Diamonds Game UI

Akshada Kulkarni, Bhavani Chalasani, Priya Mandot

1 Introduction

The game of Diamonds is a two-player strategic card game where players bid over diamond cards to score points. The bidding process involves players selecting a card from their hand to represent their bid, and the player with the highest bid wins the diamond card.

2 Problem Statement

Create a UI for the diamonds game with genAI

3 Teaching the AI

I refreshed the AI with the code we had settled on last time, and asked it to draw a UI with pygame following the strategy of that code.

4 Iterating upon strategy

We had to fix many UI issues, such as size and spacing of the cards, making sure that once a card was clicked it disappeared from the deck, making sure that the accurate card was selected etc.

5 Code

The code we settled on is as below :

```
import pygame
import random

# Initialize pygame
pygame.init()

# Define colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

# Define card values and points
```

```
card_values = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']
points = {'2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9, '10': 10, 'J': 11, 'Q': 12, 'K': 13, 'A': 14}

# Initialize players' decks
player_deck = card_values.copy()
computer_deck = card_values.copy()
diamond_deck = card_values.copy()
random.shuffle(diamond_deck)

# Initialize points for each player
player_score = 0
computer_score = 0
comp_last_choice=0
diamond_last_choice=0

# Shuffle diamond cards
diamond_cards = card_values[0:13]  # Selecting cards 2 through Ace
random.shuffle(diamond_cards)

# Set up the display
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Diamonds Game")

# Load card images and resize them
card_images = {}
for card in card_values:
    original_image = pygame.image.load(f"cards/{card}.jpg")
    scaled_image = pygame.transform.scale(original_image, (original_image.get_width() // 5, original_image.get_height() // 5))  # Adjust scaling factor
    card_images[card] = scaled_image

# Function to display text
def draw_text(text, font, color, x, y):
    text_surface = font.render(text, True, color)
    text_rect = text_surface.get_rect()
    text_rect.midtop = (x, y)
    screen.blit(text_surface, text_rect)

# Function to display cards

def draw_cards(deck, x, y):
    card_width = card_images['2'].get_width()
    spacing = 60  # Spacing between cards

    for i, card in enumerate(deck):
        card_x = x + i * (card_width + spacing)
        card_y = y
        screen.blit(card_images[card], (card_x, card_y))

# Main game loop
running = True
while running:
    screen.fill(WHITE)
```

```
65
66      # Event handling
67      for event in pygame.event.get():
68          if event.type == pygame.QUIT:
69              running = False
70          elif event.type == pygame.MOUSEBUTTONDOWN:
71              # Determine which card the player clicked on
72              mouse_x, mouse_y = pygame.mouse.get_pos()
73              card_width = card_images['2'].get_width()
74              card_height = card_images['2'].get_height()
75              spacing = 60
76              player_choice = "NO"
77              for i, card in enumerate(player_deck):
78                  card_x = 50 + i * (card_width + spacing)
79                  card_y = HEIGHT - 200
80                  if card_x <= mouse_x <= card_x + card_width and
    card_y <= mouse_y <= card_y + card_height:
81                      player_choice = card
82                      print(player_choice)
83                      break
84              if player_choice!="NO":
85              # Determine the computer's choice
86                  computer_choice = random.choice(computer_deck)
87                  comp_last_choice=computer_choice
88
89                  drawn_diamond_card = random.choice(diamond_deck)
90                  diamond_last_choice=drawn_diamond_card
91
92                  # Determine the winner of the round
93                  if points[player_choice] > points[computer_choice]:
94                      player_score += points[drawn_diamond_card]
95                  elif points[player_choice] < points[computer_choice
    ]:
96                      computer_score += points[drawn_diamond_card]
97                  elif points[player_choice] == points[
    computer_choice]:
98                      computer_score += points[drawn_diamond_card]/2
99                      player_score += points[drawn_diamond_card]/2
100
101                 # Remove cards from decks
102                 player_deck.remove(player_choice)
103                 computer_deck.remove(computer_choice)
104                 diamond_deck.remove(drawn_diamond_card)
105
106
107     # Draw player's cards
108     draw_cards(player_deck, 50, HEIGHT - 200)
109
110     # Draw player's score
111     draw_text(f"Player Score: {player_score}", pygame.font.Font(
    None, 36), BLACK, WIDTH // 2, HEIGHT - 400)
112     draw_text(f"Computer Score: {computer_score}", pygame.font.Font
    (None, 36), BLACK, WIDTH // 2, HEIGHT - 450)
113     draw_text(f"Computer's Last Choice: {comp_last_choice}", pygame
    .font.Font(None, 36), BLACK, WIDTH // 2, HEIGHT - 350)
114     draw_text(f"Last Drawn Diamond Card: {diamond_last_choice}",
    pygame.font.Font(None, 36), BLACK, WIDTH // 2, HEIGHT - 320)
```

```
115
116
117     pygame.display.flip()
118
119 # Quit pygame
120 pygame.quit()
```

6 Analysis and Conclusion

It was harder for the AI tool to write code for UI, as compared to the logic from the last assignment. It displayed little knowledge about how the UI would actually look, and couldn't easily consider spacing of the cards etc.

However, with repeated prompting it improved, to correctly display the UI and ensure everything was displayed and selected accurately.

In conclusion, AI was a tool that provided a good skeleton upon which to build the project further, and completed the project correctly after being nudged in the right direction.