Crafting the User Interface Experience for the 'Diamonds' Game with Pygame

Kashish Bhurani, Aashna Dogra, Nyasha Vasoya

May 12, 2024

1 Introduction

The task assigned to us was a continuation of the previous task about the game of diamonds: Using generative AI, Introduce ChatGPT to the game of "Diamonds" and using Pygame build a graphic user interface for the same. Despite its apparent simplicity and after completion of the previous task, we had expected this to be a cake walk. Instead this task proved to be difficult and insightful. As we started with the given task we encountered that even though ChatGPT knew the game, it didn't remember the game and we had to start from the scratch and introduce the game once again. Explaining each and every aspect of the game again, seemed like a tedious task.

2 Methodology

2.1 Approach

Approach 1: Started a new chat and explained it the game with the bullet points it gave before. We worked with the code it had previously given and asked it to write on a pygame script for it. But it continuosly gave code that did not give any output.

Approach 2: Starting the explanation from the very start seemed unnecessary and a tedious task, that is why instead of explaining it the game again, I continued the chat where I had asked ChatGPT to form the logic. I worked with the code it had given me and started with it. The code it gave me at the start did not work at all and it just gave me a blank screen. Even after a couple of times when I continuosly gave it prompts to improve itself, it still kept on giving me bad code.

2.2 Giving Prompts and Learning Pygame

Things that did not work: ChatGPT did a good job writing basic pygame code but as we progressed to more complicated parts of the game it started making mistakes. We asked it to improve the code and explain why it was using the functions it used in it's code. We used the documentation to understand pygame and point out ChatGPT's mistakes but the prompts we gave made it give almost similar answers and no improvement was shown.

Things that worked: Asking ChatGPT the code for the gui using pygame proved more challenging than I had anticipated, To improve its learning, I asked ChatGPT to start teaching me Pygame from the start and explain how to simply form a rectangle and and write a number in it. I then asked ChatGPT to give me functions and started working on it step by step correcting each and every mistake it made. I did not know pygame that well and the one thing that helped me immensely was the documentation. I started from the very basic and then made my way up.

2.3 Prompts Given

Here are some prompts which did not work very well in helping to understand pygame-

- Can you write a code that can implement the game ...
- how to write a pygame code to have three players with different suites ...

- also keep updating scorecard ...
- this code isn't working change your approach ...

Here are the examples of some of the prompts given to ChatGPT that worked.

- Explain me how to use pygame and start with telling me the basics...
- (After giving the code) Try to make a graphic user interface for this code using pygame...
- this does give me the display but it does does nothing else. I want it to let me see the random diamond card that was bid, i also want it to show ...

To refer to the entire conversation click here .

2.4 Code Snippets

```
def draw_card(drawn_cards:list[str])-> str:
    diamonds = ['2', '3', '4', '5', '6', '7',
                '8', '9', '10', 'J', 'Q', 'K', 'A']
    available_cards = [card for card in diamonds if card not in drawn_cards]
    if available_cards:
        card = random.choice(available_cards)
        drawn_cards.append(card)
        return card
    else:
        return None

def make_bot_bid(bot_bids:list[str], drawn_card:str)->str:

    if drawn_card in ['2', '3', '4', '5']:
        bot_bid = random.choice(['2', '3', '4', '5'])
    elif drawn_card in ['6', '7', '8', '9']:
        bot_bid = random.choice(['6', '7', '8', '9'])
    else:
    bot_bid = random.choice(['10', 'J', 'Q', 'K', 'A'])
    if bot_bid in bot_bids:
        return make_bot_bid(bot_bids, drawn_card)
    else:
        bot_bids.append(bot_bid)
        return bot_bid

def display_cards(screen, cards, font, user_bids):
    x = (WINDOW_WIDTH - (CARD_WIDTH + CARD_SPACING) * len(cards)) // 2
    y = WINDOW_HEIGHT - CARD_HEIGHT - CARD_SPACING
    for card in cards:
        color = BLACK
        if card in user_bids:
            color = (255, 0, 0)
        pygame.draw.rect(screen, GRAY, (x, y, CARD_WIDTH, CARD_HEIGHT))
        display_text(screen, card, x + CARD_WIDTH // 2,
                     y + CARD_HEIGHT // 2, font, color)
        x += CARD_WIDTH + CARD_SPACING
```

3 Challenges faced

- ChatGPT forgets the earlier bits of the conversations

- It was changing the logic of the code so we had to provide it the logically correct code again and again.
- It could not understand what was wrong in its code when told to improve it and gave the same thing

3.1 Important Learnings

- We did not know Pygame and we realised the only way we could be able to learn it thoroughly was with the help of documentation.
- Also we learnt some aspects of prompt engineering, how just telling GenAI that it is wrong you need to explain properly how it is wrong and what it should do to improve.
- It is important to refactor the code, point out small mistakes make ChatGPT understand the entire working behind it.

3.2 Reflections and Conclusion

What we learnt was that ChatGPT had immense potential and great learning capacity when prompted the right way. From the initial code ChatGPT gave to its final rendition, it made a lot of progress and continuously improved the code and its quality when its mistakes were pointed out. We also observed that ChatGPT also introduces the problem of ”forgetfulness” and ”hallucinations” and has to be told the same thing multiple times for it to understand it completely. One major observation we made was that generative AI learns best with examples and the minute problems of ChatGPT fixable if constant iteration on the solution is done, to modify and refine it.

To view the entire workable code of the diamond game implemented in pygame click here.