GenAI-Report

Divija Arora

March 2024

1 Objective

Diamonds is a trick-taking card game akin to traditional favorites like Hearts and Spades, conceived by Zach Barth and distributed by Stronghold Games. Suitable for 3 to 6 players, the game usually lasts around 45 minutes. In Diamonds, the primary goal is to accumulate points through the acquisition of diamond cards stored in your vault and by prevailing in tricks featuring diamond cards.

2 Setup and Gameplay

- Setup :
  - Every participant is dealt a standard 52-card deck, with the diamonds suit omitted. As a result, each player is assigned a distinct suit comprising hearts, clubs, or spades.
  - The diamond cards are segregated and thoroughly shuffled, earmarked for employment during the auction phase.
- Gameplay:
  - Auction Phase-
    - The top diamond card is revealed from the shuffled diamond deck.
    - Players take turns bidding on the diamond card using one of their own cards from their hand, placed face down. The bid is a single card, and its rank determines the bid value.
    - The highest bid, determined by the card with the highest rank, secures the diamond card.
    - If multiple players bid the same highest rank card, the points from the diamond card are divided equally among them.
    - The winning bidder takes the diamond card and places it in front of them to keep track of their points.
  - Scoring-

    - The points from the won diamond cards are recorded in each player's column on the scorekeeping table.
    - The ranks of the cards correspond to their point values: 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < T < J < Q < K < A .
    - At the end of the game, the player with the most points wins.

- Additional Rules:
  - Certain variations of the game may introduce rules specifying when players can bid particular cards, such as limitations on bidding cards from the same suit as the diamond card being auctioned. .
  - There may also be variations on how ties are resolved, such as splitting the points evenly or using a tiebreaker mechanism.

3 Reflections on conversation with genAI and prompting

If you were to write a code for this game, what are the essential functions you would need to consider? Could you assist with identifying them?

Consider including the following functions in your code:

1. setupGame(players)
2. conductAuction()
3. computePoints(cardValue)
4. adjustScores(player, pointsEarned)
5. checkEndGameCondition()
6. displayGameScores()
7. mainGameLoop()
8. announceGameWinner()

4 Code that was generated by LLM and later improved

```python
import random

def setupGame(players):
    # Define the non-diamond suits
    suits = ['Hearts', 'Clubs', 'Spades']
    # Initialize an empty dictionary to store player hands
    player_hands = {}
    # Deal cards to each player from the non-diamond suits
    for player in range(1, players + 1):
        # Initialize the player's hand
        player_hand = []
        # Deal 10 cards to each player
        for _ in range(10):
            # Randomly select a suit and rank for the card
            suit = random.choice(suits)
            rank = random.randint(2, 14) # 2 to Ace (14)
            # Add the card to the player's hand
            player_hand.append((rank, suit))
        # Sort the player's hand for easier display
        player_hand.sort()
        # Add the player's hand to the dictionary
        player_hands[player] = player_hand
```

```python
    # Prepare the diamond deck for the auction phase
    diamond_deck = [(rank, 'Diamonds') for rank in range(2, 15)] # 2 to Ace
    random.shuffle(diamond_deck) # Shuffle the diamond deck
    # Return the player hands and the shuffled diamond deck
    return player_hands, diamond_deck
```

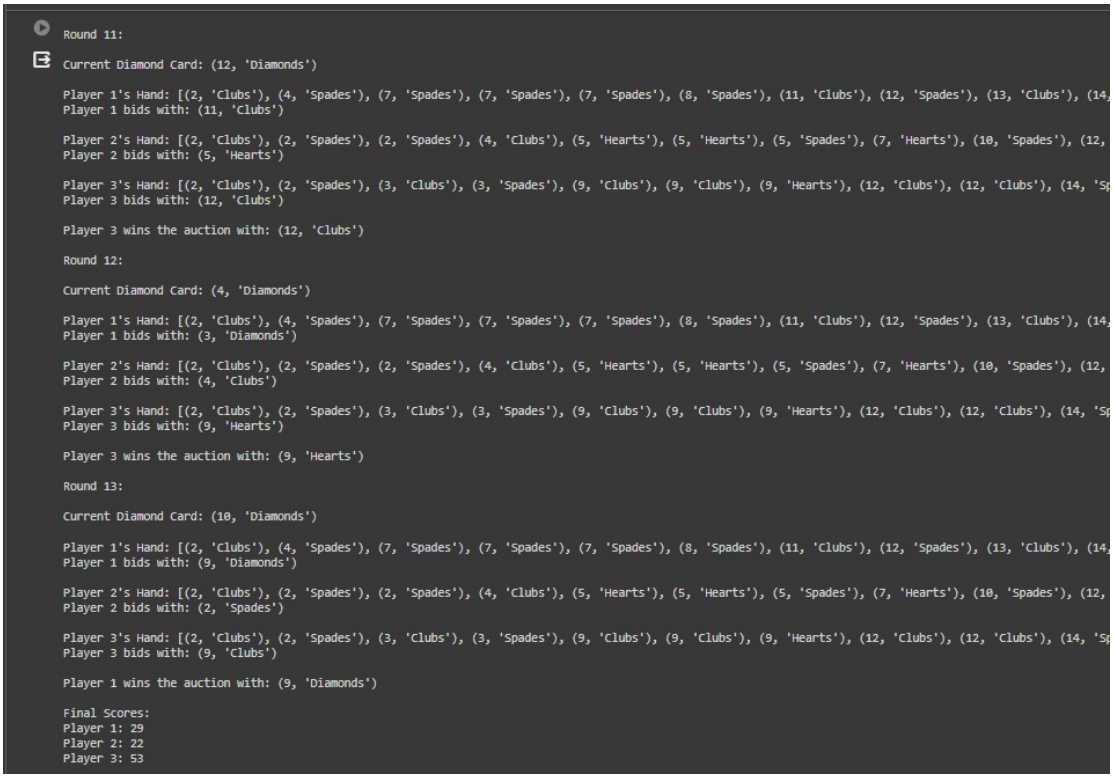
```
def conductAuction(player_hands, diamond_deck):
    # Retrieve the top card of the diamond deck
    top_diamond_card = diamond_deck.pop(0)
    print("\nCurrent Diamond Card:", top_diamond_card)
    # Dictionary to store bids from each player
    bids = {}
    # Accept bids from each player
    for player, hand in player_hands.items():
        print(f"\nPlayer {player}'s Hand:", hand)
        bid_card = random.choice(hand) # For demonstration purposes, randomly
            selected
        print(f"Player {player} bids with:", bid_card)
        bids[player] = bid_card
    # Determine the winning bid
    winning_player = max(bids, key=lambda x: bids[x][0]) # Player with the
        highest-rank card
```

```
def computePoints(cardValue):

    '''Calculate the points earned from a won diamond card based on its value.

    Args:
    - card_value (int): The numerical value of the diamond card (2 to 14).

    Returns:
    - int: The number of points earned from the diamond card.
    '''
    # Diamond cards are worth their numerical value
    return cardValue
```

```
def adjustScores(scores, player, points):
    '''Update the score of a player by adding the specified number of points.
    Args:
    - scores (dict): Dictionary storing the scores of each player.
    -
    · player (int): The ID of the player whose score needs to be updated.
    - points (int): The number of points to be added to the player's score.
    Returns:
    - dict: Updated scores dictionary.
    '''
    # Check if the player exists in the scores dictionary
    if player in scores:
        # Update the player's score
        scores [player] += points

    else:
        # If the player doesn't exist, create a new entry
        scores [player] = points

    # Return the updated scores dictionary
    return scores
```

```
def checkEndGameCondiion(diamond_deck, rounds_played, max_rounds):

    # Check if all diamond cards have been auctioned
    if not diamond_deck:
        return True

    # Check if the maximum number of rounds has been reached
    if rounds played >= max_rounds:
        return True

    # If neither condition is met, continue the game
    return False
```

```
def mainGameLoop(num_players, max_rounds):

    # Initialize the game
    player_hands, diamond_deck = setupGame (num_players)
    scores = {player: 0 for player in range(1, num_players + 1)}
    rounds_played = 0

    # Main game loop
    while not checkEndGameCondiion (diamond_deck, rounds_played, max_rounds):
        print (f"\nRound {rounds_played + 1}:")
        # Execute one round of the auction phase
        winning_player, top_diamond_card = conductAuction (player_hands,
            diamond_deck)
        # Update scores based on the winning bid
        points_earned = computePoints (top_diamond_card[0]) # Assuming card
            value is 0
        scores = adjustScores (scores, winning_player, points_earned)
        # Increment the number of rounds played
        rounds_played += 1

    # Display final scores
    print("\nFinal Scores: ")
    for player, score in scores.items():
        print("Player {player}: {score}")
```



5 Iterating Upon Strategy

- Evaluate Card Value: Assess the value of cards in hand and bid accordingly, prioritizing higher-ranking cards for more valuable Diamond cards.
- Manage Risk: Balance between bidding aggressively to win valuable Diamond cards and conservatively to avoid over-committing and losing valuable cards.

I evaluated the strategy against a computer opponent across multiple rounds, noting satisfactory outcomes. GenAI exhibited competitive gameplay and gradually adjusted to diverse game scenarios. While each iteration reduced ambiguity, precision wasn't consistently achieved.

6 Analysis and Conclusion

In summary, employing prompts to engage with the AI (Gen-AI) offered a smooth and effective method for improving code generation. This collaborative approach enabled me to polish code, assess its performance across diverse scenarios, and scrutinize the AI's responses to prompts. Such a methodology facilitated a comprehensive comprehension of programming principles while harnessing the AI's prowess to expedite development endeavors.