Group 3 Project Document: Educational Snake Game

INTRODUCTION:

For our project we built an educational gaming application. This application is built on the concept of the traditional Snake game.

As children love to spend most of their time playing games, we want to provide an environment for them to also learn in a fun way. In order to achieve this goal, we implemented the following features:

- Finding the word from the given letters
- Solving an arithmetic problem

In addition to the above, we wanted to have:

- A leaderboard containing the individual scores for each category.
- Multiple levels of complexity in the game

We used an Agile approach, with the team collaborating iteratively. We followed Agile principles such as: welcoming any changes suggest that would benefit the game or make the process more streamlined/efficient; try and simplify steps so we can keep track of the areas needing completing and more effectively deal with errors; regular stand-ups that ensure communication, allowing us to keep track of project process.

The initial idea was to have 3 separate games that cover: numeracy, spelling, memory.
Once beginning to work on the project, our expectations of what could be achieved within the timescale changed, therefore we dropped the memory game. This is an example of how our agile approach was beneficial, as it drastically reduced the time requirements for the project.

We decided on a set format for the code e.g. where to put comments, using snake case instead of Camel Case. We also agreed to regularly meet to discuss our progress and understand what each person was working on; as outlined in the implementation section of this report, this proved challenging.

Within this report, the reader can expect to find sections on: the background of the project; design specifications and requirements; implementation and any challenges faced; running tests.

BACKGROUND:

The application builds on the concept of the traditional snake game, by including educational features like solving simple arithmetic equations and unscrambling the given words. At the end of the game, players are allowed to save their names with the latest score, to keep a track on their progress.

Our main aim of building this application is to help children 'Learn Through Play'.

Pygame library is used for building the gaming application. The user is given an option to play a 'Number Game' or a 'Word Game'. Following are the instructions and descriptions of how each game works.

Number Game:

- Initially the game starts with a score of 0 and maximum of 3 lives for the snake.
- There is an arithmetic equation that is displayed on top of the screen.
- Answer to the equation is displayed as one of the random numbers on the screen.
- If the correct number is picked up by the snake:
  - The score increases by 1.
  - A new arithmetic equation is displayed on top of the screen.
  - New random numbers along with the answer to the equation are displayed for the snake to pick.
- If a wrong answer is picked up by the snake:
  - Total number of lives has decreased by 1.
  - Another random number is generated and displayed along with the remaining numbers.
- If the snake hits itself or the edges of the screen, the total number of lives decreases by 1.
- Game is over when the snake loses all 3 lives.

Word Game:

- Initially the game starts with a score of 0 and maximum of 3 lives for the snake.
- The word to be guessed is displayed with blank spaces at the top of the game.
- All the letters of the word to be guessed are scattered on the screen for the snake to pick them up in the correct order.
- As the snake picks up each letter one by one, this appears correspondingly on top of the screen.
- Once all letters are picked up the game checks if the word formed is correct.
- Player's score increases by 1 when the correct word is formed, otherwise it loses a life.
- A new word with the corresponding letters is displayed on the screen.
- If the snake hits itself or the edges of the screen, the total number of lives decreases by 1.
- Game is over when the snake loses all 3 lives.

SPECIFICATIONS AND DESIGN:

Functional Requirements:

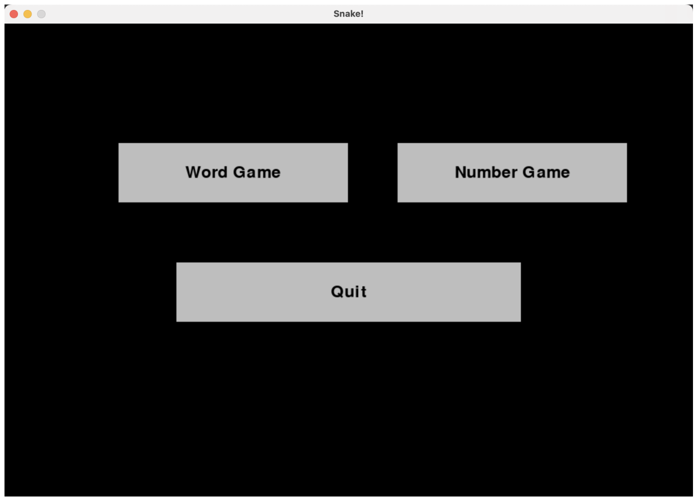
- A window must open to run the game
- Some kind of trigger to start the game
- The user must be able to move the snake
- Something must appear at the top of the grid (equation or blank space for word)
- Lives must be lost if the snake hits the boundary or collects the wrong answer
- There must be a way for the game to end

Non-functional Requirements:

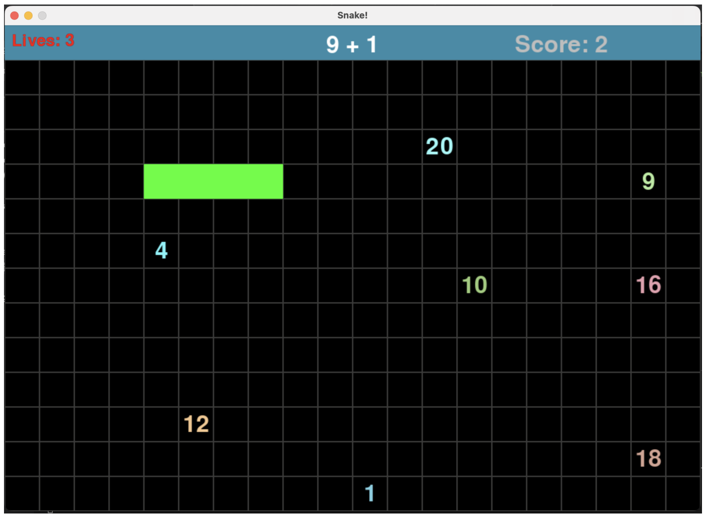
- The snake must move at a speed that allows players to play the game without losing all three lives quickly i.e. allows time to process the information on screen and collect the right answer
- Current number of lives should be visible to the player so they know how close they are to game over
- Lives, equation/word, score, should all be visible on screen to allow for a more cohesive player experience

Design

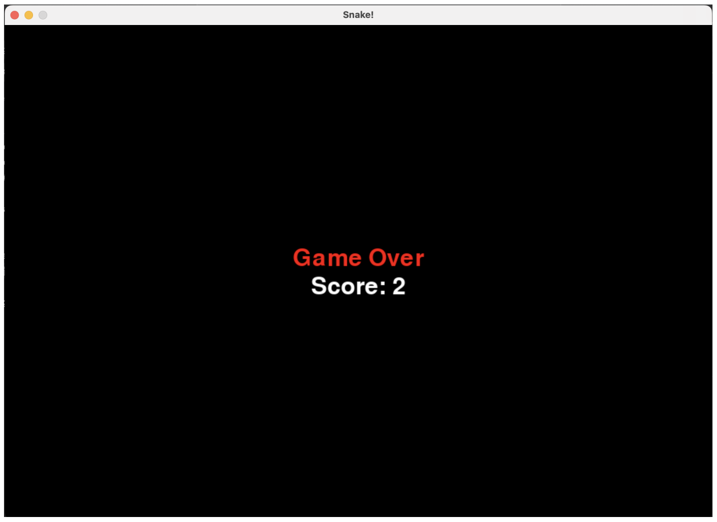
- Ensuring a smooth user experience with the programme:
    - Making it clear where to click
    - Effective labeling of information such as buttons
    - No repetition of data
- Creating code that is easy to read and understand:
    - Utilise classes to create a clearer sequence of code
    - Split code up where possible so it can be more easily managed and problems isolated.
- The intended database would store the information gathered during the game over screen.
    - Game over screen would take an input and then save this to the database
    - This is discussed further in the following section



Window with 3 buttons appears when the programme is run. The user has a choice of word game or number game or to quit.



Once a game has been selected, the game immediately starts. Snake always starts from top left. The score is always 0 and lives always 3 when the game starts. The equation or space for the word is within the top blue bar. Numbers/letters appear within the grid with a distance between them all, and they vary in colour to make it more visually accessible.



Once 3 lives have been lost, whether by collecting incorrect answers or hitting the boundaries, then the game over screen is displayed. Originally this included the 'Enter your name' function for the player to add their name to the leaderboard (more information in following section). The score is presented to the player on screen. The player must now exit the game.

IMPLEMENTATION AND EXECUTION:

Development and team roles
As previously stated, we took an Agile approach to the project. Each member of the team specialised in specific areas: Alisha worked on the database, Lima did some work on a possible timer, Jaya implemented the word game, Timea created the base code for one of the games, Sian worked on introducing the lives/game over aspect of the game. This list is in no way exhaustive - please refer to the project log for further clarity over each member's involvement in the project.

Agile development

- Peer review pull requests and assign people to the pull request. Code should only be merged when thoroughly reviewed and tested by another group member.
- Refactoring the code and modularisation - some major changes required throughout development, such as creation/removal of classes, relocating code, etc, to improve clarity of the project.

<u>Implementation process</u>
Achievements

- We were quite successful in arranging meetings outside of the sessions, despite 3-4 of the team working full time also.
- We were conscious of each other's varying availability.
- Tasks were split evenly amongst the group and we made sure we could reach out to each other if there were any problems.

- We created not only one but two games and created a start screen to choose one of the two.
- Some reasonably large changes were made throughout the development process which we adapted well to.

Challenges

- Organising *enough* team meetings: it was difficult to find a convenient time for all 5 of us every week; we needed more frequent but short meetings to check on the status of the project.
- Communication was not regular in Slack: there was a lack of communication across Github and Slack; project management was agreed via Github but any problems about specific issues were raised in Slack. On reflection, we should have made the process of internal feedback clearer for all team members.
- No regular updates from team members about the status of the issues we were working on. There were times where team members did not respond to messages, did not show up to meetings, or offered to work on code that didn't end up being completed/requesting a pull request for.
- Incorrect code pushed to Github: a lack of thorough testing prior to pushing to Github; failure to update local repository before pushing to Github; list of current issues was not acknowledged by some team members before creating a new one, resulting in duplicates and two people working on the same issue at once; some pull requests were not assigned a reviewer.
- Organisation of issues: many issues were created without an assignee; some issues were worked on but no-one was assigned to it; duplication of issues; lack of comments / further information on issues, allowing for further misunderstandings to take place.
- Incorrect code was merged: some code was not reviewed before merging despite team agreement to peer review before merging; took time to fix the errors caused by the merging of incorrect code; possibly took the wrong solution with this challenge as could have reverted the pull request.
- There was not sufficient understanding of the code by all members of the team: big changes were made to the code without clear explanations of what had been changed (in Github or Slack); no comments were added on Github, which made project management challenging.
- Connecting the database: the database was completed and the code set up to connect, but this was not completed. As a solution, we removed the external database from the final playable game.

TESTING AND EVALUATION:

Testing strategy

As the code had been modularised into separate classes, we had to test each file separately. This helped us to separate the testing tasks easily among team members.

- Each team member was responsible for writing test cases for a few files.

- All the testing files are contained in a ‘tests’ folder.
- This ‘tests’ folder is then used for running all the test files of the application.
- Unit testing has been used for testing different parts of the application.
- We had done manual testing while building the code to check if each part was working successfully.
- Regression testing after every change to ensure that the change introduces no unintended breaks.
- Reviewed the changes made by another team member and tested other parts of the application to make sure everything runs successfully with the new changes.

Functional and user testing
Various test cases were created which involved testing core functions integral to the programme working, and testing whether pieces of code created for a more user-friendly experience, such as display messages, were working correctly.

System limitations:
- Circular import errors: modularisation was helpful for organising code, but it introduced restrictions due to circular dependencies.
- External Dependencies: use of external libraries and packages introduced version compatibility issues.

CONCLUSION

In conclusion throughout the development process we remained committed to our objectives and consistently applied an agile approach to tackle challenges. The utilisation of the pygame library enabled us to bring our ideas to life, and we employed modularisation and code reviews to maintain code quality and enhance collaboration among team members.

Our educational gaming application stands as a testament to our dedication to merging education and entertainment. We believe that our project showcases the potential of gamification in learning and emphasises the importance of adaptability and collaboration in the software development process.

As we move forward, we recognise the importance of ongoing communication and regular updates within the team. The lessons learned from this project will guide us in improving our teamwork, project management, and technical skills in future endeavors.