Big Snake's Brain

1. Project Overview

It's a mathematical + snake game. The game will provide an equation for the player and there will be a character (our snake) that will move like the original snake game. There will be more than one food. In every food there will be a different number in the food. The objective is to solve the equation and when you get the answer, you've to move the snake to eat the food with the right answer. If your answer is wrong, the health bar will decrease. And when the health bar is empty, you'll lose the game. When the snake hits the border, it will appear on the other side and when the snake eats itself, its health will decrease.

2. Project Review

Relevant existing project : Snake game

Improvements :

- Make the food more than one (original only has one food).
- Create a new section for showing score, health bar, equation.
- Make every food have a number inside.
- Make the health bar decrease when you eat the wrong answer and eat itself.
- Make the score move up when you eat the correct answer.
- If the health bar is empty, the game will display the final score and ask the player whether to play again or not.

3. Programming Development

3.1 Game Concept

- A mathematical fuse with the snake game.
- On the top of the screen, there will be a section for displaying the health bar, score, equation.

- Our character can move up, down, left, right. It can eat food just like the original game but when it eats, it won't grow.
- The food will be more than one. Each one will have a number inside and only one of them will be the answer for the equation.
- If you eat the wrong answer or eat yourself, the health bar will decrease.
- If you eat the correct answer, the score will go up.
- If the health bar is empty, the game will stop and display your score, then it will ask you if you want to play again or not.

3.2 Object-Oriented Programming Implementation

- class Snake
  - Create and control the snake.
- class Food
  - Create foods and assign numbers to them.
- class Game
  - Control the gameplay.
- class Equation
  - Create the equation.
- class HealthBar
  - Create the health bar. If the player eats the wrong food, the health bar will decrease.
- class Score
  - Create the score. Starts at 0, then goes up when the player eats the correct food.

might have more classes in the future

3.3 Algorithms Involved

Rule-based logic : the player must answer the question correctly and the player must follow the snake game rule.

4. Statistical Data (Prop Stats)

4.1 Data Features

- Player's score : Collect player's score in every game.

- Player’s mathematical accuracy skill : Calculate and collect player’s accuracy by (number of times player got the question wrong)/(number of all question)*100
- Character movement : Collect number of times player hits up, down, left and right button.
- Health bar status : Collect health bar status when the health changes e.g. collect health bar status when the snake eats itself (health decrease).
- Time status : Collect time status when something happens e.g. collect time status when health decreases or collect time status when the score increases.

4.2 Data Recording Method

Everything will be stored in a CSV file.

4.3 Data Analysis Report

The data will be presented with a **table**.

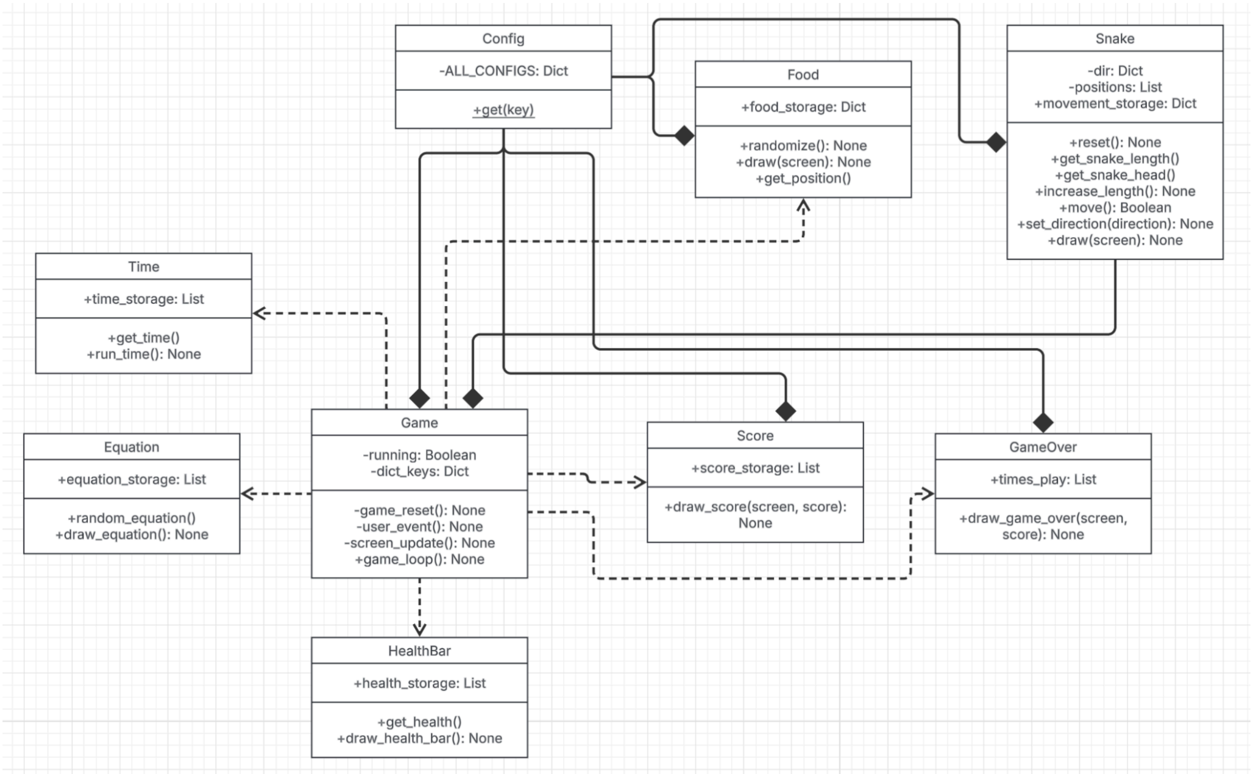
- For a player's score, there will be max, min and average score.
- For a player's mathematical accuracy skill, it will show all the equations and answers that the player got when the player played the game and state which equation the player got correct and which equation the player got wrong. It will also show the player's accuracy by calculating the number of questions the player got wrong divided by the number of all questions the player got times 100 (all of the questions will be generated randomly with the range of 1 to 100).
- For character movement, there will be a record for how the player controls the character. The table will display how many times the player hits the up, down, left and right button.
- For health bar status, there will be a number of how much health is left in every round.
- For time status, there will be a time record for every important data (for example, it will record the time when the snake eats the food, it will record the time when the health bar decreases) and the data will be displayed in a table. The record data, for example, are the time the player got each question right (reset when the player got the question

right), the time the player used in that one specific game (start the clock when the game starts and end the clock when the player dies), etc.

5. Project Timeline/planning how you will approach and complete your project

Week	Task
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	Proposal submission #2
3 (24 March)	Proposal submission #3
4 (31 March)	Proposal submission #4
26 March - 2 April	Project complete 20 %; ● Complete Snake class ● Complete Config class
3 April - 9 April	Project complete 30 %; ● Complete Food class ● Complete Score class
10 April - 16 April	Project complete 50 %; ● Complete GameOver class ● Complete HealthBar class ● Complete Equation class
17 April - 23 April	Project complete 75 %; ● Complete Time class ● Complete Game class ● Double check all of the code
24 April - 11 May	Final submission, Project complete 100 %; ● Collect all data

6.UML diagram



7.Collected Data Table/Statistical Data Revision

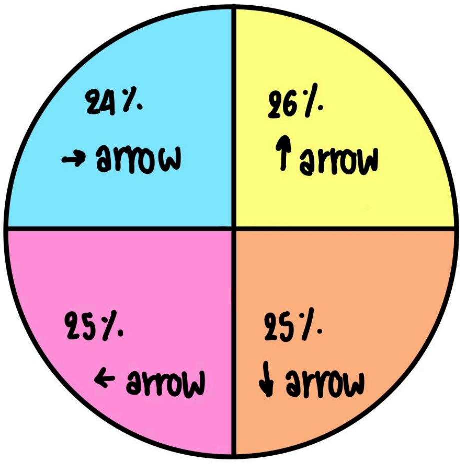
	Why is it good to have this data? What can it be used for?	How will you obtain 50 values of this feature data?	Which variable (and which class) will you collect this from?	How will you display this feature data (via summarization statistics or via graph)?
Player's score	Used to tell the player their best score and they might want to beat it.	Play 50 games.	Variable score_storage, class Score	With line graph
Player's accuracy	Players can use this data to develop their	Play 50 games.	Variable health_sto	With boxplot

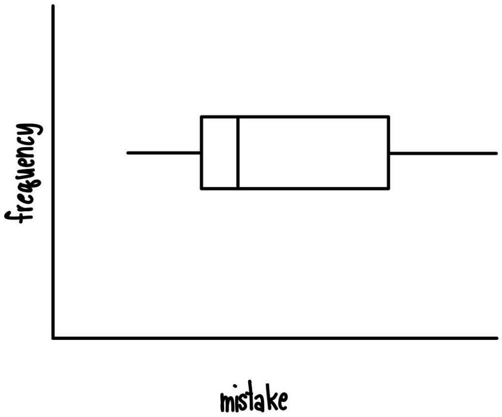
	mathematical skill.		rage, class HealthBar.	
Character movement	Players can use this data to track how they control their character.	Play the game until there are at least 50 values.	Variable dict_keys, class Game.	With pie graph
Health status	Players can use this data to track how much they've made mistakes.	Play 50 games.	Variable health_storage, class HealthBar.	With summarize information in table
Time status	Players can use this data to track how much time they have played, the time when they made mistakes, etc.	Play the game until there are at least 50 values.	Variable time_storage, class Time.	With information in table

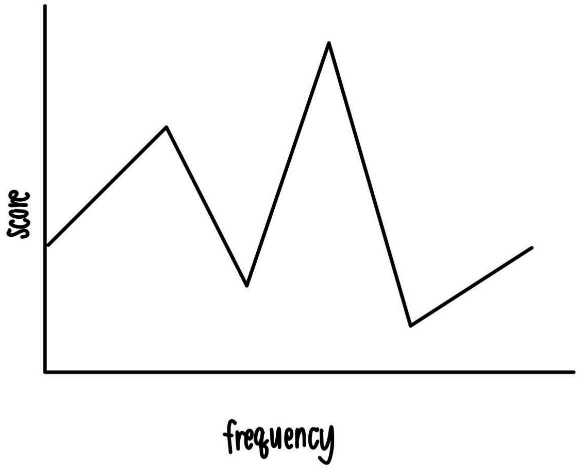
8. Graph

	Feature Name	**Graph objective**	**Graph type**	**X-axis**	**Y-axis**	**Section**
Graph1	Player's accuracy	To display player's accuracy in mathematical skill and can help them improve their skill.	Boxplot (distribution)	How much mistake the player make each game	Frequency	-

Graph2	Character movement	To display character movement so players can track how they control their character.	Pie graph (proportion/distribution substitution)	-	-	1.up arrow button 2.down arrow button 3.left arrow button 4.right arrow button
Graph3	Player’s score	To display player’s score and their best score.	Line graph (time-series)	Frequency	Score	-







9. Table

Table1 (Time status) : display time status so the player can track how much time they have played, the time when they made mistakes, etc.
Statistical value : average time played, max time played and min time played.

Table2 (Health status) : display health status so the players can use this data to track how much they've made mistakes.
Statistical value : min, max, average and SD

10. Document version

Version: *1.0*
Date: *31 March 2025*