Blackjack Design Proposal

Language and/or Framework of Choice

I will be programming in the language of: Python

I will also use: Tkinter

Project Goal:

By the end of the project I hope to have and accomplish:

- A functional blackjack virtual card game
- Functional player (able to hit and stand)
- Functional AI (also able to hit and stand on its own)
- Interactable GUI
  - Card visuals
  - User Interface visuals
  - Functional Buttons
- Betting system
- Reset function
- Win or loss outcome

Project Explanation:

In this project, I will be making a virtual card game. This card game is called Blackjack, where the player and dealer both have two cards that will be added together to make a sum. The goal is to get as closest to 21 as possible. Anyone may choose to "hit" (draw card) as many times as they want until they are satisfied with the sum or until the total sum of your hand gets over 21, which is called a "bust", and you lose the game. There are other things which make the card game interesting, such as being able to decide whether or not an ace will be 1 or 11. Face cards are also all 10. I have decided to make Blackjack because it is simple to create, yet puts me to the challenge of applying my knowledge. Compared to other card games such as poker, AI needs to be much more advanced and I am personally not as familiar with the rules of poker as opposed to Blackjack. This card game will use an AI, which I am interested in creating. In addition to making AI, I will create a GUI for it to be an actual game in which the player is free to interact with it for the code to run. Mobile card game apps use a similar conjecture, except having mobile controls. Being able to create a card game allows me to explore how many popular mobile card games were made. With my current knowledge on Object-oriented programming, I will be able to apply it to coding the game. I can apply how to organize code, how to create efficient methods for the AI and player, and how to create different attributes for the AI and player. I have also learnt how to reuse code which will help with the program's efficiency. I can explore my knowledge by creating objects such as the cards. I am also able to distinguish when to use functions vs. methods - methods are associated with objects, while functions are not. As for extra knowledge I must gain myself, this project allows me to learn how to connect functions and classes with a GUI.

Feature List

Player - Player class object

- Contains data with list of card class objects
    - Player is able to hit
    - Player is able to stand
    - Player is able to place a bet

Betting System - Player is able to place a bet in the entry TKinter function.

- Player starts with $500 and resets every time the player closes the game
- After inputting a bet, game starts
- If player loses, they lose the amount they have bet
- If player wins, they earn 1.5x the amount they have bet

Card class objects

- Card ranks
- Card suits
- Card facing up/facing down boolean value

AI (dealer) - AI class object

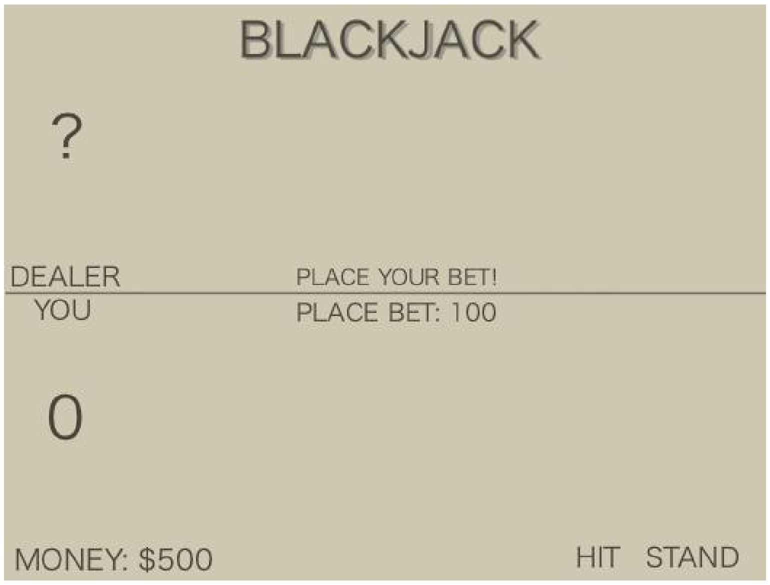
- Contains data with list of card class objects
    - AI is able to hit and stand on its own
        - Must stand if the value is 17 or more
        - Must draw another card if total is 16 or under until total is 17 or more

Interactable GUI

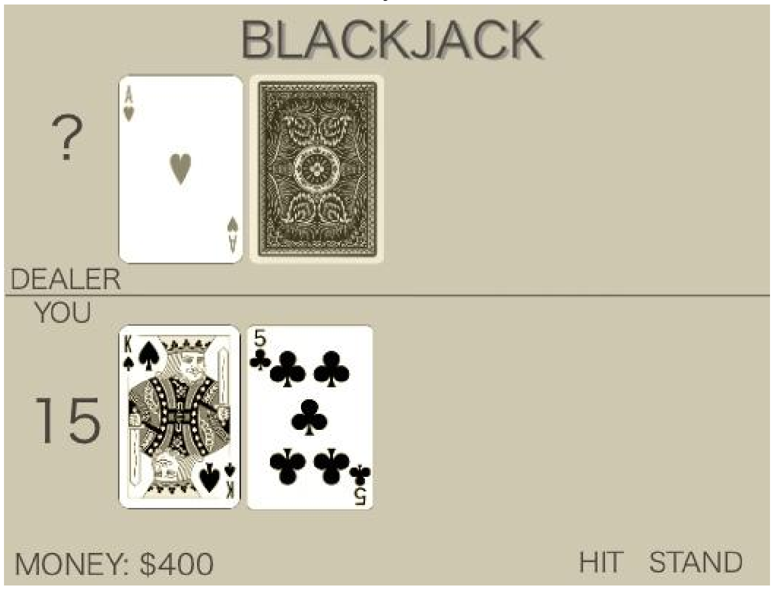
- Buttons
- Text Boxes
- Layout
- Images

<u>prototype: created on Procreate</u>

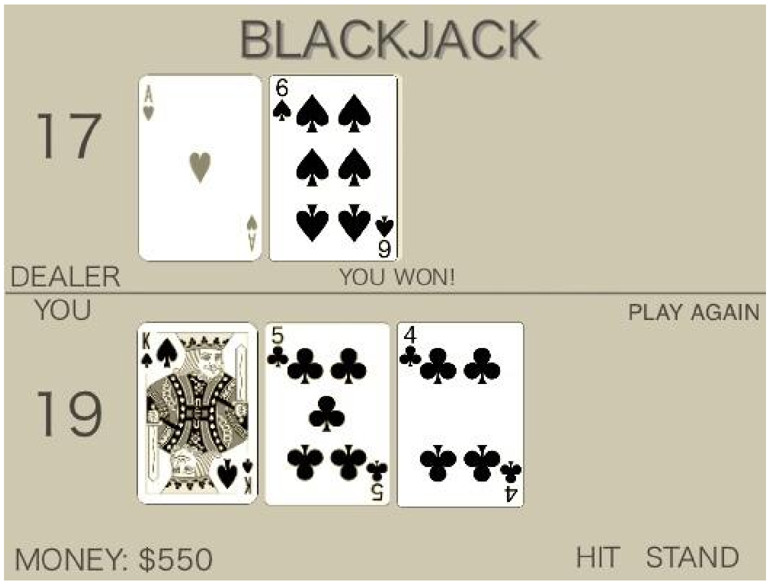
Pregame - player places a bet of their choice ($0-$500). Player always starts with $500. If they do not have enough money to place a bet, the system will display "not enough funds!". If they place a bet out of the range, the system will display "bet out of range ($0-$500)". In this scenario, we placed a bet of $100. When the player has bet their desired amount and hits 'enter', the game will start.



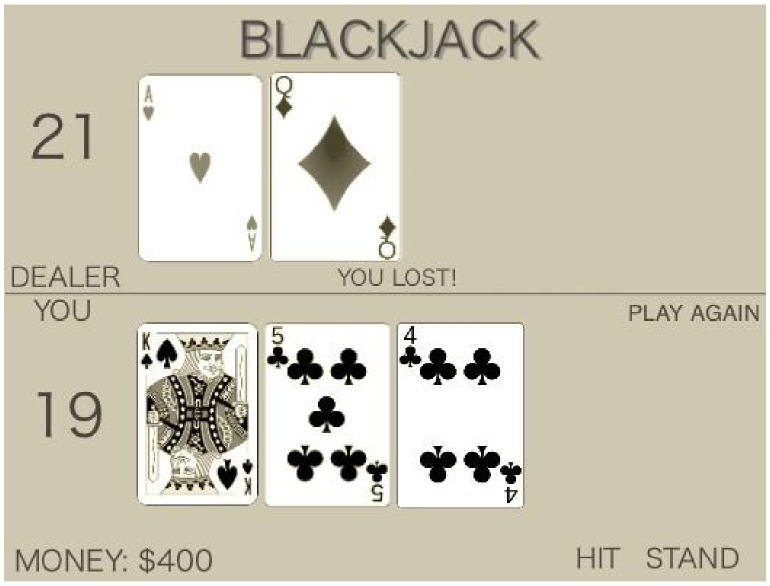
Start of game - Player can only view one of the dealer's cards and may not know their sum. Player starts with two random cards that will make up a sum as displayed on the left. Player has an option to 'hit' (draw card) until they click 'stand' or until the sum goes above 21. Player may also wish to click 'stand' if they are satisfied with their sum.



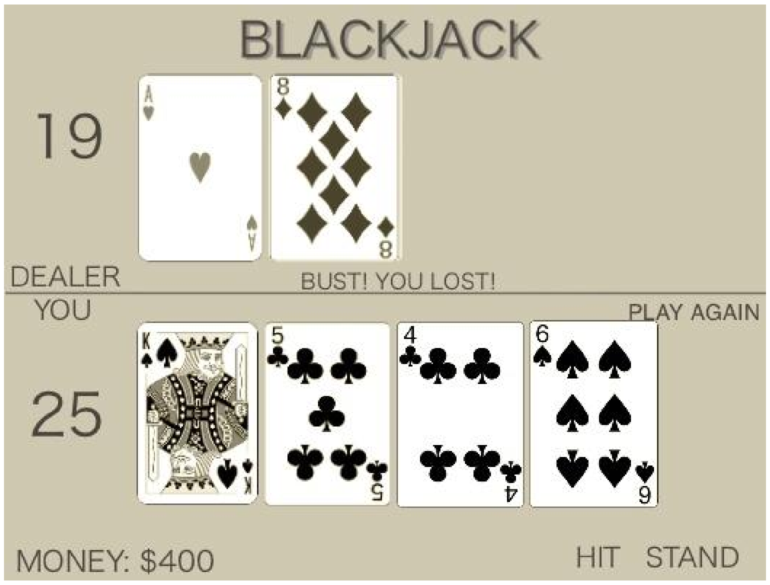
Outcome 1: win - When the player has a higher sum than the dealer after both "stands", there will be a dialogue that shows up in the middle that says "you won!". Dealer's cards are revealed as well as their sum. This outcome also occurs if the dealer "busts" (see outcome 3). The player has the option to play again. The bet that the player placed will receive 1.5x the amount. In this case, we bet $100, so we get $150 back.



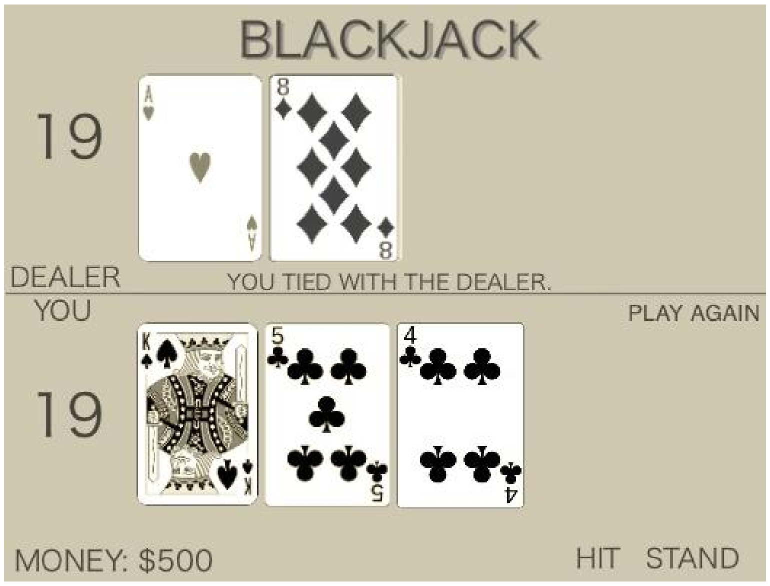
Outcome 2: Loss - When the player's sum is less than the dealer's after both stands, a dialogue appears in the middle stating that the player has lost. This loss outcome also occurs when the dealer reaches 21 and the player has not. The dealer's cards are shown as well as their sum. The player then has the option to play again. The bet that has been placed is kept by the dealer.



Outcome 3: loss #2 (bust) - If the player hits and reaches a sum of over 21, a dialogue automatically appears, saying that the player has hit a "bust". The dealer's cards appear as well as their sum. The player is able to play again and their bet is kept by the dealer.



Outcome 4: Tie - When the player has the same sum as the dealer after both have chosen to "stand", a dialogue appears saying "you tied with the dealer". The bet that the player has placed is returned. The player has the option to play again.


BLACKJACK
19
DEALER
YOU TIED WITH THE DEALER.
YOU
PLAY AGAIN
19
MONEY: $500
HIT
STAND