Solitaire

Session 2023 – 2027

Submitted by:

ESA IRFAN 2023-CS-104

Supervised by:

Mr. Nazeef ul Haq

Mr. Waseem

Course:

CSC-200 Data Structures and Algorithms

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Contents

1.1 Overview 4
1.2 Game Features 4
1.2.1 Classic Solitaire Gameplay 4
1.2.2 Tableau Columns 4
1.2.3 Foundation Piles 4
1.2.4 Stock and Waste Piles 4
1.2.4.1 Stock Pile 4
1.2.4.2 Waste Pile 5
1.2.5 Card Movement Logic 5
1.2.5.1 Tableau to Tableau 5
1.2.5.2 Tableau to Foundation 5
1.2.5.3 Foundation to Tableau 5
1.2.5.4 Waste to Tableau/Foundation 5
1.2.6 Revealing Hidden Cards 5
1.2.7 Game Progression and Win Condition 5
1.2.8 Score and Time System 5
1.2.9 Moves Counter 6
1.2.10 Ace Controller 6
1.2.11 Background Music 6
1.2.11 Extra Missions 6
1.2.12 Red Sequence Card Tracking 6
1.2.13 Graphical Interface with Pygame: 7
1.3 Game Rules 7
1.4 Game Design and Architecture 8
1.4.1 Programming Language 8
1.4.2 Game Structure 8
1.4.3 Data Structures 8
1.4.4 User Interface 8
1.5 Mission Features 9
1.6 Code Implementation 9
1.6.1 Key Classes and Methods 9

1.6.1.1 Card class 9
1.6.1.2 Deck Class 9
1.6.1.3 Foundation Class 9
1.6.1.4 Mission Class 10
1.6.1.5 Tableau Class 10
1.6.1.6 Game Control Class 10
1.6.1.7 Stock and Waste Class 10
1.7 Wireframes 11
1.7.1 Starting of Game 11
1.7.2 Mission Menu 11
1.7.3 During Game Frame 12
1.7.1 Pausing Menu 12
1.8 Conclusion 13
1.9 References 13

Figure 1 Startup Page 11
Figure 2 Mission Menu 11
Figure 3 During Game Frame 12
Figure 4 Pausing Menu 12

1.1 Overview

The **Solitaire Game** project is a digital implementation of the classic card game "Solitaire" using Python and the **Pygame** library. The objective of the game is to move all cards from the tableau columns to the foundation piles by following a set of specific rules.

The game is designed for a single-player experience, where the player is tasked with arranging cards in descending order and alternating suits in the tableau, and placing them in ascending order in the foundation piles (Ace to King) by suit. The player can also interact with the stock and waste piles to draw new cards, which can then be moved into the tableau or foundation piles.

1.2 Game Features

1.2.1 Classic Solitaire Gameplay

The game follows the traditional rules of Solitaire, where the primary objective is to move all the cards from the tableau to the foundation piles. Players need to arrange cards in descending order, alternating between red and black suits in the tableau. In the foundation piles, cards need to be arranged in ascending order (Ace to King) by suit.

1.2.2 Tableau Columns

The tableau is made up of seven columns, where cards are arranged in a specific layout at the beginning of the game. Cards can be moved between columns following the Solitaire rules:

- Cards in the tableau must alternate in color (red or black) and be in descending order.
- Only the topmost card in each column can be moved to another column or the foundation piles.
- When a card is moved, if there are any remaining hidden cards in the column, they are revealed.

1.2.3 Foundation Piles

The foundation piles are where cards are ultimately moved in ascending order from Ace to King, each pile representing a suit (hearts, diamonds, clubs, spades). The goal is to complete each foundation pile by placing the cards in the correct order:

- Ace → 2 → 3 → 4 → ... → King.
- The game ends once all four foundation piles are completed.

1.2.4 Stock and Waste Piles

1.2.4.1 Stock Pile

The stock pile contains cards that are not yet in use. Players can draw cards from the stock pile to add to the tableau or foundation. After drawing from the stock, the card moves to the waste pile

1.2.4.2 Waste Pile

Once a card is drawn from the stock pile, it moves to the waste pile. Players can use the top card of the waste pile in the tableau or foundation piles.

1.2.5 Card Movement Logic

1.2.5.1 Tableau to Tableau

Cards can be moved from one tableau column to another, as long as the move follows the rule of alternating colors and being in descending order.

1.2.5.2 Tableau to Foundation

Cards can be moved from the tableau columns to the foundation piles if they fit in the correct order (Ace to King).

1.2.5.3 Foundation to Tableau

Cards can be moved from foundation to tableau to decrease the complexity or making the move to continue the game

1.2.5.4 Waste to Tableau/Foundation

Cards from the waste pile can be placed in the tableau or foundation, depending on the rules.

1.2.6 Revealing Hidden Cards

When a card is moved from the tableau and a previously hidden card is revealed, the game automatically uncovers the next card in the column. This adds an extra layer of strategy, as players must plan moves carefully to uncover hidden cards.

1.2.7 Game Progression and Win Condition

The game ends when all cards have been successfully moved to the foundation piles in the correct order. Players are rewarded based on their score, number of moves, and whether they completed any missions.

---------- Additional Features ----------

1.2.8 Score and Time System

The game includes a scoring system that rewards players for making valid moves:

- Moving a card to the foundation pile adds a higher score (e.g., +7 points).

- Moving a card to the tableau adds a smaller score (e.g., +3 points).
- Players are encouraged to complete foundations as efficiently as possible for the highest score.
- On completing the mission there will be the addition of +100 points

It also includes timer that notes the time of playing game.

1.2.9 Moves Counter

Each successful move is counted, and players are shown how many moves they've made throughout the game. This adds a challenge to the game, as players try to complete it in the fewest moves possible.

1.2.10 Ace Controller

The Ace controller tracks the movement of Ace cards, which are crucial in completing foundation piles. Moving all Aces to the foundation piles within the first few moves is part of one of the extra challenges.

1.2.11 Background Music

For User-friendly and attractiveness there is addition of sound which create suspense during playing.

1.2.11 Extra Missions

The game includes a set of missions that players can complete while playing, adding extra challenges to the game:

- **Mission 1:** Score 200 points.
- **Mission 2:** Move all four Aces to the foundation within the first 20 moves.
- **Mission 3:** Move 5 cards from the foundation (valid moves).
- **Mission 4:** Remove 10 cards from the stock (valid moves).
- **Mission 5:** Successful 4 move of foundation in sequence.
- **Mission 6:** Empty stock and waste piles.
- **Mission 7:** Move 3 red to the foundation in the sequence.

These missions add variety to the gameplay and encourage players to try different strategies to complete them.

1.2.12 Red Sequence Card Tracking

If a red card (either hearts or diamonds) is successfully moved to the foundation, the game tracks the number of red sequence cards moved. This feature adds another layer of complexity for players who enjoy completing specific sequences within the game.

1.2.13 Graphical Interface with Pygame:

- The game is visually rich, with a clean, user-friendly interface built using the Pygame library.
- The cards are drawn and animated, allowing players to visually interact with the game by dragging and dropping cards between the tableau, foundation, stock, and waste piles.
- Cards have distinct colors and designs, making it easy for players to identify their suits and ranks.
- Background is interactive and attractive

1.3 Game Rules

1. **Objective:** Move all cards to the foundation piles in ascending order (Ace to King) by suit.
2. **Tableau:**
    - Cards in the tableau must be arranged in descending order and alternate between red and black suits.
    - Only the top card of each tableau column is visible and can be moved.
    - A King can be placed in an empty tableau column.
3. **Foundation:**
    - Cards must be placed in ascending order (Ace → 2 → 3 → … → King) and by suit (hearts, diamonds, clubs, spades).
    - Only Aces can start the foundation piles.
4. **Stock and Waste Piles:**
    - Cards are drawn from the stock pile and placed in the waste pile.
    - The top card of the waste pile can be moved to the tableau or foundation.
    - If the stock pile is exhausted, the waste pile is reshuffled to become the new stock pile.
5. **Card Movement:**
    - Cards can be moved between tableau columns if they follow the descending order and alternate colors rule.
    - Cards can be moved to the foundation if they follow the ascending order rule.
6. **Winning Condition:** The game is won when all cards are placed in the foundation piles in the correct order.
7. **Score System:**
    - In tableau, successful move cause +3 score.
    - In Foundation, successful move cause +7 score.
    - Mission completion cause +100 score
8. **Moves Counter:** Increased on every successful move from anywhere to anywhere.
9. **Pause:** On pressing escape key the game is paused.

1.4 Game Design and Architecture

This section outlines the technical aspects of the Solitaire game, including the programming language, libraries used, game structure, data handling, and user interface design.

1.4.1 Programming Language

- **Python**: The game is developed using Python, which provides an easy-to-use framework for game development.
- **Pygame**: Pygame is used to handle graphics, user input, and the overall game loop.

1.4.2 Game Structure

- **Card Class**: Manages individual card attributes like suit, rank, and face-up/face-down state.
- **Tableau Class**: Controls the tableau columns, validating moves and maintaining card stacks.
- **Foundation Class**: Handles foundation piles and validates card placements.
- **Stock and Waste Class**: Manages the stock and waste piles, allowing cards to be drawn.
- **Mission Class**: Tracks the player's progress with various missions throughout the game.
- **Deck Class**: Create Deck and shuffle it for game.
- **Game Controller Class:** Mange Pause and exit system of game.
- **Solitaire.py:** Mange all frontend and connect backend system with frontend

1.4.3 Data Structures

- **Lists**: Used as stacks and queue.
- **Stack:** Used for storing, popping and pushing in foundation and tableau
- **Queue:** Used for popping from stock and waste piles.
- **Dictionaries**: Store card positions and facilitate quick access to various components and for storing missions.
- **Classes**: Used to represent different game components such as cards, tableau, foundations, etc.
- **2D-List:** Implementation of tableau.

1.4.4 User Interface

- **Game Window**: Displays the game environment, including tableau columns, foundations, stock, and waste pile.
- **Game Background**: Use custom images for this.
- **Card Graphics**: Custom images for cards, with face-up and face-down states.
- **Buttons and Labels**: Interactive elements for starting the game, accessing missions, and displaying score and moves.

1.5 Mission Features

The **Missions** class in your Solitaire game adds additional objectives for the player to complete during gameplay. These missions are not only challenges but also help guide the player in progressing through the game while providing extra goals to achieve beyond the regular Solitaire rules. Following are the missions of my solitaire game

1. **Get the score of 200**: This mission requires the player to achieve a score of 200 points.
2. **Move all four aces to the foundation within the first 30 moves**: The player must move all four aces to the foundation piles before completing 30 moves.
3. **Move 3 cards from the foundation (valid moves)**: The player needs to make three valid card movements from the foundation piles.
4. **Remove 10 cards from the stock (valid moves)**: The player must draw and remove 10 cards from the stock pile in valid moves.
5. **Successfully move 4 sequences of cards to the foundation**: The player needs to move four valid sequences of cards (following the rules) to the foundation piles.
6. **Empty the stock and waste piles**: This mission requires the player to successfully empty both the stock and waste piles.
7. **Move 3 red cards to the foundation in a sequence**: The player must move three red cards in a specific sequence to the foundation piles.

For each completed mission, the player is rewarded with points (e.g., 100 points). This is part of the overall progression and helps the player track their achievements during the game.

1.6 Code Implementation

1.6.1 Key Classes and Methods

1.6.1.1 Card class

The Card class represents an individual playing card, defined by its suit (e.g., hearts, diamonds) and rank (e.g., Ace, 2, 3, ... King). Each card also has a color (red or black) based on its suit, and an image file loaded and resized to fit the game's card display size. Cards can be face-up or face-down, controlled by the revealed attribute, which is set to True by the reveal method. The class also provides a way to load the image for each card from a file and represents each card as a string showing its rank and suit (e.g., "A of hearts")

1.6.1.2 Deck Class

The Deck class manages a full deck of 52 cards, shuffling them at initialization to create a randomized order. It includes a deal method to draw the top card from the deck, returning None if no cards remain.

1.6.1.3 Foundation Class

The Foundation class represents one of the four foundation piles, each dedicated to a specific suit. It includes methods for adding cards in ascending order by rank, starting with Ace, and only

accepts cards matching its designated suit. The is_complete method checks if the foundation contains all 13 cards (from Ace to King), indicating a completed pile.

1.6.1.4 Mission Class

The Missions class adds a set of mini-challenges, or missions, to the game interface. It contains attributes and methods to display these missions to the player and check their completion during gameplay. When a mission is completed, it is removed from the list, rewarding the player with 100 points.

Each mission has a specific function that checks the game's current state, such as score, number of aces moved, or stock pile usage. If a mission's conditions are met, it is marked as complete and removed. The player can view their mission list by clicking the "Missions" button, which opens a separate display with the current objectives and an option to resume the game by pressing any key.

1.6.1.5 Tableau Class

The Tableau class manages the tableau columns in a Solitaire game. It initializes seven columns, sets up the tableau by dealing cards into each column with only the bottom card revealed, and provides functionality to check if a card can be moved to another column based on the Solitaire rules. The class also includes methods for rendering cards on the screen, either showing the face or back of the card depending on whether it's revealed. It allows access to and modification of specific columns, enabling players to interact with the tableau during the game.

1.6.1.6 Game Control Class

The PauseMenu class in this code implements a pause menu for a game using Pygame. It handles pausing the game, showing options to continue or exit. The class has an is_paused flag to track the pause state. When paused, a semi-transparent surface is drawn on the screen with two buttons ("Continue" and "Exit"). The handle_events method listens for mouse clicks on these buttons and performs the respective actions—either unpausing the game or quitting the application. The draw method renders the pause menu surface and the buttons on the screen when the game is paused.

1.6.1.7 Stock and Waste Class

The StockAndWaste class in the Solitaire game manages the stock and waste piles. The stock pile holds the remaining cards yet to be used, while the waste pile stores cards that have been drawn. The class includes functionality to render the piles on the screen, displaying the back of the cards for the stock pile and the top card of the waste pile. It also handles the interaction when a player clicks on the stock pile, drawing a card to the waste pile, and when the stock is empty, it resets by moving the waste pile back into the stock. This setup is essential for the game's card cycling mechanics.

1.7 Wireframes

1.7.1 Starting of Game

Figure 1 Startup Page

1.7.2 Mission Menu

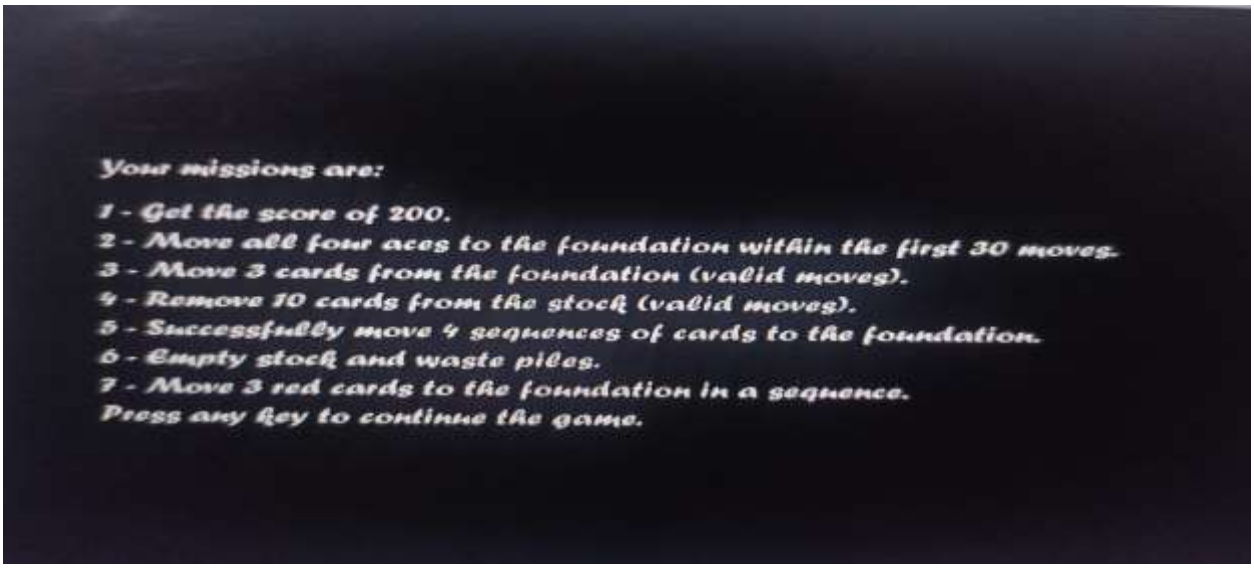

Figure 2 Mission Menu

1.7.3 During Game Frame

Figure 3 During Game Frame

1.7.1 Pausing Menu

Figure 4 Pausing Menu

1.8 Conclusion

The Solitaire game project is a fantastic example of how Data Structures and Algorithms (DSA) can be used to create a fun and engaging game. By employing stacks, queues, lists, and arrays, the project efficiently manages the cards and their movements. It's like using a toolbox of special tools to build a game. These tools help the computer keep track of the cards, move them around, and ensure that the game follows the rules. This project showcases how DSA can be applied to create a wide range of interactive and fun games.

1.9 References

1. "Pygame Documentation" - Pygame, https://www.pygame.org/docs
2. "The Official Rules of Solitaire" - World of Solitaire, https://www.worldofsolitaire.com/
3. "Python Game Development" - Python.org, https://www.python.org/doc/essays/
4. “Data Structures implementation” - GeeksForGeeks. https://www.geeksforgeeks.org/courses/Data-Structures-With-Python