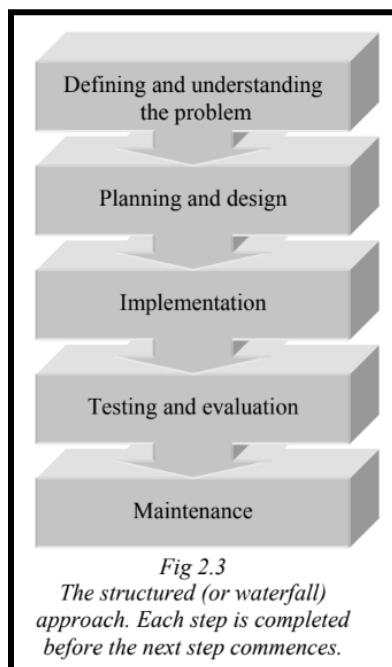
The environment includes all external factors that can interact with the game but are not directly controlled by it. The environment consists of the player, who interacts with the game through the user interface. It includes the device on which the game will be played on, in this case, a computer and the monitor or laptop. In detail, the Operating System of the Computer, and any potential software that could interact with the game, for example screen recording or live streaming. The interface between the game (system) and its environment would be the points of interaction between the game and the player (input and output), and between the game and the device's operating system. For example, the player inputs their response to the game sequence through mouse clicks and the game outputs the sequence and feedback through the screen and speakers. The game also interacts with the operating system for tasks like saving progress or accessing system resources.



In terms of Legality of the Software Application, Brain Beat is inspired from the Simon Memory Game, it is obvious that some aspects of the GUI and the Game may seem identical. However that is not the case. Social and Ethical Considerations are taken into consideration whilst crafting Brain Beat and here are the following alterations. The changing nature of work for the users is that instead of a normal RGBY game with audio, Brain Beat has an Education Mode, where an Animal Mode is present. This includes Snakes and Lions, etc, making the noise of a “Sssss” or “Roar” once clicked to make it more visually and cognitively appealing. The effects on the Public are the same as Simon. And that is to create a game which depletes all levels of stress and increases overall brain performance. Due to this, the Software is free, and ‘Brain Beat’ falls under the category of **Shareware**. The reason being that the game is not suitable for public modifications and reproduction of the product, where derivatives are made. Therefore, the Software is covered by Copyright, as in the market, there are not many games that explore this type of game mode and functioning. However, copies can be made for archival and distribution purposes. But, as mentioned earlier, modifications and reverse engineering is not permitted as it may defeat the purpose of the game, to be for the goodness of humanity. However, if there are inquiries, the public is allowed to ask for permission, permission may be granted after careful consideration.

Cost is not exactly an issue, as the development is being accomplished by one person on a Macbook Air 13”. The programming language is also free, as Python is being used along with libraries such as ‘pygame’. As this is part of my HSC, no outsourcing will be executed and there is no staff that I am paying for. However as there is only 1 person working on it, me, that too with additional commitments (other subjects), this results in development taking time; lots of it, but at the same time, I have 8-9 weeks to complete.

This leads to the decision as to which approach will be more suitable for this. Given the large time period I have to accomplish this and the sheer magnitude of the problem that I want to resolve, the most suitable Software Development Approach would be the **Structured Approach** also known as the Waterfall Method.



Design Specification

Needs -

A need is an instance in which some necessity or want exists. Needs of the Software Product for the Target Audience and the Developer include:

Developer's Aspect	User's Aspect
<ul style="list-style-type: none">- Integers for Scoring System- Arrays to store the order of pattern- Intrinsic Documentation such as variables to ensure a smooth development environment- Documentation for the EndUser so that the application is able to be executed without fault.- Relevant Libraries such as 'pygame' to ensure optimal GUI- Must stay consistent to using MacOS libraries	<ul style="list-style-type: none">- A visually appealing game with consistent use of colours, whitespace (black) and fonts and audio.- Appropriate messages such as "Are you sure you want to quit?"- Appropriate Icons such as hover over buttons to make sure that it needs to be clicked on.- Must be able to run well on MacOS and Windows

Objectives -

Objectives are the short & long term aims & plans for the software being developed.

The objective for Brain Beat consist of the following:

- To provide an entertaining and engaging gaming experience.
- To challenge and improve the player's short-term memory skills.
- To ensure the game runs smoothly and reliably on a computer regardless of Operating System.
- To create a game that is easy to understand but difficult to master, thereby encouraging repeated play, formulating a healthy addiction

Boundaries -

Boundaries define the limits of the problem or system to be developed. Anything outside the system is said to be a part of the environment. The system interacts with its environment via an interface. In this situation, the Boundaries of the 'Brain Beat' Program include the following:

- The Sequence Generation Logic (Cannot be altered by the Player)
- The Scoring System
- The Progression System

Communication

To improve the accuracy of the solution, as a developer, they will be given opportunities to add their input as they will be the ones using the product. To accomplish this, a survey will be conducted on public account, and, for people who I know, an interview will be conducted with more thorough questions asked after the stage of development. Such as - "Are there any features from the GUI that you dislike? If so, what is it about this feature that throws you off? Please describe in simple terms."

I will be using Software such as 'Google Forms' to easily distribute questionnaires across my Social Media and through pyramiding (The distribution of copies from one person to another). So, with an overview of questions, I can see what answers were written in terms of each respective aspect, such as I may ask simple questions such as - "When navigating through the game, what aspect would you like to be made easier?", "How are you feeling about Brain Beat?", and "What score out of 10 would you give the product?", and "What is the most difficult part of using our app?". These questions will be in the Google Form.

This results in a sense of ownership which users will feel; that they have contributed to such a large project. This builds confidence, and confidence will result in a successful implementation scheme. <https://forms.gle/JYWELj1fddJrUkg26> - Link to google form

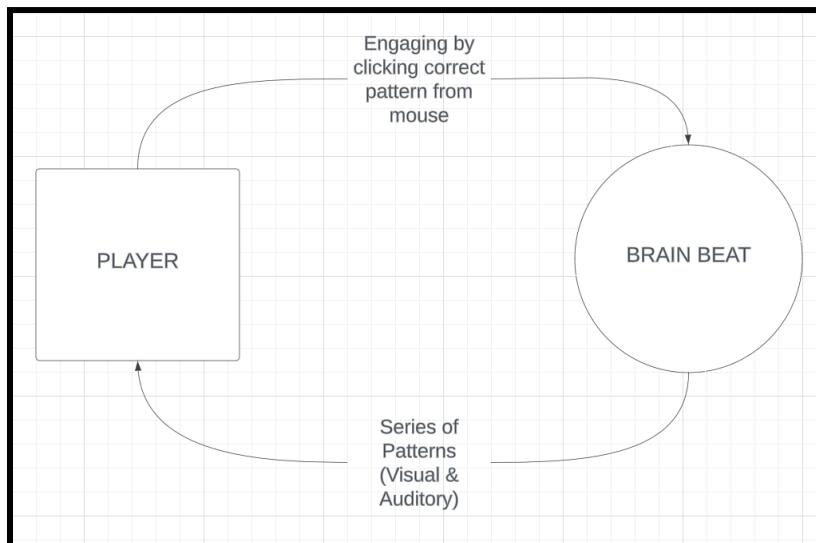
System Modelling

As a Software Developer, I am viable to ensure the best possible outcome for the End Users. To do this, I cannot jump into the stage of implementation (Actual coding in Python). I need to first model modules and model logic. A simple yet powerful example of this is that I cannot start coding the GUI aspect without first modelling, sketching, colouring a screen design. If I do start implementing right away, it will result in messy, inconsistent screens with incompatible functionalities. Therefore, it is important to get it right from the beginning.

Screen designs aren't the only use of system modelling. In relation to 'Brain Beat', each piece of logic is important.

For example, the use of a Context Diagram, Storyboards, and Concept Prototypes are vital and appreciated below.

Context Diagram:



Pseudocode:

There have been some additional changes made to the Pseudocode from V.01. Only the relevant changes will be listed - Below is a rough pseudocode logical response that makes sense according to me, the developer of the solution. Although some of it may not follow Australian Pseudocode Standards, it is important that it is easy to understand as the developer.

Initialise the Game:

Set up the game screen, load resources (images, sounds), and initialise variables.

Set the initial score to zero.

Display the main menu.

Main Menu:

While the game is running:

 Display menu options: "Play", "Options", "Quit".

 If the player selects "Play":

 Go to the game mode selection.

 If the player selects "Options":

 Go to the options menu.

 If the player selects "Quit":

 Exit the game.

Game Mode Selection:

While the player is in the game mode selection screen:

 Display game mode options: "Education", "Normal", "Challenge", "Back".

 If the player selects "Education":

 Start the education mode.

 If the player selects "Normal":

 Start the normal mode.

 If the player selects "Challenge":

 Start the challenge mode.

 If the player selects "Back":

 Return to the main menu.

Game Modes:

For each game mode (Education, Normal, Challenge):

While the player is in the selected game mode:

Generate a sequence of colours/sounds to display (increase the sequence length as the game progresses).

Display the sequence to the player, one colour/sound at a time.

Wait for the player to repeat the sequence by clicking the correct buttons.

If the player correctly repeats the sequence:

 Increase the score.

 Generate a longer sequence and repeat.

If the player makes a mistake:

 End the game mode and display the score.

 Automatically restart game logic

Options Menu:

While the player is in the options menu:

Display options: "Volume Slider", "Clear High Scores", "How to Play", "Back".

If the player adjusts the volume slider:

 Update the game volume.

If the player selects "Clear High Scores":

 Confirm the action and clear the high scores if confirmed.

If the player selects "How to Play":

 Display instructions on how to play the game. - Web page redirection

If the player selects "Back":

 Return to the main menu.

End of Game:

When the game ends (either by player choice or error):

 Display the final score.

 Play Game over Animation

Exit Game:

Close the game window and terminate all processes.

`sys.exit()`

Summary of Defining and Understanding

In Summary, a conclusion has been made on what the solution is. A game is going to be created to cater for all age groups to battle against the rising tensions of dementia and unhealthy addictions; or just for pure entertainment. The name of the Game is 'Brain Beat', inspired by the 1970s retro indie-pop game 'Simon Memory Game'.

As a Software Developer, I have acknowledged the criteria that the game can realistically abide by. The game as an executable file (.exe) will only be available to MacOs users. The game will consist of 3 modes. Education Mode, Normal Mode, and Challenge Mode.

PLANNING AND DESIGNING THE SOFTWARE SOLUTIONS

Custom designed logic used

Multi Dimensional Arrays used:

An example of a multidimensional array used is:

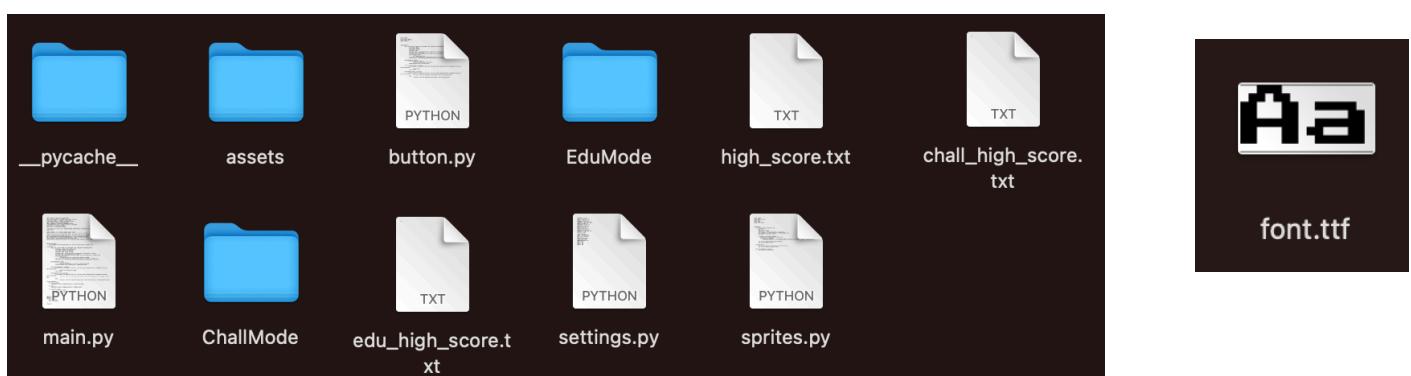
```
27  class Button():
28      def __init__(self, image, pos, text_input, font, base_color, hovering_color):
29          self.image = image # Set button image
30          self.x_pos = pos[0] # X position
31          self.y_pos = pos[1] # Y position
32          self.font = font # Font for text
33          self.base_color, self.hovering_color = base_color, hovering_color # Colors
34          self.text_input = text_input # Text on button
35          self.text = self.font.render(self.text_input, True, self.base_color) # Render text
36          if self.image is None:
37              self.image = self.text # Use text as image if no image
38          self.rect = self.image.get_rect(center=(self.x_pos, self.y_pos)) # Button rect
39          self.text_rect = self.text.get_rect(center=(self.x_pos, self.y_pos)) # Text rect
40
```

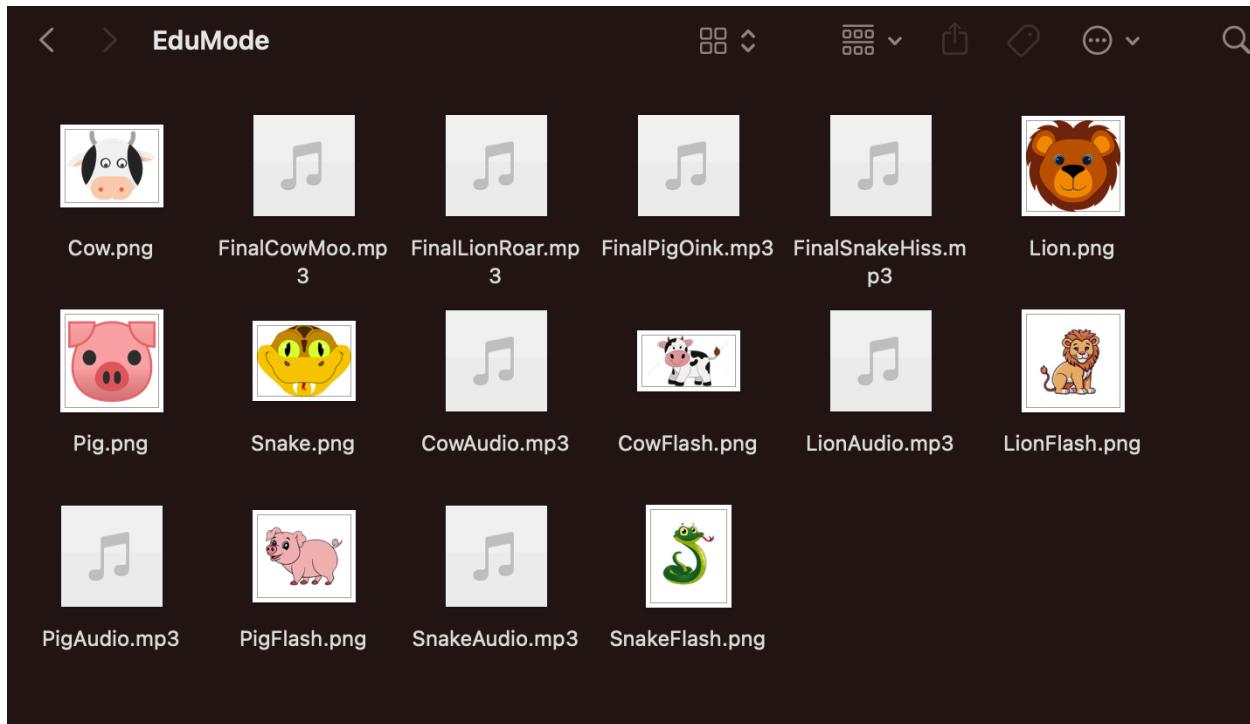
Under this Button Module, we can grasp many items and take what is needed in order to shape, size, aesthetically please, and place the button on the screen. Calling this module would result in an application like this:

```
PLAY_BUTTON = Button(image=None, pos=(215, 450),
                     text_input="PLAY", font=get_font(40),
                     base_color="White", hovering_color="#d6b4fc") # Create the Play button
```

Files:

A file, in terms of SDD, means a collection of data stored in a logical and concise manner. Other than python files, I have also used various other types of files as seen below:





Here is a table as to all the types of files that I have used:

File	Description
Python	Python file - where I write according to syntax logically
MP3	Sound file - used so that the user can hear on command
.txt	Text File - used so that I can store temporary data such as High Score and Text
.png / .jpeg	Image File - used so that user can experience visual prompts on command
.ttf	Font file - used so that text fonts and sizes are not default (Ariel, Times New Roman), visual appearance enhanced. Downloaded from internet, source below
.ico	Icon File - used so that when the game is launched the icon is the logo. Used a simple .png to .ico converter online to achieve this.

Interface Design

When designing an appropriate screen design for each window, below is the criteria:

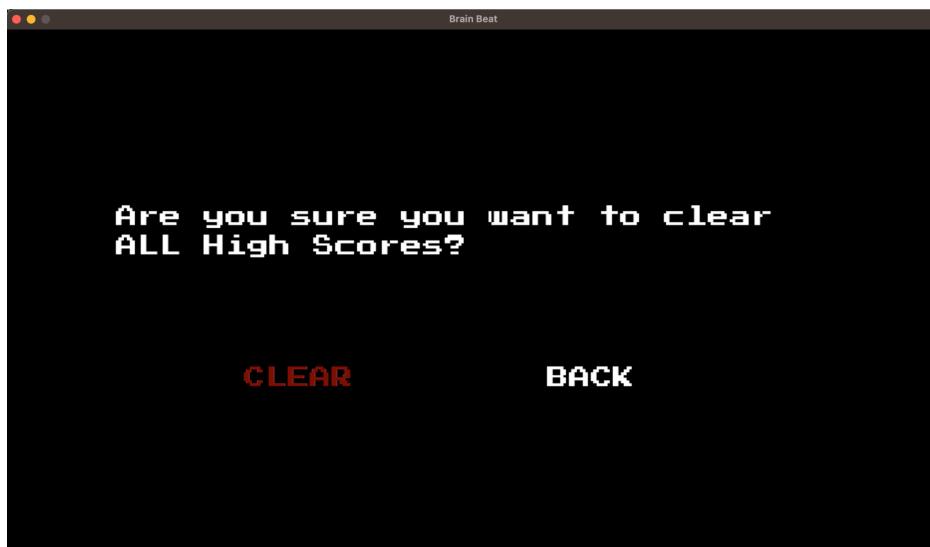
Current popular approaches

Current popular trends in Software around the world and not just Australia include the following aspects:

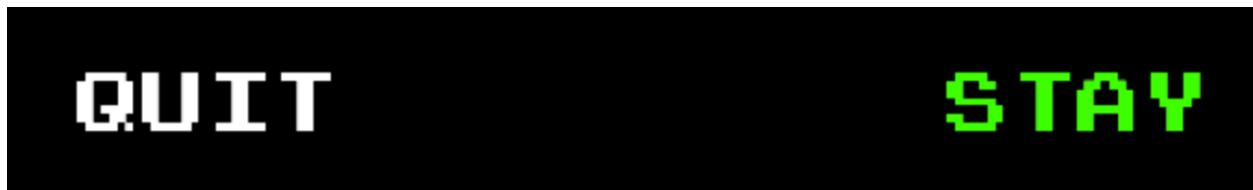
- Button colour hovering

Play Hover	
Options Hover	
Quit Hover	

- Confirmation Screens



- Command Buttons



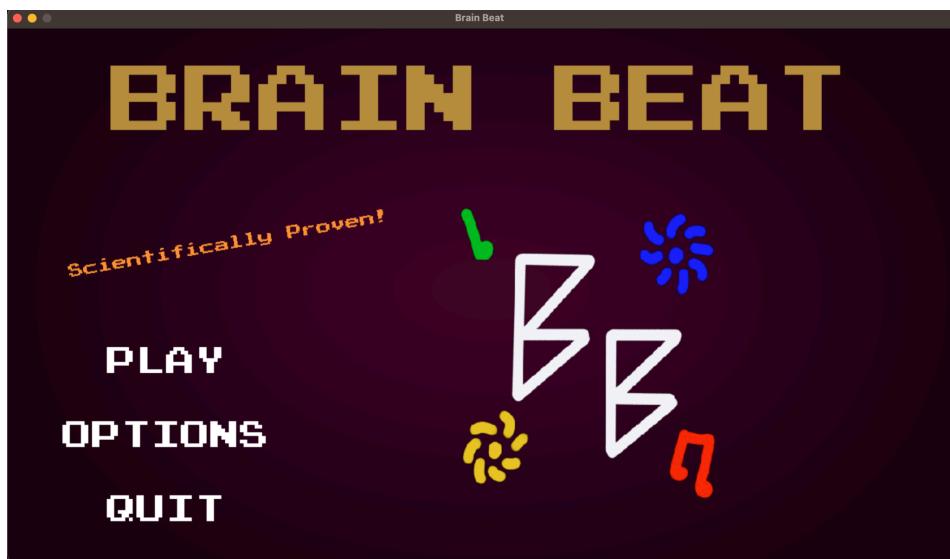
(Options given to users - Command buttons are used to select a different path for execution. Often command buttons are used to confirm a task that should take place.)

- Scroll Bars and Sliders



(Volume Scroll bar Slider - They display the position of a numeric data value within a given range comma in this case zero to 100 and the user has selected volume 75. This volume scroll bar gives users a visual interpretation of the current volume of the game.)

- Picture / Image Boxes

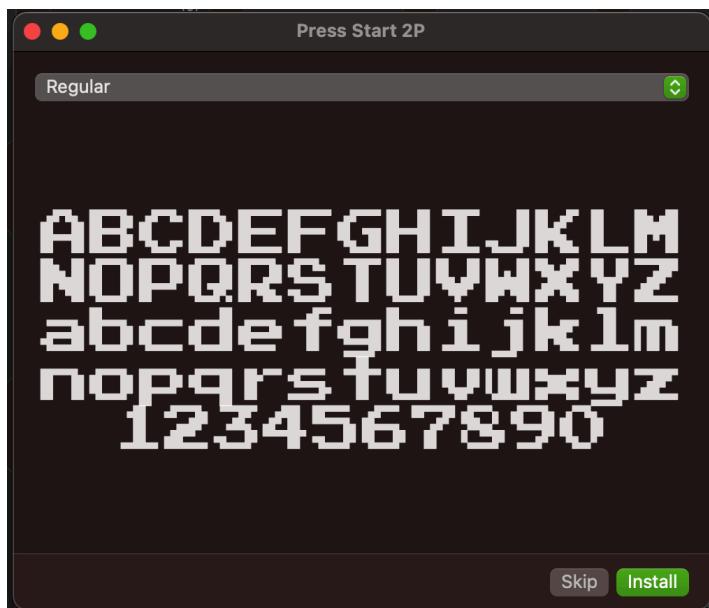


(The picture is the Brain Beat Logo - where I used an online Software to get rid of the Black Background to make it transparent - Images I used to display graphics. These are output screen elements.)
I used the following website <https://www.remove.bg/> to remove the black background.

Consistency in approach

A consistency in the Application is achieved by keeping the same colour schemes where:

Element Type	Element	Used for
Colour	White (Pygame Module)	Any button without hover effect stands out from the dark purple background.
Colour	#b68f40	Any text which is the heading, for example “SELECT GAME MODE”, and “BRAIN BEAT”
Colour	#800000	“Quit”, Exiting in confirmation buttons, all things to do with exiting and quitting
Colour	Green (Pygame module)	“Stay” - Used to continue program as it was, not terminating or redirecting to other Path
Colour	#08787f	Under Options, whichever colour #08787f is, it means it has a settings function next to it
Image		To keep backgrounds consistent between main_menu(), play(), and options(), I wanted a not too boring (black or white) and not too much of a vibrant colour (neon green or blue). To utilise WHITESPACE , I used this dark purple undertone colour.
Font	<i>Image Below</i>	Installed on internet from google fonts to get “game vibe”



By keeping the above consistent, I was able to achieve Ease of Use, Intuitive navigation, and a professional appearance so that the Game was more to today's standards.

Customisation of existing software solutions

An existing software solution is the ‘Simon Memory Game’. To change, as mentioned in the **Task 1 - Programming Project Report**, changes include:

- Adding different game modes (Challenge Mode, Education Mode)
- Within the game mode, adding different audio files instead of monotonic tones such as Snakes and Lions, etc, making the noise of a “Sssss” or “Roar” once clicked
- Adding customisable actions such as adjusting volume, resetting the high score, and viewing a HELP website

Selection of language to be used

When selecting an appropriate language to implement the solution, it is important to consider the following criteria:

What is the experience of the developers?

As a developer, I have been invested in learning and enhancing my Python skills since Year 8, 2020. Therefore, **Python** seems to be the strongest and the most logical Programming language to proceed with.

Year	Skills learnt
2020	<ul style="list-style-type: none">- Interacted with text based Python skills- Worked around “if” and basic “while” loops
2021	<ul style="list-style-type: none">- Used Python Turtle to implement these skills to a visual representation of coordinates and positioning of writing code to moving and creating visuals on screen
2022	<ul style="list-style-type: none">- Completed this online course:- https://www.youtube.com/watch?v=uQrJ0TkZlc- Throughout the year, enhancing my skills by completing one exercise a day
2023	<ul style="list-style-type: none">- Used the Tkinter Python Module to make a Coffee Shop POS System- Encountered difficulties of converting .exe from MacOs to Windows and vice versa
2024	<ul style="list-style-type: none">- Finally using Pygame to account for the overcoming of Operating System issues- Fully understand basic arrays, structures, and how Graphical User Interfaces interact with users for the best experience

Is the programming logic driven by the user or by the programmer?

The Brain Beat game's programming logic is driven solely based on how the user inputs in response to the computer's randomised order.. For example, if the theoretical array is [Blue, Yellow, Red] and the user inputs [Blue, Red], the program logic should display the game_over() function. In some cases, if the logic continues, the sequence will be wrong and mixed up.

Therefore, from online research, "Python offers both arrays and lists for handling such collections, each with its own advantages. Understanding how to create, access, modify, and manipulate arrays is essential for effective programming in Python." -

<https://www.simplilearn.com/tutorials/python-tutorial/python-arrays#:~:text=Python%20offers%20both%20arrays%20and,for%20effective%20programming%20in%20Python.>

Does the language provide for all the required features of the solution?

As researched from the below website, it states that "Python is suitable for game development, especially 2D games and prototypes. While it may not be as performance-oriented as some other languages, its simplicity, readability, and extensive libraries make it a viable choice for many game projects." - Which is perfect for the use of my Game Development.

<https://www.oyoloo.com/is-python-good-for-game-development/#:~:text=Frequently%20Asked%20Questions-,Is%20Python%20A%20Suitable%20Language%20For%20Game%20Development%3F,choice%20for%20many%20game%20projects.>

Is a graphical user interface (GUI) required?

Yes, a GUI is required as stated in the marking criteria. The game will not be text based and will implement the required design principles as stated above and in V.01.

Factors to be considered in terms of selecting appropriate technology

Performance requirements

Hardware and Software Performance issues should not be a problem as the Code is going to be developed mainly on a Macbook Air 13" with the Apple M1 chip, 8-core CPU with 4 performance cores and 4 efficiency cores, 7-core GPU, 8-core GPU and 16-core Neural Engine. It has 8GB unified memory and storage of 256GB SSD. The code will be executed and written on the famous integrated development environment (IDE) of Visual Studio Code - installed from the internet. Therefore, as the specifications are well met and there are no issues, the performance requirements are in check.

IMPLEMENTATION OF SOFTWARE SOLUTIONS

Implementation is the process of actually coding the game. This means appropriately sourcing and logically creating source code.

Techniques used in developing well written code

In this section, we introduce a number of techniques and tools available to the software developer to assist in the task of code creation. These techniques and tools aim to reduce the number of errors in the final product and improve the maintainability of the software over time.

Some Error detections included in the process of implementation are:

- | | |
|--|--|
| <ul style="list-style-type: none">- Debugging Output Statements (Error Trapping)- Flags- Stubs- Peer checking | <ul style="list-style-type: none">- Use of clear Module Structures- One logical task per subroutine- Version control and regular backup- Elegance in Design (Screen Design) |
|--|--|

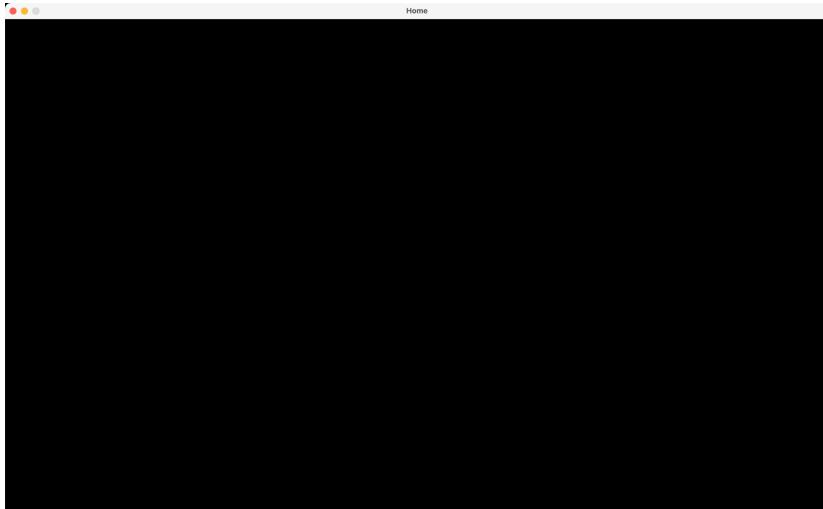
Debugged Output Statements (Error Trapping)



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Hello from the pygame community. https://www.pygame.org/contribute.html
2024-05-09 12:04:52.254 Python[23580:15317693] WARNING: Secure coding is not enabled for restorable state! Enable secure coding by implementing NSA
applicationDelegate.applicationSupportsSecureRestorableState: and returning YES.
Traceback (most recent call last):
  File "/Users/aravzad/Desktop/Task 3, SDD/main.py", line 31, in <module>
    homescreen.run()
  File "/Users/aravzad/Desktop/Task 3, SDD/main.py", line 27, in run
    self.clo.tick(FPS)
    ~~~~~
AttributeError: 'HomeScreen' object has no attribute 'clo'
aravzad@Aravs-MacBook-Air Task 3, SDD %
```

Ln 35, Col 1 Spaces: 4 UTF-8 LF Python 3.11.4



In line 27 of the code, there is a syntax error, I had missed characters, to be specific, “ck”. I then modified it by changing the code to “self.clock.tick(FPS)” and this was the output (Black screen)

This was the base of my Homescreen, where I added a various amount of GUI and Buttons to lead to the next windows.

```
File "/Users/aravzad/Desktop/Task 3, SDD/testing.py", line 17, in __init__
    self.button = Button(200, 200, RED, 100, 50, "Start")
                                         ^
TypeError: Button.__init__() takes 4 positional arguments but 7 were given
```

In line 17 of my code, I had wanted to create a button that would change colour once hovered, and noticed that the type of error was Logic Error. My thinking was that:

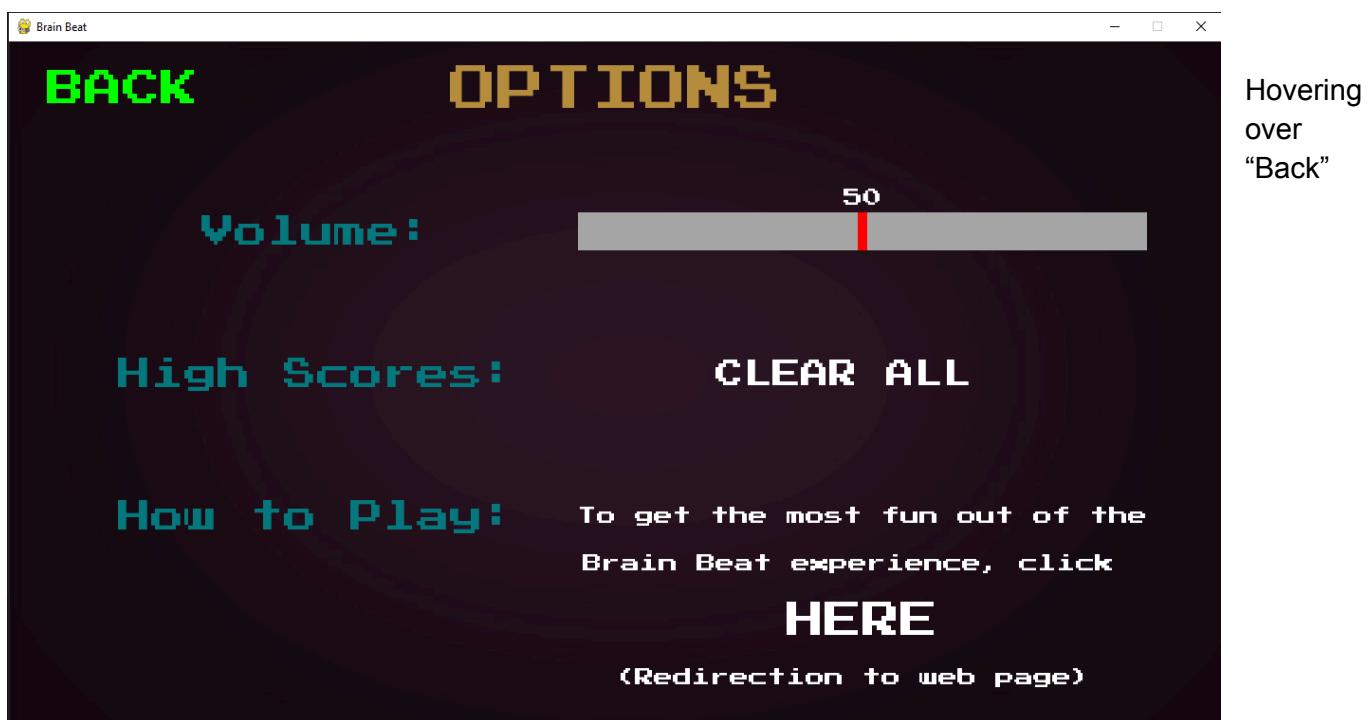
- (200, 200) specifies the position of the button (x, y).
- RED would be the colour
- 100, 50 would be the dimensions of the button
- "Start" would be the text written on there

However, since the terminal states that the `Button.__init__()` takes only 4 arguments but I had provided 7, I researched its requirements and made the following changes:

- Rechanged the Module for the Button

```
27 v class Button():
28 v     def __init__(self, image, pos, text_input, font, base_color, hovering_color):
```

- Adding a Hovering colour, I was therefore able to create a button with the hovering colour at the end.
- When calling it, I can say this with 6 arguments and therefore the output looks like this:



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

File "/Users/aravzad/Desktop/Task 3, SDD/main.py", line 169, in <module>
    main_menu()
File "/Users/aravzad/Desktop/Task 3, SDD/main.py", line 160, in main_menu
    play()
File "/Users/aravzad/Desktop/Task 3, SDD/main.py", line 68, in play
    edubutton.draw()
File "/Users/aravzad/Desktop/Task 3, SDD/button.py", line 41, in draw
    pygame.display.set_mode((1280, 720)).blit(self.image, (self.rext.x, self.rect.y))
                                         ^^^^^^
AttributeError: 'ImageButton' object has no attribute 'rext'. Did you mean: 'rect'?
○ aravzad@Aravs-MacBook-Air Task 3, SDD % □

```

Syntax Error - 15 Minute wasted time - Changing the “rext” to “rect” solved the problem

```

button_animation(button)
File "/Users/aravzad/Desktop/Task 3, SDD/main.py", line 953, in button_animation
    flash_surface = pygame.Surface((button.width, button.height)) # Adjusted to button dimensions
                                         ^^^^^^
AttributeError: 'ChallButton' object has no attribute 'width'
aravzad@Aravs-MacBook-Air Task 3, SDD % □

```

When doing Challenge Mode, after resizing the images the flashes were not in the correct dimensions. So once I tried to fix that, I got this Logic Error. By going to the line and determining its previous modular links, I was able to fix the problem.

```

173
174     def edumode():
175         while True:
176             print("In progress...")
177
178
179
180
181
182
183
184
185
186
187         |
188     main_menu()
189
190
191
192
193

```

When I clicked on Education, because of the While loop, the computer froze. This is a documentation of Runtime Error.

By removing the “While True:” I was able to use it as a basic Stub. Below is the same example.

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

In progress...

```

```
Hello from the pygame community. https://www.pygame.org/contribute.html
Traceback (most recent call last):
  File "/Users/aravzad/Desktop/Task 3, SDD/main.py", line 6, in <module>
    from button import Button
  File "/Users/aravzad/Desktop/Task 3, SDD/button.py", line 5, in <module>
    from main import get_font
  File "/Users/aravzad/Desktop/Task 3, SDD/main.py", line 7, in <module>
    from button import NormButton
ImportError: cannot import name 'NormButton' from partially initialized module 'button' (most likely due to a circular import)
```

First time encountering an Import Error. By searching up on the internet:

Python circular import is an error that occurs when two modules mutually dependent on each other try to import before fully loading. It causes a loop in which each function tries to load the other and fails, creating unpredictable behavior and runtime errors. 4 Mar 2024



 Built In
<https://builtin.com/articles/python-circular-import> ::

[Python Circular Import Error Solved - Built In](#)

I fixed so by copying and pasting some modules of code from different python files and importing them into the main.py

Flags

A flag is used to indicate that a certain condition has been met. Usually a Flag is set to either true or false, that is, most flags are Boolean variables.

```
615     # Function to run the main game loop
616     def run_game():
617         nonlocal playing, clicked_button
618         playing = True
619         while playing:
620             clock.tick(FPS)
621             clicked_button = None
622             events()
623             draw()
624             update()
```

An example where I have utilised flags is where I set playing = True (LINE 618), creating a loop where the game functions take place. With this, the clock ticks, at FPS = 60, and subprograms such as events(), draw(), update() run (LINE 619). However, if the user's pattern (STORED IN ARRAY), does not match the theoretical computerised pattern (LINE 647), then playing will be set to False (LINE 650) and hence, the game_over_animation() will flash.

```
647         elif clicked_button and clicked_button != pattern[current_step]:
648             game_over_animation()
649             save_score(score, high_score)
650             playing = False
651
```

Stubs

A stub is a small subroutine used in place of a yet to be coded subroutine. The use of stubs allows higher-level subroutines to be tested. Stubs do not perform any real processing; rather they aim to simulate the processing that will occur in the final subroutine they replace.

Whilst programming the game logic, I needed the challenge mode to function for all other game modes to run. However, I had not coded and installed all relevant files and audio. So, a very simple application was used to make sure that once it was clicked, the console terminal would simply display - "open challenge mode". This simple yet powerful Stub allowed me to continue working on other subprograms whilst this one was still functional.

```
908
909
910     def challmode():
911
912         print("open challenge mode")
913
914         pass
915
916
917
```

Peer Checking

Often errors will occur that seem to be impossible to correct for the original programmer. Colleagues on the development team are often able to see the problem from a fresh point of view. Peer checking involves other team members in the task of analysing each other's work to detect and correct errors.



Fig 5.30

Informal peer checking is crucial to the successful development of software products.

During development, in my SDD Class, I utilise the help of students similar in my position of making a game (Mukyl V. and Jesse J.). Below are main questions and answers asked and received from them:

Date	Asked to	Question	Answer				
30/05/2024	Mukyl	"How do you implement such good GUI features in your game? I would also like to implement similar attributes such as your refreshed frame rate per second"	<ul style="list-style-type: none"> - He sets his Frames Per Second rate to 60 - In the Godot Engine, he uses modules called: <table border="1" style="margin-left: 20px; margin-bottom: 10px;"> <tr> <td style="padding: 5px;">int</td> <td style="padding: 5px;"><code>get_ticks_msec () const</code></td> </tr> <tr> <td style="padding: 5px;">int</td> <td style="padding: 5px;"><code>get_ticks_usec () const</code></td> </tr> </table> <ul style="list-style-type: none"> - Looking into python, modules for FPS are called: <div style="background-color: #e0f2e0; padding: 10px; margin-top: 10px;"> <p style="margin: 0;"><code>pygame.time.wait</code></p> <p style="margin: 0;">pause the program for an amount of time</p> </div> <p>AND</p> <div style="background-color: #e0f2e0; padding: 10px; margin-top: 10px;"> <p style="margin: 0;"><code>pygame.time.Clock</code></p> <p style="margin: 0;">create an object to help track time</p> </div>	int	<code>get_ticks_msec () const</code>	int	<code>get_ticks_usec () const</code>
int	<code>get_ticks_msec () const</code>						
int	<code>get_ticks_usec () const</code>						
14/05/2024	Jesse	"I know that you are using Python's Visual Studio IDE. My question is that I have messed up and have accidentally closed	<ul style="list-style-type: none"> - Showed me shortcut on School Computers where I can recover my folders in the Visual Studio Code Settings Functionality 				

		my folder and I have forgotten my laptop at home. Is there any way I can recover the folders with files in them?"	
4/06/2024	Jesse	"I need help using the automatic debugger in VSC. The youtube videos are complicated and I want your specific help in determining how I can utilise this CASE tool the best way possible"	<ul style="list-style-type: none"> - Showed me the easiest way in using the debugging tool: - HELPED ME A LOT

RUN AND DEBUG

VARIABLES

```
> PLAY_CHALLMODE: <__main__.Button object at 0x1046d94d0>
> PLAY_EDUMODE: <__main__.Button object at 0x1046d8dd0>
> PLAY_MOUSE_POS: (645, 592)
> PLAY_NORMMODE: <__main__.Button object at 0x1046d9450>
> PLAY_RECT: <rect(240, 25, 800, 51)>
> PLAY_TEXT: <Surface(800x51x32 SW)>
> button: <__main__.Button object at 0x1046d94d0>
```

special variables

```
> function variables
> base_color: (255, 255, 255)
> font: <pygame.Font object at 0x1046df7e0>
> hovering_color: (0, 255, 0)
> image: <Surface(180x20x32 SW)>
> rect: <rect(970, 590, 180, 20)>
> text: <Surface(180x20x32 SW)>
```

WATCH

CALL STACK

```
Paused on breakpoint
normmode
main.py @0:1
play
main_menu
<module>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
aravzad@Aravzad-MacBook-Air:~/Desktop/Task 3, SDD % /usr/bin/env /usr/local/bin/python3 /Users/aravzad/.vscode/extensions/ms-python.debugpy-2024.0.0-darwin-arm64/bundled/libs/debugpy/adapter/.../debugpy/launcher 53782 — /Users/aravzad/Desktop/Task 3, SDD/main.py
python 2.5.2 (SDL 2.2.8, Python 3.11.4)
Hello from the pygame community. https://www.pygame.org/contribute.html
2024-06-30 11:30:49.827 Python[8000:340655] WARNING: Secure coding is not enabled for restorable state! Enable secure coding by implementing NSApplicationDelegate.applicationSupportsSecureRestorableState: and returning YES.
```

Ln 392, Col 1 Spaces: 4 UTF-8 LF Python 3.11.4 64-bit Go Live

Use of Clear Modular Structures

Software projects should be developed using a clear modular approach. This refers to the top-down design and through this good habit as a developer, I can easily determine the source of error in each module as stated in the terminal. In general the main program, in my case, “main.py” should be uncluttered and clear, as shown in the screenshot below. Through this, I can easily call subroutines effortlessly and on call.

```
264  
265  
266  
267 # In play(), add images for each respective game mode  
268 > def play():...  
332  
333  
334  
335  
336 # In main_menu(), might want to add some additional GUI features (ASK SIR), BG MUSIC?????  
337 > def main_menu():...  
383  
384  
385  
386  
387 # In options(), make sure that volume slider works. Additionally, create a link to the website/download to readme.pdf – MAKE THAT  
388 > def options():...  
464  
465  
466  
467  
468 # In edumode(), FIX THIS UP  
469 > def edumode():...  
631  
632  
633  
634  
635 # In normmode(), DONE!  
636 > def normmode():...  
790  
791  
792  
793  
794 # In challmode(), still need to do WHOLE – Should be easy after edumode() is done  
795 > def challmode():...
```

As seen in the screenshot above, I have utilised a bunch of functions - Allocating one function per screen. To call the function when I want to execute it once called on command, I do this:

```
if event.type == pygame.MOUSEBUTTONDOWN:  
    if OPTIONS_BACK.checkForInput(OPTIONS_MOUSE_POS):  
        main_menu() # Go back to the main menu
```

For example, to go “BACK”, I place this call in each function. In this case, from the “OPTIONS” Screen, once clicked will go to “MAIN MENU”

One logical task per subroutine

To implement this and make sure that each subroutine is not getting overcomplicated and is only performing relevant tasks and actions required, I have sorted each one by naming its program with a short and succinct name. Below is an example of how I have utilised this in Brain Beat:

The subroutine of LINE 328 is named “game_over_audio” - put quite simply, it plays this audio once the user is wrong. The sole purpose of this subroutine is to perform the task of playing games over audio and that is it.

```
328     def game_over_audio():
329         beeps = [Audio(BEEP1), Audio(BEEP2), Audio(BEEP3), Audio(BEEP4)]
330         for beep in beeps:
331             beep.play() # Play all beats synched to get the "BEEP!" – Game over sound
```

Similarly, looking below, the “confirm_exit” module plays whenever the user clicks “BACK” whilst in the middle of the game. Its sole purpose is to bring up a screen which says “Are you sure you want to exit? Progress may not be saved” creating assurance for the user.

```
228 def confirm_exit(screen, return_to_function):
229     # Create a semi-transparent overlay
230     overlay = pygame.Surface((WIDTH, HEIGHT))
231     overlay.set_alpha(128) # Set transparency
232     overlay.fill((0, 0, 0)) # Fill with black color
233
234     font = get_font(30) # Get font
235     message = ''
236
237     Some progress may not be saved.
238     Are you okay with this?''' # Confirmation message
239     lines = message.split('\n') # Split message into lines
240
241     yes_button = Button(image=None, pos=(800, 475), text_input="STAY", font=get_font(30), base_color="White", hovering_color=GREEN) # Stay button
242     no_button = Button(image=None, pos=(400, 475), text_input="QUIT", font=get_font(30), base_color="White", hovering_color="#800000") # Quit button
243
244     while True:
245         screen.blit(overlay, (0, 0)) # Draw overlay
246
247         y_offset = 200 # Initial Y offset for text
248         for line in lines:
249             text_surface = font.render(line, True, WHITE) # Render text line
250             screen.blit(text_surface, (150, y_offset)) # Draw text line
251             y_offset += 40 # Update Y offset
252
253         yes_button.update(screen) # Update stay button
254         no_button.update(screen) # Update quit button
255         pygame.display.update() # Update display
256
257         for event in pygame.event.get():
258             if event.type == pygame.QUIT:
259                 pygame.quit() # Quit pygame
260                 sys.exit() # Exit program
261             if event.type == pygame.MOUSEBUTTONDOWN:
262                 mouse_x, mouse_y = pygame.mouse.get_pos() # Get mouse position
263                 if yes_button.checkForInput((mouse_x, mouse_y)):
264                     return_to_function() # Call return function
265                 if no_button.checkForInput((mouse_x, mouse_y)):
266                     return True # Confirm exit
267             if event.type == pygame.MOUSEMOTION:
268                 mouse_x, mouse_y = pygame.mouse.get_pos() # Get mouse position
269                 yes_button.changeColor((mouse_x, mouse_y)) # Change stay button color
270                 no_button.changeColor((mouse_x, mouse_y)) # Change quit button color
```

Version Control and Regular Back up



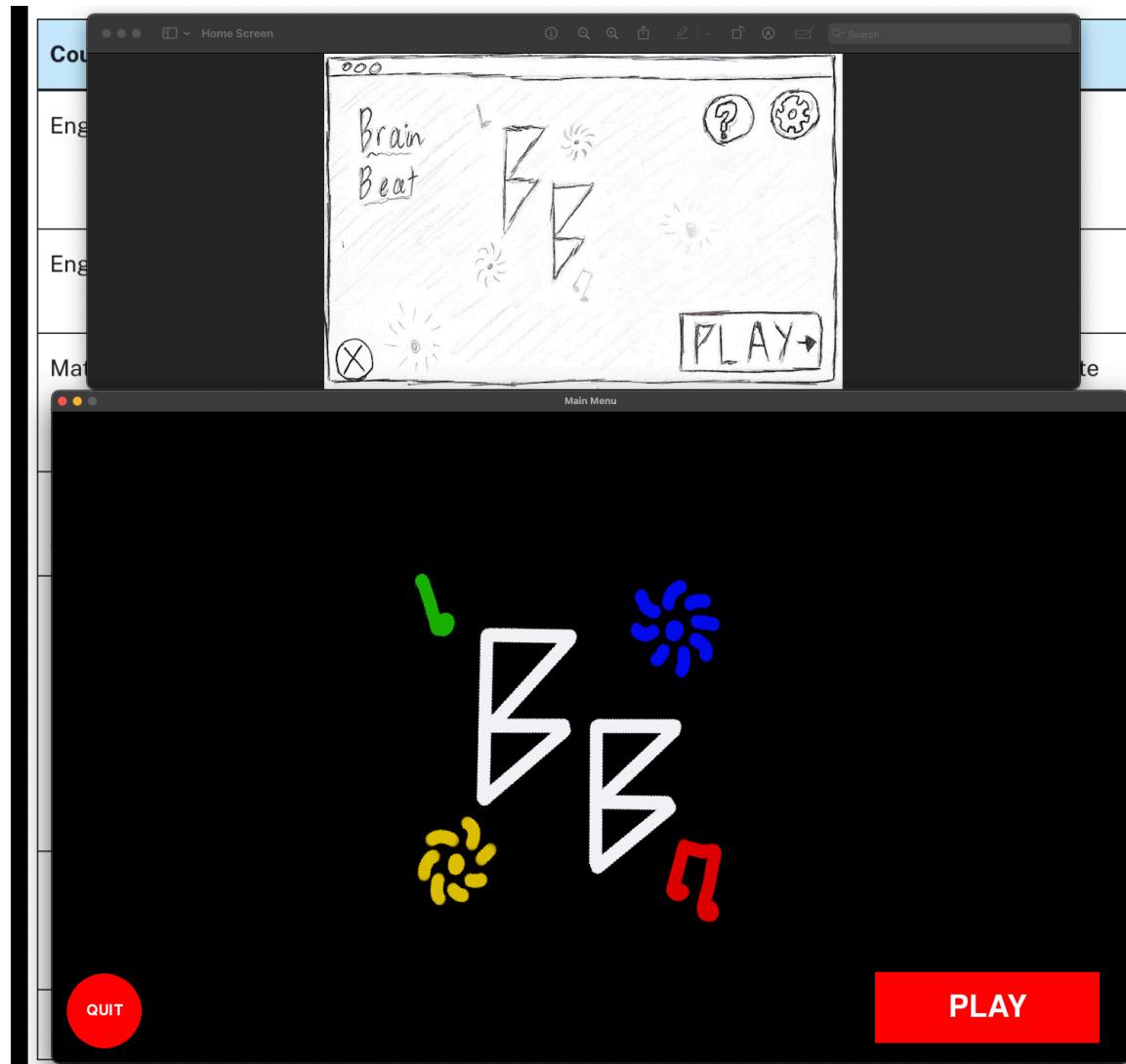
During coding all programmers should regularly save their work. This is to prevent loss of the source code should the power go out however it also allows different versions of the program or more often the current module or even subroutine to be stored. To make sure that I am regularly saving my work, and making sure that I can work on the code on different devices and computers, I utilised the application named 'Github'.

A screenshot of the GitHub Desktop application interface. At the top, there are three tabs: "Changes" (2), "History" (selected), and "Fetch origin". The "History" tab shows a single commit labeled "V1" with a commit message "aravzad -> e033b55 +385 -0". Below this, it says "29 changed files". On the left, a list of files is shown with their status: assets/Background.png, assets/BrainBeatLogo.jpeg, assets/Htp Rect.png, assets/Options Rect.png, assets/OptionsBG.jpg, assets/Play Rect.png, assets/Quit Rect.png, assets/StarsGifBg.gif, assets/font.ttf, assets/icon info.png, button.py, high_score.txt, main.py (selected), reset_high_score.py, score.txt, settings.py, sprites.py, and testing.py. On the right, the "main.py" file is displayed with its code. The code is a Python script using Pygame for a game menu. It includes imports for pygame, sys, settings, sprites, random, and button modules. It sets up the screen, loads background images, and defines a play function that fills the screen with black.

Version 1 “V1” is written on the top as this was the first code that I saved to Github so that I could work on it on different computers.

Elegance in Designs (Screen Design)

Initially, from V.01, my screen design looked like this:



However, the problem was that it looked too bland, the buttons had no depth in dimension and the user interface was simply boring. To enhance this, I read the Textbook, and viewed some online sources - I asked Artificial Intelligence (Chat GPT) this prompt:

“How can I make my GUI for my game better? List aspects” -

And this was the answer which related to the Textbook and Online Sources:

"Improving the GUI for your game can enhance user experience and make your game more engaging. Here are some aspects to consider:

1. **Visual Design and Aesthetics**

- **Consistency:** Ensure consistent use of colors, fonts, and styles throughout the game.
- **Theme:** Align the design elements with the overall theme of the game.
- **Animations:** Use subtle animations to make interactions feel smooth and responsive.

2. **Layout and Navigation**

- **Intuitive Layout:** Organize elements logically. Important buttons and information should be easily accessible.
- **Spacing:** Use appropriate spacing between elements to avoid a cluttered look.
- **Hierarchy:** Establish a clear visual hierarchy to guide players' attention to the most important elements.

3. **Feedback and Interactivity**

- **Hover Effects:** Implement hover effects to indicate clickable elements.
- **Sound Feedback:** Use sound effects to provide feedback on actions like button clicks.
- **Animations:** Use animations for transitions and to highlight changes in state (e.g., button press, score update).

4. **Accessibility**

- **Readable Fonts:** Use fonts that are easy to read and ensure text is large enough.
- **Color Contrast:** Ensure sufficient contrast between text and background.
- **Alternative Inputs:** Consider players who might use different input devices or have disabilities.

5. **Information Display**

- **Clarity:** Make sure information is clear and concise.
- **Real-time Updates:** Ensure that dynamic elements (e.g., scores, timers) update in real-time without lag.
- **Tooltips:** Provide tooltips for additional information or guidance.

6. **Customization Options**

- **Settings Menu:** Allow players to adjust settings like volume, screen resolution, and control bindings.
- **Themes:** Offer different themes or color schemes to cater to player preferences.

7. **Loading Screens and Transitions**

- **Loading Indicators:** Provide visual or textual feedback during loading times.
- **Smooth Transitions:** Ensure transitions between screens or game states are smooth and not jarring.

8. **User Guidance**

- **Tutorials:** Include a tutorial or help section to guide new players.
- **Pop-up Tips:** Display contextual tips to help players understand game mechanics.

9. **Performance Optimization**

- **Responsive Design:** Ensure the GUI performs well on different devices and screen sizes.
- **Minimal Lag:** Optimize the GUI to prevent lag or slowdowns, especially during intensive gameplay.

10. **Polish and Refinement**

- **Bug-Free:** Regularly test and fix any bugs related to the GUI.
- **Player Feedback:** Collect and incorporate feedback from players to continually improve the interface.

Implementing these aspects can significantly improve the user experience and make your game more enjoyable and accessible.”

IMAGE 1

Simplicity and Clarity:

- The main menu is straightforward with only three primary options (Play, Options, Quit), making it easy for users to navigate.

Consistency:

- The visual style is consistent with pixelated fonts and retro game aesthetics.
- Button styles and colours are consistent across the interface.

Feedback:

- The buttons change colour when hovered over, providing immediate visual feedback to the user.

Readability:

- The use of large, clear fonts ensures that text is easily readable.
- High contrast between the text and the background improves readability.

Alignment and Layout:

- Elements are well-aligned and spaced, contributing to an organised and balanced appearance.
- The main title is centred at the top, and buttons are aligned vertically on the left side, making the layout clean and structured.

Aesthetics:

- The visual design, including the logo and background, is aesthetically pleasing and thematic, matching the game's overall style.
- The "Scientifically Proven!" text is tilted to catch the user's eye without cluttering the main interaction area.

Visual Hierarchy:

- The most important elements (title and buttons) are given prominence through size and placement, guiding the user's attention effectively.

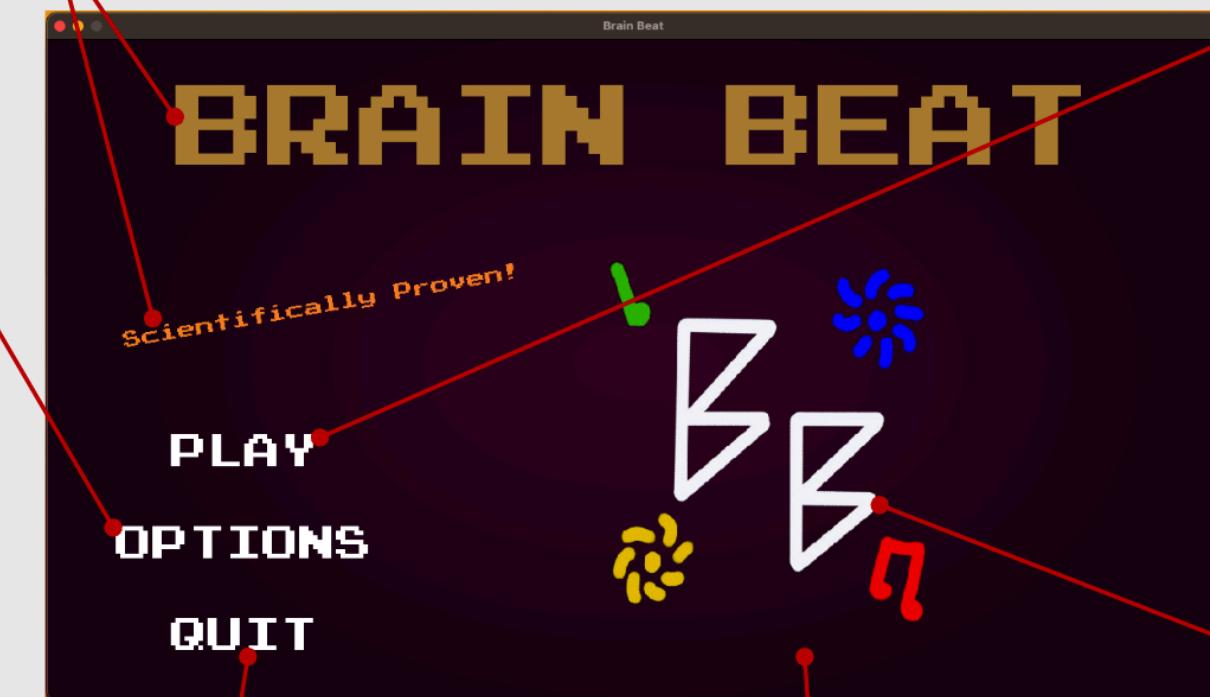
UI DESIGN 1 V.02 (MainMenu)

Eye captivating messages

Large fonts and words such as “scientific” makes people want to play the game!

Options

For User Friendly handling, options() such as: Clear High Score, Sound and How to Play
Redirection to Web



Quit Application

Quits application saving progress

Whitespace

Easy colours for user's eye to settle into

PLAY BUTTON

Proceeds to terminate Current Window, then open play()



Brain Beat Logo

Captivating, with Normal Mode Theme (Transparent)

IMAGE 1

IMAGE 2

Simplicity and Clarity:

- The options menu presents only a few essential settings, making it easy for users to understand and use.
- Each option is clearly labelled with a brief description.

Consistency:

- The visual style, including fonts and colours, is consistent with the main menu and overall game design.
- Button and text styles match the rest of the interface of the retro theme.

Feedback:

- Interactive elements like buttons provide visual feedback when hovered over, indicating that they are clickable.
- The volume slider updates in real-time as it is adjusted, providing immediate feedback to the user.

Affordance:

- The slider and buttons clearly indicate their interactivity through their design and placement.
- Text labels such as "Volume," "High Scores," and "How to Play" clearly indicate the purpose of each section.
-

Alignment and Layout:

- Elements are well-aligned and spaced, contributing to an organized and balanced appearance.
- The "BACK" button is positioned at the top left, adhering to common design conventions.

User Control:

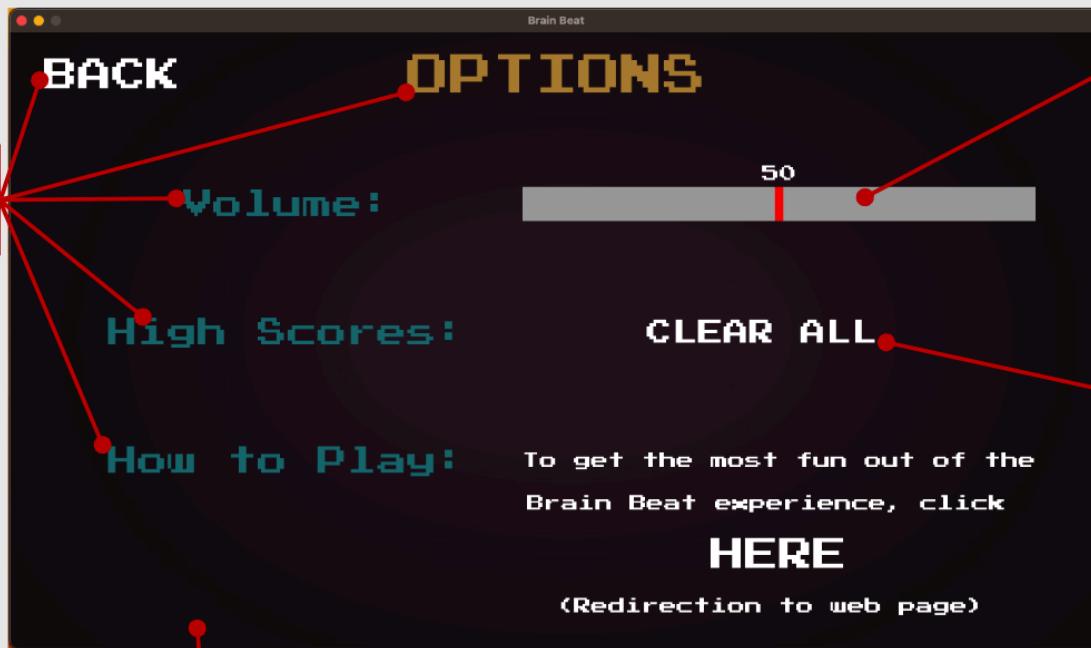
- Providing clear options for volume control, clearing high scores, and accessing a how-to-play guide gives users control over their experience.
- The confirmation prompt for clearing high scores ensures that users do not accidentally lose important data.
- The "HERE" button redirects users to a detailed tutorial, enhancing the overall user experience by providing comprehensive documentation.

UI DESIGN 2 V.02 (Options)

Note: All buttons in white hover different colour!

Eye captivating messages

Large fonts and words with consistent colour and font sizes for easy readability



Volume Slider

High quality slider scroll bar which reflects integer of what percentage the volume is at. Default = 50

Clear HS

Once clicked, a user assurance screen pops up so that user knows that all High Scores will be cleared

Whitespace

Easy colours for user's eye to settle into

IMAGE 2

IMAGE 3

Simplicity and Clarity:

- The screen offers three distinct game mode options, making it easy for users to understand their choices.

Consistency:

- Fonts, colours, and button styles are consistent with other parts of the game interface.
- The "BACK" button maintains its position and style from previous screens.

Feedback:

- The button for "Normal" mode changes colour on hover, indicating that it is selectable and interactive.

Visual Hierarchy:

- Game modes are visually distinct, with unique images and text to draw attention and distinguish between options.
- The "SELECT GAME MODE" header is prominently displayed to guide users.

Affordance:

- The images and text clearly indicate clickable areas for game mode selection.

Readability:

- Large, clear fonts and contrasting colours ensure that text is easily readable.

Alignment and Layout:

- Elements are well-aligned, creating a balanced and organised appearance.

Aesthetics:

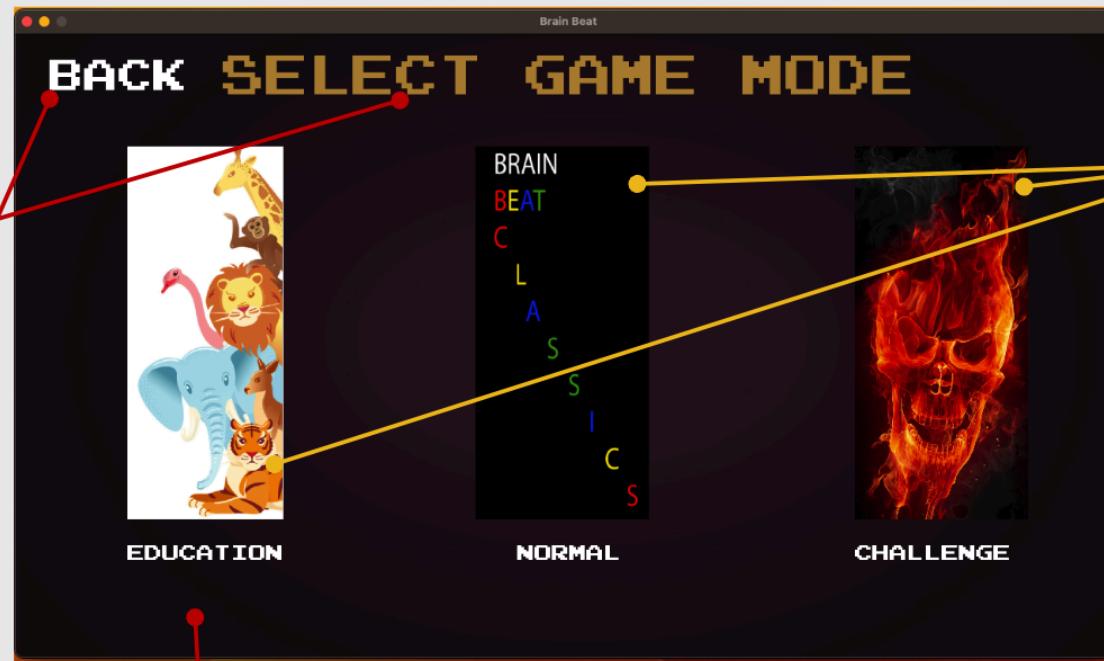
- The visual design is engaging and thematic, with vibrant images representing different game modes.

UI DESIGN 3 V.02 (Play)

Note: All buttons in white hover green!

Eye captivating messages

Large fonts and words with consistent colour and font sizes for easy readability



Whitespace

Easy colours for user's eye to settle into

Currently hovering over "NORMAL"

Respective Images

Each respective image represents the Game Mode that will proceed to go in:

Animals = Edu

Normal = Norm

Fiery Skull = Chall

IMAGE 3

IMAGE 4

Interactive Animal Faces:

- The animal faces should be visually appealing and easily recognizable for children. Use bright colours and clear illustrations.
- When hovered over, the animal faces should respond (e.g., change image and new sound) to indicate interactivity.

Audio Feedback:

- When an animal face is clicked or hovered, play the corresponding animal sound. This reinforces the association between the visual representation and the sound.
- Ensure the audio is pleasant, not too loud, and suitable for young children.

Score Tracking:

- Include a visible score display (e.g., "Score: 0") to motivate players.
- Update the score dynamically as the player progresses.

High Score Display:

- Show the highest score achieved by the player (e.g., "High score: 0").
- Encourage replay by setting a goal to beat the high score.

Back Button:

- Implemented a "BACK" button to allow users to return to the previous menu.
- Make sure it's clearly labelled and easy to find.

Responsive Design:

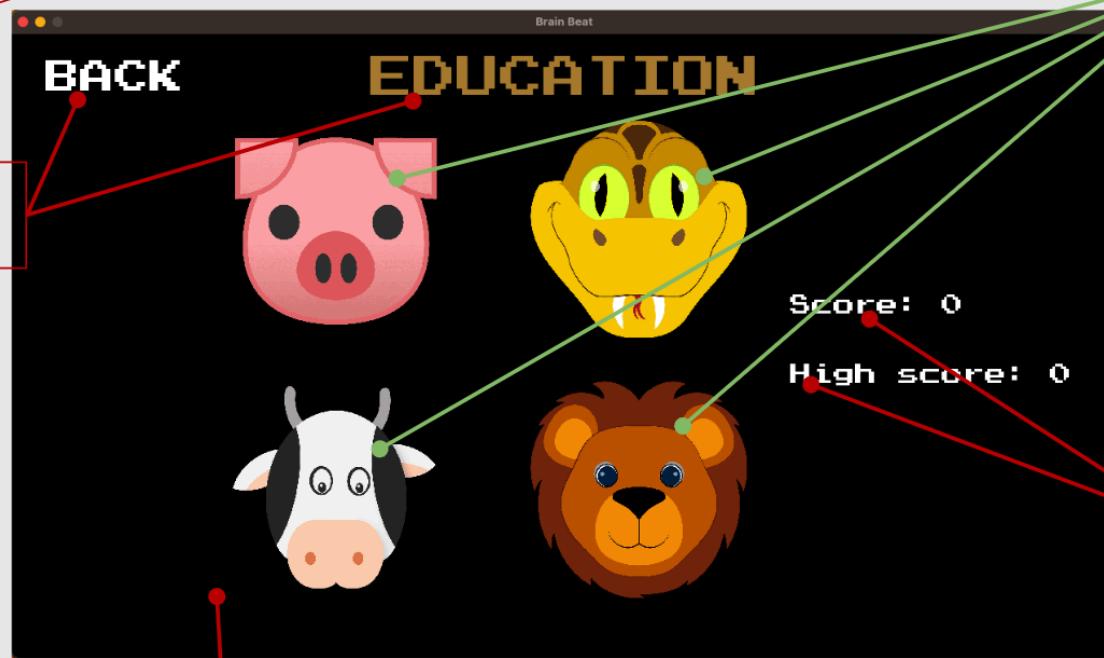
- Ensure the GUI works well on different devices (desktop, tablets, smartphones).
- Adapt the layout and size of elements based on the screen size.

UI DESIGN 4 V.02 (Edu)

Note: Once Hovered over animal, separate animation and audio plays and displays!

Eye captivating messages

Large fonts and words with consistent colour and font sizes for easy readability



Whitespace

Easy colours for user's eye to settle into

Respective Images

Each respective image represents the animal and its noise:

Pig: Oink

Snake: Hiss

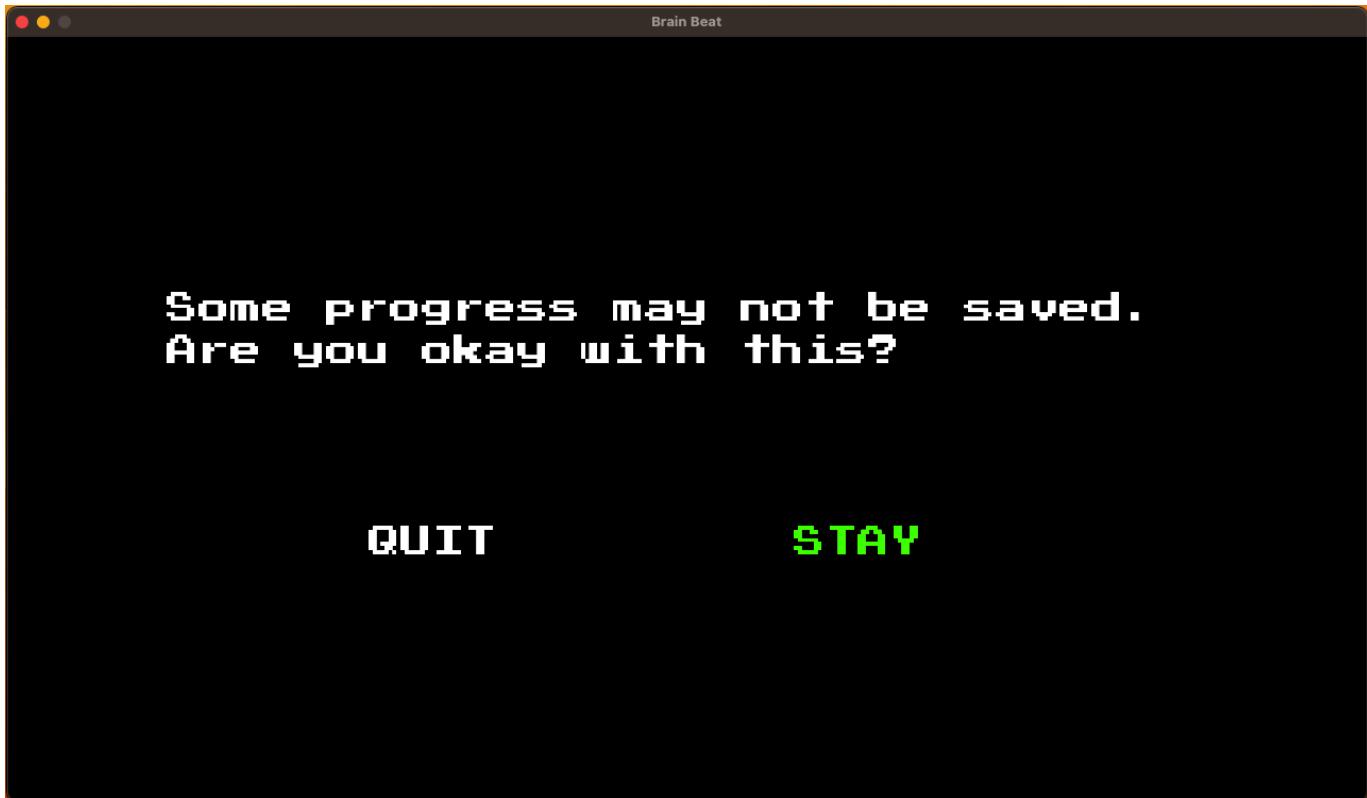
Cow: Moo

Lion: Roar

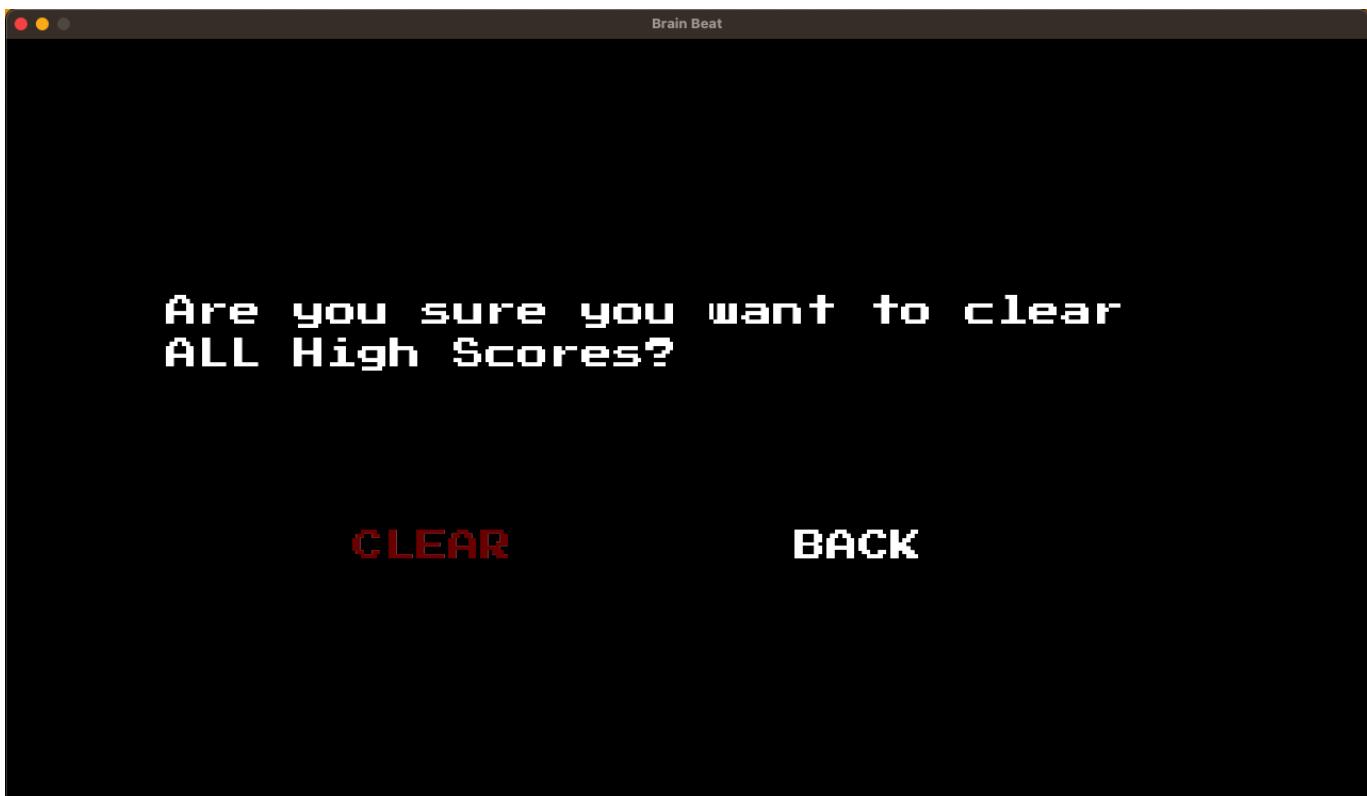
Scores

Score increments every time user is right, and once wrong, that score is saved to High Score.

IMAGE 4



REASSURANCE SCREENS FOR USER - integration of user help and assistance USED
message boxes, labelling, text tips



User Documentation

On Google Classroom

 Brain Beat User Manual V1

<https://docs.google.com/document/d/1RY9VKyYTxTxVHhZ0ugkm4lrxRQNdFZjL-ZBC2OidgTA/edit?usp=sharing>

TESTING AND EVALUATING SOFTWARE SOLUTIONS

The testing and evaluation process is central to a software development company's quality assurance. Our major aim is to ensure the product meets the original design specifications created during the 'Defining and Understanding the Problem' stage.

Beta Testing

As mentioned before, in order to do this stage, I have incorporated a Google Forms Sheet to fill out by 5 people:

- Link:
<https://forms.gle/JYWELj1fddJrUkg26>

Results from the Google Sheets:

By automating the google workspace so that whatever response I get there will go on a Google Sheets (Excel Duplicate) file, I was able to analyse my application with 5 people. These 5 people were: Grandma, Friend, Brother, Mum and Dad

RESULTS AND ANALYSIS:

From the spreadsheet:

Pros	Cons
<ul style="list-style-type: none">- Smooth clicking and transitioning between windows- Nice Game in general (9 average)- Normal Mode was the best (10 average)	<ul style="list-style-type: none">- Volume Slider not functioning- Images in play() subroutine looked a bit cheap and ruined aesthetics- Challenge Mode needs to be harder

To fix these Cons, I will need to:

- Modify Slider() Module
- Modify the timings in challmode() to decrease timings from 3 seconds to 1-1.5seconds
- Research the internet for better images - OR modify the play() subroutine to make sure that once hovered over these images, the game description pops up

Beta testing

Testing of the final solution by a limited number of users outside the software development company using real world data and conditions.

Timestamp	Are there any features from the User Interface that you dislike? If so, what is it about this feature that throws you off?	When navigating through the game, what aspect would you like to be made easier?	How are you feeling about Education Mode?	How are you feeling about Normal Mode?	How are you feeling about Challenge Mode?	How are you feeling about Brain Beat as a whole?	Have you seen our YouTube Channel?	What is the most difficult part of using our app?
6/24/2024 22:42:13	The Volume Slider does not work	Aspects are all easy	8	10	9	9	No	The volume slider
6/25/2024 20:30:27	volume scroll bar - does not work	nothing all good	10	10	8	9	No	other than volume very good
6/25/2024 17:33:40	I dont like the images from the internet in N/A		9	10	7	9	No	everything is good just fix the volume, change images and make challenge mode harder
6/27/2024 9:35:42	Volume slider and Photos in sleecting ga	very smooth transitioning, i like it!	9	10	7	9	No	nothing difficult, would want to make challenge mode more difficult
6/29/2024 13:37:17	The Volume Sliding does not work!	Smooth clicking, and like the hover feature!	10	10	8	9	No	Change the Volume Slider! I want to play it!

BASED ON THESE RESULTS, I USED SPREADSHEET SKILLS TO DETERMINE AVERAGES AND MOST IMPORTANTLY, WHAT NEEDED THE MOST WORK:

How are you feeling about Education Mode?	How are you feeling about Normal Mode?	How are you feeling about Challenge Mode?	How are you feeling about Brain Beat as a whole?
8	10	9	9
10	10	8	9
9	10	7	9
9	10	7	9
10	10	8	9

Average: 9.2	Average: 10	Average 7.8	Average: 9
--------------	-------------	-------------	------------

Quality Assurance

Efficiency: Brain Beat ensures efficient use of resources through well-structured code and optimised algorithms, providing smooth performance across various system configurations. The game modes and GUI, including the volume slider, have been designed to minimise processing time and enhance user experience.

Software Quality

The extent to which desirable features are incorporated in to software so it can reliably and efficiently continue to meet the needs of users.

Integrity: The game maintains data integrity by implementing robust validation for user inputs, ensuring high scores and settings are correctly stored and accessed. Security measures are not in place due to the privacy of the game being non restrictive and available to all.

Reliability: Brain Beat is designed to be reliable, with the ability to recover quickly from failures and adapt to updates in operating systems. The game includes mechanisms for updating and maintaining functionality over time, ensuring consistent performance.

Usability: The user interface of Brain Beat is intuitive and consistent, making it easy for users to learn and navigate. Features like the options menu, volume slider, and clear high score functionality enhance usability, while advanced users benefit from streamlined interactions.

Accuracy: The game performs its functions accurately according to specifications, with thorough testing of algorithms and game logic to ensure correct behaviour across all modes. Internal and intrinsic documentation is provided as seen in the code to make implementation easier.

Maintainability: Brain Beat's code is modular and well-documented, allowing for easy maintenance and updates. The structure supports future enhancements and corrections, ensuring the game can evolve with minimal disruption.

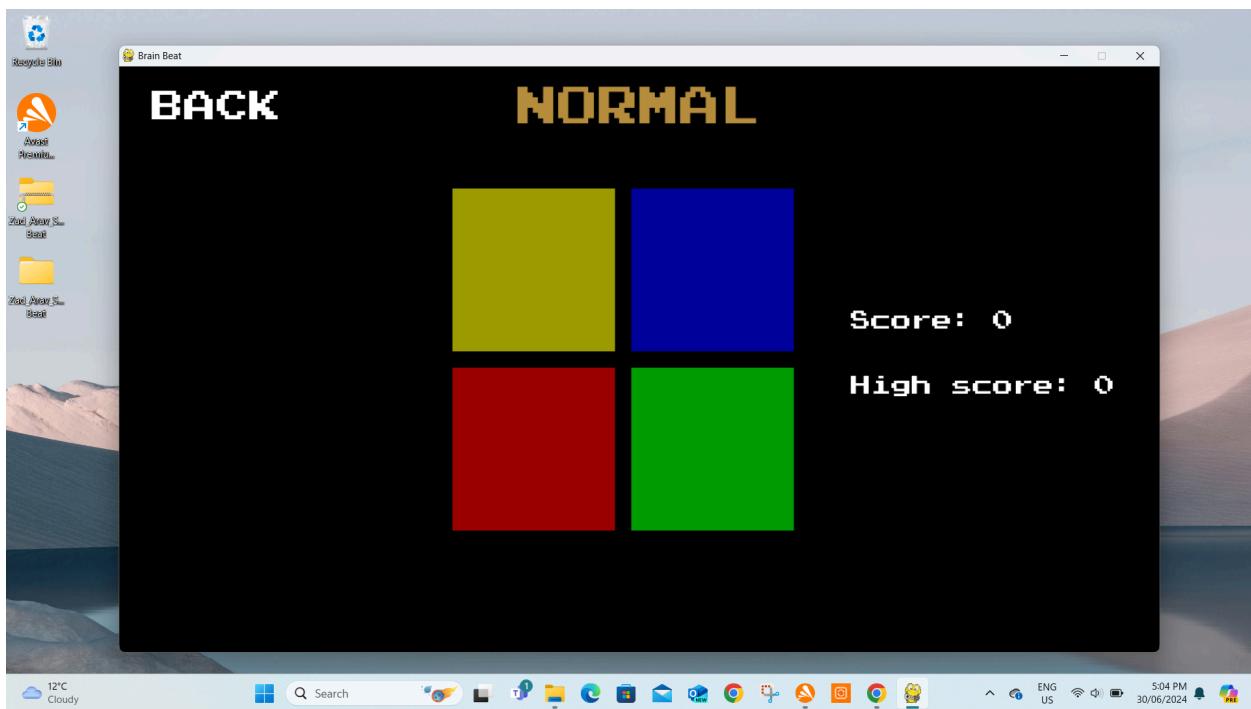
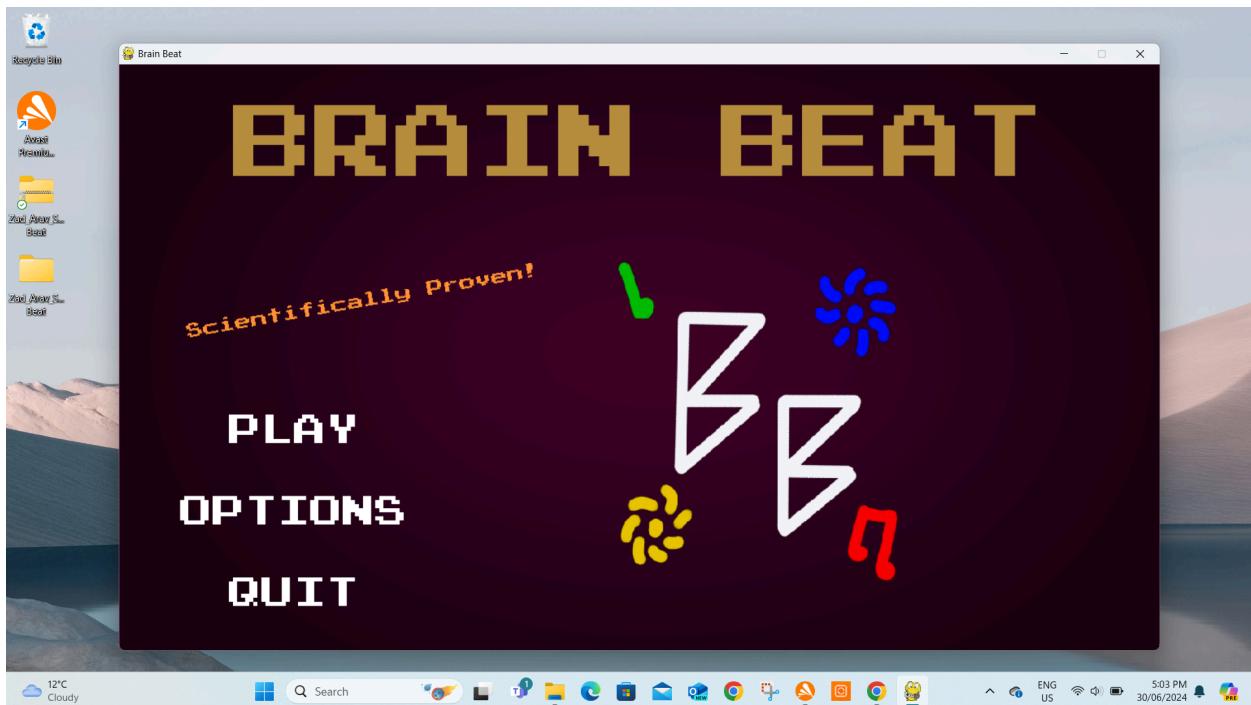
Testability: The game is designed with testability in mind, allowing for comprehensive testing of individual modules and the entire system. This ensures that all aspects of the game function as intended and meet user requirements. *Testing is shown below*

Re-usability: Brain Beat's code includes reusable modules and routines, enabling their application in future projects - For example the Fonts function is used to keep a consistent font subject to change for size.

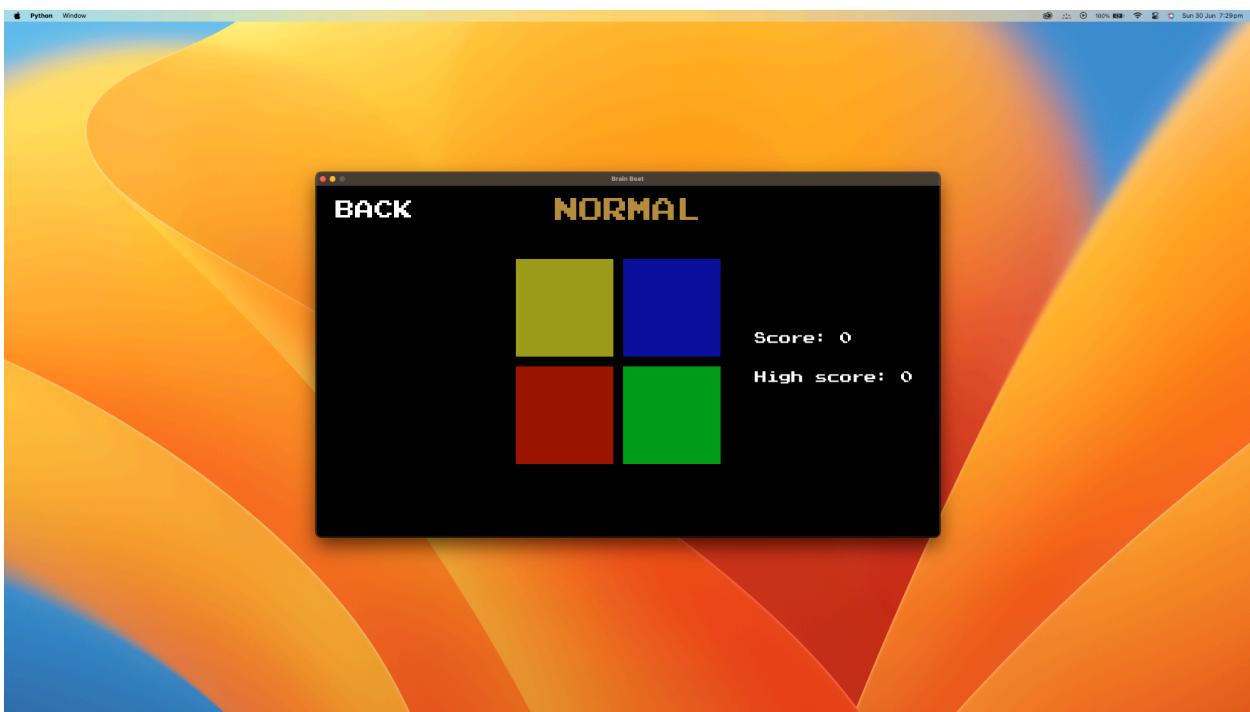
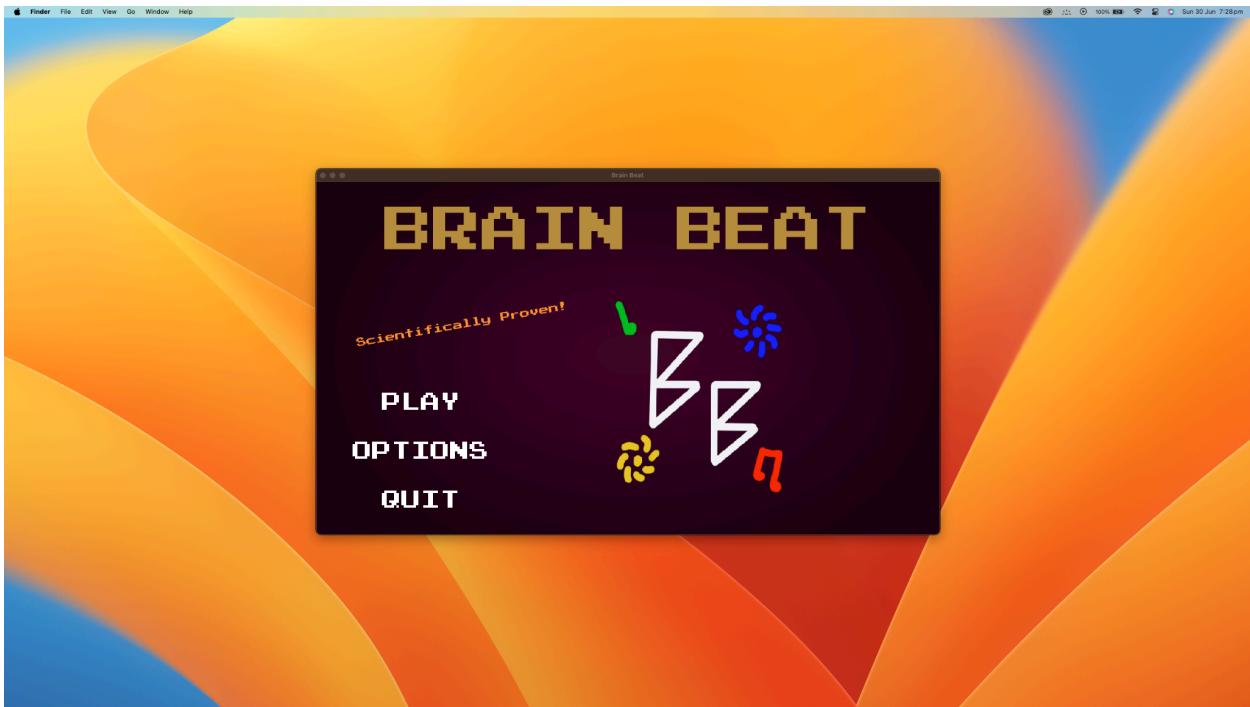
```
def get_font(size):
    return pygame.font.Font("assets/font.ttf", size) # Load a font of a specific size
```

Interoperability: Brain Beat's code can run on Windows and MacOS: Below are Screenshot Evidences:

Running successfully on Windows from the .exe file



Running successfully on MacOS as a .dmg file (MacOS version of .exe - Same purpose of executable) - LOOKS LARGE ON MACOS BECAUSE CONNECTED TO A LARGE MONITOR

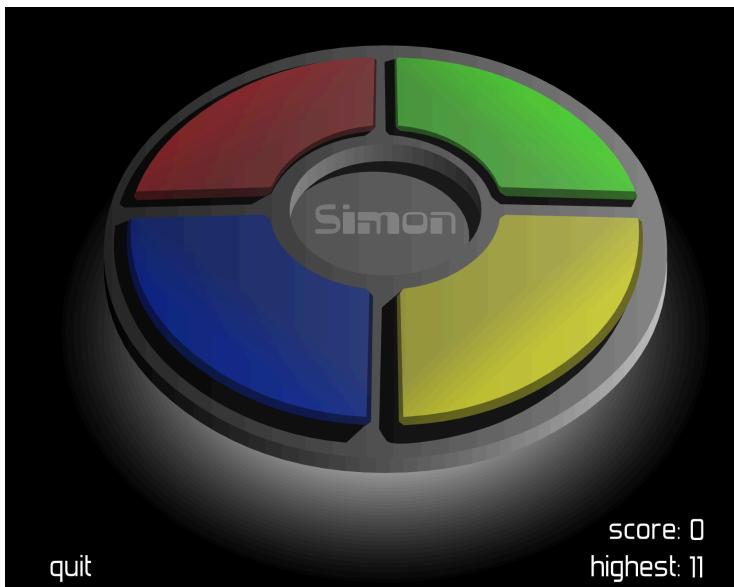


Response Times

The response time is the time taken for a process to complete. The process could be user activated or it could be activated internally by the application. Response times are dependent on all the system components, together with their interactions with each other and other processes that may be occurring concurrently. 0.1 seconds is preferable. When running the code, it is easy to run and all response times in Brain Beat are WELL UNDER 0.1 SECONDS.

Benchmarking

Brain Beat's standard is the Simon Memory game. Online, at <https://freesimon.org/> the GUI of the game looks like this:



Benchmarking

The process of evaluating something by comparison to a standard. Benchmarking enables competing products to be fairly compared.

I wanted to get the similar original gist of the game with a twist of my own so I proceeded to implement a different order of colours.

Coming to technicality, the FPS on the original simon game is also 60, so for a nice smooth gameplay, I also implemented 60 ticks per second.



Expected Output vs Actual Output

1.



On left is Actual Output
Below is expected output

Expected “BRAIN BEAT” to be in centre

Found out the correct coordinates using VSC PyDebugger:



```
88     MENU_MOUSE_POS = pygame.mouse.get_pos()
89
90     MENU_TEXT = get_font(100).render("BRAIN BEAT", True, "#b68f40")
91     MENU_RECT = MENU_TEXT.get_rect(center=(300, 100))

```

main.py Git local working changes - 1 of 1 change

```
90    90     MENU_TEXT = get_font(100).render("BRAIN BEAT", True, "#b68f40")
91    91     MENU_RECT = MENU_TEXT.get_rect(center=(640, 100))
92    92     MENU_RECT = MENU_TEXT.get_rect(center=(300, 100))

```

2.

High Score not resetting despite clicking the Clear All High Score Button. - Actual (LEFT)
Expected (RIGHT)



Was the final product's expectations met by the final specifications?

The Brain Beat game aimed to incorporate several key components as outlined in the initial design proposal. These components included different game modes (Education, Normal, Challenge), a user-friendly GUI with options such as a volume slider, the ability to clear high scores, and a user manual directory. Additionally, the game was to feature interactive and dynamic visual elements, such as changing button colours on hover, and a well-structured gameplay.

I successfully integrated all the major components of the game as planned:

Different Game Modes: Education, Normal, and Challenge modes were implemented, providing varied experiences for users.

Interactive GUI: The GUI included features like a volume slider, buttons for clearing high scores, and access to a comprehensive user manual.

Visual Elements: The buttons were designed to change colour on hover, enhancing the user experience.

Despite these accomplishments, the volume slider was not perfected. It did not function as for volume and was just a slider, which affected the overall user experience. This one shortfall meant that the final product did not fully meet the initial specifications. While the game was functional and included all the planned components, the imperfect volume slider highlighted the importance of thorough testing and refinement.

Overall, the project was a success in many areas, but the issue with the volume slider serves as a reminder of the need for meticulous attention to detail and the value of testing and improvement.

EVALUATION

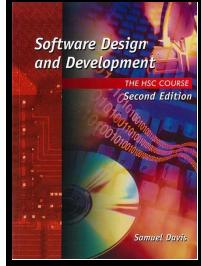
Peer Evaluation

Above under Testing - I used the Peer Evaluation so I can better understand how software design and development works - and improve myself as a student with a good work ethic.

Self Evaluation

In developing the Brain Beat game, I have learned a great deal about game design, user interface principles, and the intricacies of software development. I've developed my skills in coding, particularly with Python and GUI libraries such as Pygame, as well as in project management and problem-solving. These skills will be crucial in any future software development roles, especially in positions that require creating engaging user experiences or educational tools. I plan to use this project in applying for Engineering Degrees through portfolios as this project was difficult and required a lot of time and attention. My strengths lie in my attention to detail and my ability to learn and apply new concepts quickly, for example as mentioned in the Google Forms, my GUI was nice and the refresh rate between windows was smooth and without bugs. Though I recognise that my time management and debugging skills could use improvement. The project was challenging, primarily due to the complexity of ensuring a smooth, bug-free user experience and the need to integrate various features seamlessly. In hindsight, I would have allocated more time for testing throughout the process to refine the game further - especially aspects such as the Volume Slider. The greatest obstacle was debugging the volume slider and ensuring the game's difficulty levels were visibly different and unique to one another. I overcame this through persistent research and by seeking advice from online developer communities such as Stack Exchange, Github, and Stack Overflow, and YouTube. Useful resources included online documentation, forums, and feedback from peers, all of which were invaluable in navigating the project's challenges.

REFERENCES

Resource Type	Source	Purpose
Website	Simon (game) , Wikipedia	Critical Analysis on existing product
Website	19 Captivating Facts About Simon (electronic Memory Game) , Facts.net	Critical Analysis on existing product
Website	Team Gantt , Team Gantt	Planning of Software Solution
Website	5 Great Memory Games for Adults , The Clare	Target Audience (Adults + Elderly)
Website	The Importance of Playing Memory Games , Curious World	Target Audience (Children)
Book	Author: Sam Davis Date Published: 25/12/2011 "Software Design and Development, THE HSC COURSE, Second Edition" 	TASK 1, PROGRAMMING PROJECT REPORT (Whole Assignment, Multiple Aspects)
Website	Chat GPT , “do a critical analysis on the Simon Memory Game - roughly 300 words” (PROMPT)	Critical Analysis of Existing Product
Website	Board of Studies Student Forum , the difference between intrinsic and internal documentation	Implementing the Solution
Website	https://ytmp3s.nu/Sj09/	Youtube to Mp3
Website	https://www.youtube.com/watch?v=t99ULJjCs_aM	Animal Song for Edu Mode
Website	https://cms1.publuu.com/flipbooks	PDF File to Online Site
Website	https://userpilot.com/blog/user-experience-survey-questions/	Why Python should be used
Website	https://www.oyolloo.com/is-python-good-for-game-development/#:~:text=Frequently%20Asked%20Questions,Is%20Python%20A%20Suitable%20Language%20For%20Game%20Development%3F,choice%20for%20many%20game%20projects.	Why Python should be used

Website	https://stackexchange.com/	Stack Exchange
Website	https://stackoverflow.com/questions/tagged/pygame	Stack Overflow Forums
Website	https://www.youtube.com/	Youtube for video help
Website	Chat GPT , “Is it possible for once a user clicks on a button, can a pdf file be downloaded - or can a link be opened in their browser? - if yes then make a pygame code for it - a plane blank window with the button leading to any website such as "google.com" will do” (PROMPT)	RESULT BELOW

```
python
Copy code

import pygame
import webbrowser

# Initialize Pygame
pygame.init()

# Set up the display
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Open Link Example")

# Define colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
GREEN = (0, 255, 0)

# Define the button
button_rect = pygame.Rect(WIDTH // 2 - 100, HEIGHT // 2 - 50, 200, 100)

# Define the font
font = pygame.font.Font(None, 36)

def draw_button():
    pygame.draw.rect(screen, GREEN, button_rect)
    text = font.render("Open Google", True, BLACK)
    text_rect = text.get_rect(center=button_rect.center)
    screen.blit(text, text_rect)

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

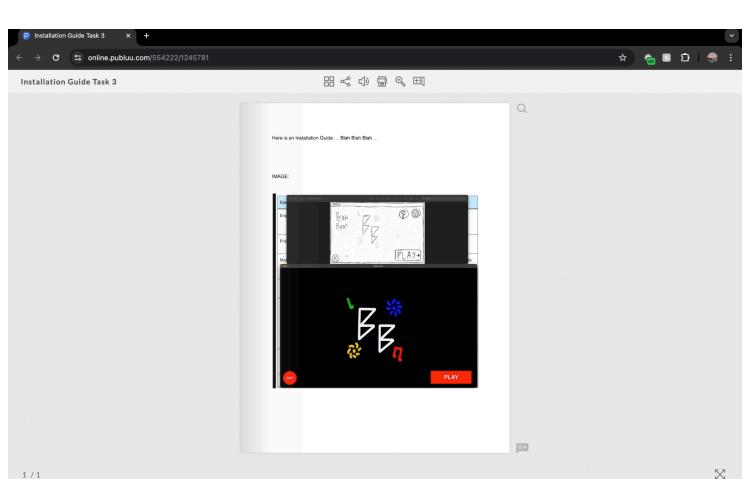
        if event.type == pygame.MOUSEBUTTONDOWN:
            if event.button == 1: # Left mouse button
                if button_rect.collidepoint(event.pos):
                    # Open the web link
                    webbrowser.open("https://www.google.com")

    # Clear the screen
    screen.fill(WHITE)

    # Draw the button
    draw_button()

    # Update the display
    pygame.display.flip()

# Quit Pygame
pygame.quit()
```



ABOVE - AFTER IMPLEMENTING CHATGP ON SAMPLE PDF, ABOVE RESULT

LEFT - AI PROMPT ANSWER