SET

Chu Li（1082477）

Tongxin Hu（4734114）

CONTENTS

1. ORIENTATION

1.1 BACKGROUND

The SET game is grounded in mathematical and cognitive principles, particularly involving pattern recognition, set theory, and quick decision-making. Each card in SET displays four features: number (1, 2, or 3), color (red, green, or yellow), shading (solid, striped, or open), and shape (ovals, squiggles, diamonds). Players must identify sets of three cards where each attribute is all the same or all different.
The mathematical foundation of SET stems from combinatorics and pattern recognition, and its core challenge is how to efficiently identify valid sets while providing real-time interactive feedback, which is our main goal in realizing the SET game.

1.2 PROBLEM

The following is a further refinement of the research question:

1.2.1 Pattern Identification Efficiency (PIE)

The primary objective of studying the SET game is to measure how efficiently players can identify valid sets among randomly distributed cards. We denote this efficiency with PIE_NNN, where NNN is the number of cards. This measure is critical for understanding cognitive speed and pattern recognition capabilities in varying conditions. Calculating PIE_NNN is complex because the difficulty of identifying patterns increases significantly as more cards are introduced, necessitating the development of better psychological and educational strategies to enhance rapid visual cognition.

1.2.2 Average Identification Time (AIT)

For a SET game session with a certain number of cards NNN, the time it takes for players to identify each set can vary. Our aim is to determine the average identification time across these sessions. This average is crucial for assessing how quickly a player processes complex visual information and makes decisions. Understanding AIT is important because it provides insights into cognitive processing speed and efficiency in both educational and competitive contexts.

1.2.3 Set Complexity Constant(μ)

The set complexity constant μ is a key metric in our studies of the SET game. It measures how the difficulty of finding a set increases on average as the number of cards grows towards a theoretical maximum. This metric is instrumental in understanding how players' pattern recognition abilities scale with increased game complexity. Calculating μ is essential because it helps quantify the cognitive load and potential training methods needed to improve efficiency in visual recognition tasks under pressure.

1.2.4 User Interface Design and Interaction

Designing an intuitive and easy-to-use user interface is another core issue. How to clearly display card characteristics through the interface while providing checking, validation and feedback functions is an important part of user experience optimization.

1.3 APPLICATION

SET has various applications in many fields, especially in education, cognitive science research, psychology, etc.

1.3.1 Education: Cognitive and Logical Skill Development

SET requires players to quickly recognize and analyze patterns on the cards, helping to enhance observational skills, attention, memory, and problem-solving abilities. These skills are also crucial for learning other subjects.

1.3.2 Cognitive Science Research: Brain Processing Studies

Psychologists and neuroscientists use SET to study how the human brain processes visual information, makes rapid decisions, and handles complex categorization tasks. For example, measuring the time it takes to find a SET can provide insights into the brain's efficiency in processing different cognitive tasks.

1.3.3 Psychological Testing: Reaction Time and Attention

SET can be used to test an individual's reaction times and ability to concentrate. In psychological experiments, these tests can help diagnose attention deficit hyperactivity disorder (ADHD) or other cognitive impairments.

1.4 MAIN CHANLLENGE

1.4.1Computational Complexity

Algorithms for generating all possible sets from a deck of cards in the SET game are computationally complex, primarily because they must verify that each potential set of three cards meets the game's unique conditions across four separate characteristics. As the number of cards increases, the computational effort and storage requirements for these algorithms escalate significantly due to the combinatorial explosion in potential sets.

1.4.2 Constraint Enforcement

In SET, ensuring that each set of three cards adheres to the rule of all same or all different across each characteristic poses a significant challenge. This requires checking the characteristics of all combinations, which becomes computationally expensive as the number of cards and thus potential combinations increases.

1.4.3 Pattern Recognition Complexity

The algorithm must efficiently handle the recognition of patterns among diverse visual stimuli, which includes various colors, shapes, numbers, and shadings. This complexity necessitates a versatile implementation that can adapt to recognize these patterns without significant changes to the core recognition logic.

1.4.4 Diverse Set Processing

The game involves processing different configurations of cards, each presenting a unique set of visual cues. The processing algorithm needs to handle this diversity effectively, ensuring that all potential sets are recognized regardless of the visual complexity, or the number of cards involved.

1.4.5 Visualization and User Interface

Developing intuitive visualization tools to represent and interact with the cards, especially in digital versions of the SET game, poses both technical and design challenges. These tools must clearly display cards and their characteristics, manage user interactions, and provide feedback on the legality of sets in real-time.

1.4.6 Theoretical Analysis and Experimental Verification

There may be deviations between theoretical strategies for finding sets quickly and the actual outcomes observed in gameplay, whether digital or physical. It is necessary to design experimental setups and analytical methods to verify the effectiveness and practicality of proposed strategies for both casual play and competitive scenarios. These methods help refine player strategies and algorithmic approaches to the game.

2. OUR CODE

In this chapter, we will explain the methods and code in the Python files dedicated to the SET game. This chapter covers the design and implementation of the algorithms used to identify sets within the game, which is played with a special deck of cards featuring various attributes such as color, number, shape, and shading.

2.1 ALGORITHM DESIGN

2.1.1 Overview of SET Game Algorithm

The SET game is a card game that challenges players to identify sets of three cards based on four attributes where each attribute among the cards must either all differ or all be the same. This section outlines the design and implementation of the algorithm that powers the SET game.

Initialization and Setup:

- Deck Initialization: The game begins with the creation of a deck of 81 cards, each uniquely defined by combinations of four attributes: number (1-3), symbol (oval, triangle, diamond), shading (solid, striped, open), and color (red, green, yellow). The (generate_deck) function systematically creates this deck and shuffles it to ensure a random distribution of cards for the gameplay.
    - Defining Attribute Options: numbers = [1, 2, 3]: each card can have 1, 2 or 3 symbols. Symbols = ['oval', 'triangle', 'diamond']: symbols can be ovals, triangles or diamonds. Colors = [COLOR1, COLOR2, COLOR3]: colors can be COLOR1 (red), COLOR2 (green), COLOR3 (yellow). Shadings = ['solid', 'transparent', 'open']: the fill can be solid, striped, or empty.
    - Generate 81 cards: Enumerate all possible combinations of attributes using list derivatives
    - Shuffle cards: Use shuffle(deck) to disrupt the deck so that the order of the cards is different each time the game is played, ensuring randomness.
- Game Board Setup: Twelve cards are dealt from the deck to form the active game board. These cards are displayed in a grid layout on the screen, facilitating easy player interaction.
    - Determine the grid layout of the game board: Use (rows, cols = 3, 4)to set the game board to consist of 3 rows × 4 columns = 12 cards.
    - Calculate the starting coordinates to center the deck: (start_x) Calculates the starting position of the cards horizontally. (card_width * cols + card_spacing * (cols - 1)) Calculates the total width of all cards (including spacing). Subtract it from WINDOW_WIDTH and divide by 2 to make sure the cards are centered overall. (start_y) Similarly.

2.1.2 Gameplay Mechanics:

- Card Selection and Interaction: Detects player clicks, determines whether a card is selected or deselected, and manages the currently selected card.
    - Card selecting: Use (event.pos) to get the (x,y) coordinates of the mouse click. Use index % cols to calculate the column position of the card in the current row (the same applies to rows). Traverse all cards, check whether a card is clicked, and determine whether the mouse click position is within the rectangular area of a card. Finally, add the selected card to (selected_cards).
- The game tracks these selections and evaluates them using the (is_set) function to determine if they form a valid set based on the game's rules.
- Set Validation: A group of three cards forms a set if, for each attribute, the values are either all the same or all different across the cards. This core rule is central to the gameplay and is rigorously checked by the algorithm each time cards are selected.
    - is_set(card1, card2, card3) Checks if these three cards match the SET rules.
    - If the player finds a SET, the player scores +1 and “You win” is displayed on the UI. The player removes the three cards from the (table_cards), then takes the new three cards from the deck and puts them in the (table_cards).

  - If not SET, uncheck and clear the list. The game page instructs the player to “try again” and unchecks the three cards.

2.1.3 Dynamic Game Progression:

- Set Identification and Replacement: When players successfully identify a set, the cards involved are removed from the board and replaced with new ones from the deck, if available. This dynamic adjustment keeps the game engaging and ensures continuous play until no more sets can be found or the deck is depleted.
- Continuous Checking for Sets: The algorithm continuously checks for possible sets within the current cards on the board, allowing for hint features and ensuring that players are aware when no more sets are available, which can signal the end of the game.

2.1.4 End-Game Scenarios:

- Exhaustion of Cards: The game ends when all cards have been used, and no further sets are possible with the remaining cards on the board.
- No Possible Sets: If at any point, no sets can be formed with the existing board cards and no replacements are available from the deck, the game concludes.

2.1.5 Additional Features:

- Scoring System: The player and the computer play a match within a specified time, the player starts first, if the player finds a set, he scores a point, and vice versa the computer scores a point. Adding a competitive edge to the game.
- Time Challenges: Introducing time constraints can enhance the difficulty, requiring players to identify sets within a given timeframe.
- Temporary Message: Present a temporary message after a player wins or loses, indicating whether the player has won or tried again.

2.1.6 Interface design

Pygame was used to implement the drawing and interaction functions of the user interface. The core design includes:

- Card drawing:
  Use the (draw_card) method to draw the shape, color and selected state of the card.
  Use (draw_symbols) to draw different symbols, colors and shadings on the card with the determined graphic shape.
- Real-time feedback
  After the player clicks on the card, the program will update the selected status in real time and verify whether a SET is formed.

2.2 DATA STRUCTURES

Card and Deck Management

- Card Object: Represents each card with attributes like number, symbol, color, and shading, essential for validating sets.
- Deck Array: A list of Card objects used for shuffling and dealing cards to the game board.

Game Board Layout

- Tableau (Game Board): Holds the active cards on the table in a list, allowing for easy access and manipulation as the game progresses.

User Interaction and Tracking

- Selected Cards: A list tracks the player's currently selected cards, simplifying the process of adding or removing cards as they are chosen or deselected.

Set Validation and Replacement

- Set Validation: Directly compares card attributes to determine set validity.
- Replacement Cards: Manages drawing new cards from the deck to replace sets removed from the tableau.

Efficient Set Searching

- All Possible Sets: Uses itertools.combinations to search all card combinations on the board to assist in features like hints or automatic set detection.

Game State Management

- Score and Time Tracking: Utilizes simple data types to keep track of scores and game time efficiently.

2.3 TIME COMPLEXITY

Analyzing the time complexity of the SET game involves understanding the computational effort required to perform essential operations such as shuffling the deck, identifying sets, and updating the game state.

2.3.1 Deck Operations

- Shuffling: The deck shuffling, performed using Python’s random.shuffle, operates in O(n)O(n)O(n) time complexity, where nnn is the total number of cards in the deck (81 cards). This ensures that every card has an equal probability of appearing in any position in the deck.

2.3.2 Set Identification

- Finding All Sets: The most computationally intensive task in the SET game is identifying all possible sets among the cards displayed on the table. Given that each set consists of 3 cards, and up to 12 cards are displayed at any time, the number of combinations to check is given by the combination formula (n3)\binom{n}{3}(3n), which translates to (123)=220\binom{12}{3} = 220(312)=220 combinations. Each combination check involves constant time comparisons across four attributes (number, symbol, color, shading), leading to a complexity of O(1)O(1)O(1) per set check. Thus, the overall complexity for this operation is O(C)O(C)O(C), where CCC is the number of combinations, which remains manageable within the gameplay context.

2.3.3 Game Updates

- Replacing Cards and Updating the Board: Once a set is found and removed, replacing these cards involves removing elements from the list of cards (deck) and adding them to the list of active cards (tableau). These operations are O(1)O(1)O(1) for adding or removing a card at the end of the list. The time complexity for updating the game board after finding a set is dominated by the time taken to find a set, as the replacement itself is a constant-time operation.

2.3.4 Overall Game Complexity

- Total Game Complexity: The overall complexity for running a game cycle — including shuffling, set finding, and board updating — is O(n+C)O(n + C)O(n+C), where nnn is the total number of cards, and CCC represents the combination checks for set validation. This is efficient enough to allow for real-time interaction and response in the game environment.

2.3.5 Optimizing Time Complexity

Efficiency improvements in SET game simulations include:

- Efficient Card Selection: Utilizing data structures like sets for quick lookup and updates can reduce the overhead during card selection and set validation.

- Memoization of Set Checks: Storing previously validated combinations can avoid redundant checks, especially in scenarios where the board changes minimally between player moves.
- Incremental Set Checking: Rather than re-evaluating all combinations each time the board updates, the game can incrementally check only new combinations arising from newly added cards.

Resulting Time Complexity Analysis

- Initial Time Complexity: The initial implementation, primarily dealing with combinations and direct checks, works effectively within the O(n+C)O(n + C)O(n+C) complexity for managing the game.
- Optimized Time Complexity: Through strategic data handling and minimizing redundant operations, the optimized approach significantly enhances efficiency, maintaining overall gameplay responsiveness and reducing computational delays.

2.4 METHOD EXPLANATIONS AND IMPLEMENTATION COMPLEXITY

generate_deck()

- Purpose: This method creates a shuffled deck of cards, each with unique combinations of attributes like number, symbol, color, and shading.
- Implementation: Iterates through all possible combinations of attributes to form a comprehensive list of card objects. After generating all cards, the deck is shuffled to ensure random distribution for gameplay.
- Complexity: Though generating the fixed set of combinations is an O(1)O(1)O(1) operation due to the constant set size, shuffling them using the Fisher-Yates algorithm scales with O(n)O(n)O(n), where nnn is the number of cards.

is_set(card1, card2, card3)

- Purpose: Checks if three selected cards constitute a 'set' based on game rules—either all similar or all different across each attribute.
- Implementation: Employs a sequence of conditional checks for each attribute across the three cards. Returns True if all attributes comply with the rules, otherwise False.
- Complexity: This method has a constant time complexity, O(1)O(1)O(1), as it checks a fixed number of attributes and conditions.

find_all_sets(cards)

- Purpose: Identifies all possible sets within the current display of cards.
- Implementation: Uses a combinatorial approach through itertools.combinations to assess each trio of cards from the displayed set, applying is_set to validate each combination.
- Complexity: The computational overhead is primarily dependent on the number of combinations, calculated as (123)=220\binom{12}{3} = 220(312)=220, making this an O(C)O(C)O(C) operation, where CCC is the total combinations checked.

draw_card(card, x, y)

- Purpose: Renders cards visually on the game screen, including their symbols and selection status.
- Implementation: Utilizes Pygame's graphics functionalities to draw each card at specific coordinates with visual details according to the card's attributes and selection state.
- Complexity: Each call to draw_card has a constant complexity, O(1)O(1)O(1), since the drawing operations do not vary with the number of cards but depend on rendering performance.

draw_symbols(card, x, y)

- Purpose: Draws the symbols on each card based on its attributes like shape, color, and shading.
- Implementation: Based on the card's number attribute, determines the positions and draws each symbol using geometric calculations and Pygame's drawing tools.
- Complexity: Directly proportional to the number of symbols on the card, it remains within a manageable constant complexity, O(1)O(1)O(1), because the maximum symbols per card is three.

display_message(text)

- Purpose: Shows game messages dynamically, such as success or failure in forming a set.
- Implementation: Renders text messages at the top center of the game window for a brief period, enhancing user interaction and feedback.
- Complexity: Involves fixed-time operations for displaying and removing messages, thus maintaining a O(1)O(1)O(1) complexity.

2.5 IMPLEMENTATION AND CHOICE REASONS

2.5.1 Implementation Details

The key to the game implementation is to combine algorithms and user interfaces to ensure that the game runs smoothly in a real-time environment.

Algorithm realization

- Deck generation and shuffling: 81 cards containing all characteristics are created by the generate_deck() method and distributed using a random shuffling algorithm to ensure that the order of cards is randomized for each game.
- Valid set validation: Three decks are tested for compliance with SET rules via the is_set() method. These rules include that each characteristic is either all the same or all different.
- Finding all SET combinations: The find_all_sets() method iterates through the currently displayed deck using itertools' combination function to find all possible SET combinations.

User interface

- The game interface is implemented in Pygame and includes card drawing, selection effects, and scoreboard display.
- The cards are stylized using transparent fills, shadow effects, and a variety of shapes (ellipses, triangles, diamonds) to enhance visual appeal.
- Time limits and score calculation features add to the challenge of the game.

Game logic

- The player selects cards by mouse clicking, and the program automatically detects whether the selected cards form a valid SET.
- When the player finds a SET, the deck is removed and replaced with a new one, and the game board is dynamically updated.
- Game over conditions include running out of cards or no available SETs.

2.5.2 Code Implementation

The main parts of the code implementation include:

- Card generation and validation: def generate_deck(): ; def is_set(card1, card2, card3):
- Interface Drawing: def draw_card(card, x, y):
- Game Logic

2.5.3 Choice Reasons

- Pygame Options
  Pygame provides powerful graphical drawing and event handling features that are well suited for implementing interaction-based games like SET.
- Algorithm Optimization
  Using itertools' combo function reduces the complexity of traversing all card combinations, while improving efficiency by only checking for newly added card combinations each time.
- Interface Design
  Enhanced the visual effects by adding transparency and dynamic shadows to attract the player's attention.

2.6 RESULT

By testing the game code, we achieved the following goals:

- The game runs smoothly and players are able to select cards and verify SET in real time.
- The score and time mechanisms work effectively and increase the challenge of the game.
- The user interface is intuitive and clear and meets the design requirements of a colorful card game.

3. DISCUSSION

3.1 KEY POINTS AND SHORTCOMINGS

3.1.1 Key Benefits

- Algorithmically efficient:
  The use of itertools combinations dramatically simplifies the SET validation process.
  Flexible card replacement mechanism ensures smooth gameplay.
- Visually appealing:
  The shapes and colors of the cards are intuitively designed to help players quickly identify features.
  Shadows and transparent fills add visual hierarchy.
- Interactive:
  The real-time feedback mechanism allows players to quickly understand the results of operations.

3.1.2 Weaknesses

- Scalability:
- The current version only supports basic SET play and lacks additional levels or multiplayer modes.
- Interactive settings:
- Does not realize the good interactive experience, the page is slightly simple

3.2 IMPROVEMENT AREAS AND EXTENSIONS

- Gameplay Expansion:
  Add a multiplayer mode that allows players to compete or cooperate to complete SETs.
  Optimized code for an optimized player experience
- Hints and learning mechanism:
  Add hint function and animation before starting to help players get started quickly.
  Provide statistics to help players analyze their performance and improve their abilities.

3.3 SUMMARY

This version of the SET game implements the basic functionality, combining efficient algorithms with an aesthetically pleasing interface design. Although there is still room for improvement in terms of performance and scalability, its current implementation is able to provide a satisfactory gaming experience, providing a good foundation for further optimization and expansion.

4. CONCLUSION

In this project, we successfully digitized the SET game, combining efficient algorithms and visual optimization to enable players to experience a digitized version of the classic tabletop game. By introducing an intuitive interface and diverse functionalities, the game not only fulfills the need for entertainment, but also demonstrates its potential in education and cognitive research.

4.1 Main results

- The game is fully functional:
  the core gameplay is ensured by implementing card generation, SET verification, real-time interaction and dynamic updates.
  Adding score and time limit mechanisms increases the competitiveness and challenge of the game.
- Balance of technology and design:
  The game code has good maintainability, and the algorithm design is efficient and suitable for real-time interaction.
  The interface design combines aesthetics and functionality to enhance the user experience.

4.2 Applications and Implications

- Educational value:
  The SET game has significant educational value by developing players' pattern recognition and quick decision-making skills, especially in the training of math and logical thinking.
- Research value:
  The game can be used for research in the fields of cognitive science and psychology to help explore human visual processing and decision-making mechanisms.

4.3 Future outlook

Although the current implementation is already relatively complete, there is still a lot of potential to be tapped:

- Add multiplayer features to enhance social interaction.
- Enriching game modes, e.g. introducing leveling and quest systems.
- Deepen the application in cognitive science research, e.g. recording and analyzing players' reaction time and error rate.

In conclusion, this project demonstrates the potential of SET games for a wide range of applications by integrating technology, design and cognitive science. Not only does it provide an interesting form of entertainment, but it also opens new possibilities for education and research.

5. TASK DIVISION

Our strategy for dividing the task was: write the Python code when we meet up (during tutorials or outside of school) and write the report when we're at home. Message each other when we don't understand something or need help.
Our approach to dividing the tasks in this project was structured and collaborative, ensuring efficiency and effective communication throughout the process. The strategy and timeline for the tasks were as follows:

Janruary18 – Janruary20: Initial Planning and Game Setup

- Janruary18 (Chu & Tongxin)
  - Kick-off meeting to discuss project objectives and detailed task breakdown.
  - Install necessary Python libraries (pygame).
  - Set up a GitHub repository for code version control and collaboration.
- Janruary19
  - Tongxin: Design the Card class and initialize attributes (number, symbol, color, shading).
  - Chu: Set up the Pygame window and define global constants and colors.
- Janruary20
  - Implement the deck shuffling logic using the random.shuffle method.Develop functions to check for sets and find all possible sets among cards.

Janruary21 - Janruary23: Core Development and Function Testing

- Janruary21 (Tongxin)
  - Implement the draw_card and draw_symbols functions to render cards on the screen.
  - Test rendering accuracy and performance.
- Janruary22 (Chu & Tongxin)
  - Code the main game loop, including event handling and game state updates.

- Janruary23 (Chu & Tongxin)
    - Intermediate review meeting to assess progress and integration.
    - Debug and refine interaction and rendering issues identified during testing.

Janruary24 – Janruary26: Advanced Features and User Interface Enhancement

- Janruary24(Chu & Tongxin)
    - Chu: Implement the scoring system and time-based game challenges.
    - Tongxin: Enhance visual feedback for selected cards and successful sets.
- Janruary25 (Tongxin)
    - Develop and integrate the difficulty selection screen using display_time_menu.
- Janruary26 (Chu& Tongxin)
    - Finalize all user interface elements and ensure responsive controls.
    - Test complete game flow from start to finish, focusing on user experience and interface aesthetics.

Janruary26 - Janruary28: Optimization and Documentation

- Janruary26 (Chu & Tongxin)
    - Review and refactor code to improve efficiency and readability.
    - Ensure all functions are optimally performing and properly integrated.
- Janruary27 (Chu)
    - Add comprehensive comments and documentation to the code for better understandability.
    - Conduct thorough testing on different systems to ensure compatibility and stability.
- Janruary28 (Tongxin)
    - Final preparations for project submission.
    - Prepare a brief presentation or demo video outlining key features and gameplay.

Janruary29 – Janruary31: Final Review and Project Submission (Chu & Tongxin)

- Janruary29
    - Review the entire project to ensure coherence and alignment with initial project goals.
    - Prepare the final package of the game, including executable files and source code.
- Janruary30
    - Final internal testing session to catch any last-minute issues.
    - Conduct a detailed review of the documentation and comments.
- Janruary31
    - Final meeting to ensure all components are complete and functioning as expected.
    - Submit the final project documentation and code to the repository and prepare any required submission forms.

This structured schedule ensures that all aspects of the SET game development are addressed methodically, with clear responsibilities and checkpoints for effective collaboration and successful project completion.

6. MANUAL

6.1 OPERATION MANUAL

Here is a detailed, step-by-step user manual for running and engaging with the SET game implemented in Pygame, which is designed for playing a card-matching game based on visual attributes:

6.1.1 System Requirements

- Operating System: Windows, macOS, or Linux
- Python Version: Python 3.x or higher
- Additional Requirements: Administrative access might be necessary for some installation steps.

6.1.2 Installation Steps

1. Install Python: Download and install Python from Python's official website. Ensure Python is added to your system's PATH during installation.
2. Download the SET Game Code: Obtain the SET game script, which is typically available from a repository or provided directly by the developer.
3. Install Necessary Python Libraries:
   - Open a command line interface (terminal for macOS/Linux, command prompt for Windows).
   - Execute the following command to install Pygame: "pip install pygame"

6.1.3 Running the Game

1. Open the Project Directory: Using the command line, navigate to the directory where the SET game code is stored: "cd path_to_SET_code_directory"
2. Execute the Script: Run the game script using Python: "python set_game.py" .This script initializes the game and starts the Pygame window.

6.1.4 Understanding the Gameplay

1. Game Interface:
   - The game window will display a grid of cards each showing different symbols, colors, and shadings.
   - Players can click on cards to select them. The goal is to find sets of three cards where each attribute (number, symbol, color, shading) either all match or all differ.
2. Game Controls:

    - Mouse Click: Select or deselect a card.
    - Timer: A timer is displayed showing how much time is left to find a set.

6.1.5 Additional Features

- Difficulty Settings: Before the game starts, players can choose the difficulty level which adjusts the time available to play.
- Scoreboard: The game keeps track of how many sets the player has found.
- Messages: Feedback messages appear to indicate if the selection is a set or not, and when the game time is up.

This manual should provide all necessary information to install, run, and play the SET game effectively. Enjoy finding as many sets as you can within the time limit!

6.2 GITHUB

Here is the link to our GitHub repository, which includes a manual on the home page (README). Note that since we did almost everything together on one computer (during the tutorials and by meeting up), the history of commits is not representative of who did what. For this, refer to chapter 5.