Black Jack:

A GUI-Based Approach

Department of Computer Science and Engineering

California State University, Fullerton

Project Advisor: Surrajkumar Prabhu Venkatesh

Submitted By: Mohammed Ali & Tonirose Virata

May 10, 2024

Introduction

Blackjack is a world-renowned card game, often played in casinos, where the goal is to have a handful of cards whose sum is or closest to 21. Our project is designed to recreate the game Blackjack with added user interface and a computer AI the player can go against.

Objectives

The objectives for this project include implementing the basic rules of Black Jack, creating an intuitive and user-friendly graphical user interface for player interactions, simulating a deck of cards and dealing cards to both the player and the dealer. Scoring will also be implemented to determine a winner at each round and give users the ability to play again after each game.

Features

This project will showcase a GUI interface, deck and card representation, player interaction, a scoring system, dealer logic, a computer AI, game flow, error handling, graphics and animations, and possibly sound effects. The Blackjack game also includes an intuitive betting system for the player to bet chips in increments of 1, 5, 10, 25, 50, 100, 500, and 1000.

Datasets

The only dataset that will be used for this Black Jack game will include a set of 52 playing cards which will be implemented using a dictionary set. Each card will have a rank and suit that will be stored respectively.

Existing Code and Extensions

There is existing code for the base Black Jack game in other languages. The added features that will be implemented in our project is a graphical user interface, a complex computer AI that the player will go against each game, display messages, scoring, the ability to play the game again, and the ability to play against multiple AI bots.

Algorithms and Approach

The computer AI will also be implemented using the Monte Carlo approach. The Monte Carlo approach is a tree search that will search through a tree populated with possible and random moves to pick the most optimal move the computer can make within the game.

Development Environment

Type	**Software**
Programming Languages	Python
IDE	Visual Studio Code
Libraries	MATLAB, Pygame, Numpy
Operating System	Windows 10/11, macOS Sonoma 14.0

Description of the Software

The Black Jack game implements a GUI through use of Python's Pygame library. The library offers a multitude of ways to create stunning visuals and sound effects. With the help of Pygame, we were able to implement a "casino table" styled background, card faces, poker chips for betting, fonts for stylized text, a betting system with chips, and card shuffling sound effects.

Using the Numpy library, we are also able to create three-dimensional graphs for users to see the rate at which our computer AI is trained with card faces and patterns to eventually win more games when going against the dealer. The AI algorithms used for the Black Jack game is the Monte Carlo approach. The Monte Carlo approach to determine the best move in the game, in this case the computer is determining whether or not it should hit or stand when looking at their hand.

Schedule

Listed below is a comprehensive implementation plan that the group plans on following closely:

1. Choose a GUI library (e.g. Pygame) and set up the basic window structure.
2. Implement visual representations of cards and the game table within the GUI.
3. Develop GUI elements for player actions such as hit, stand, and quit.
4. Implement the logic for dealing cards to the player and dealer visually.
5. Allow the player to choose actions through GUI interactions, updating the visual representation accordingly.
6. Implement the scoring system and determine the winner visually.
7. Implement the dealer's logic for taking turns and update the GUI accordingly.
8. Manage the flow of the game within the GUI, including rounds, player decisions, and game outcomes.
9. Test the GUI thoroughly, fixing any bugs and refining the user experience.

2024	Apr.				May			Summary	
Tasks:	1	2	3	4	1	2	3	Hours	Percent
Research	8	8						16	7%
Design		4	4					8	3%
Development			8	8	8			24	10%
Testing				5	5	5		15	6%
Modification					2	2		4	2%
Final Report					4	4	4	12	5%
Demonstration						4	4	8	3%
Hours	8	12	12	13	19	15	8	87	36%

Special Computing Platform

There is no special computing platform. The game will be able to run on any Windows or macOS computer.

Roles and Responsibilities

The whole team will be hands-on coders for all aspects of the game. Mohammed will handle the base Black Jack game. Tonirose will handle the graphical user interface and player interactions. Both of them will work on implementing the computer AI that the player will go against.

Evaluation

For our Black Jack game, there are two game modes: Game and Ai. In Game mode, the player is able to play against the AI dealer. In Ai mode, the user is able to see the Computer AI play against the Dealer AI. The Computer AI will analyze the card patterns through use of the

Monte Carlo approach and will find the best move. The Computer AI will start to win more rounds after five thousand iterations against the Dealer AI. At certain rounds, two graphs are generated to visualize the learning data of our Computer AI. One graph shows how many rounds it wins using Ace cards and another graph showing how many rounds it wins using no Ace cards. The number of winning rounds for the Computer AI is plotted against the Dealer AI.

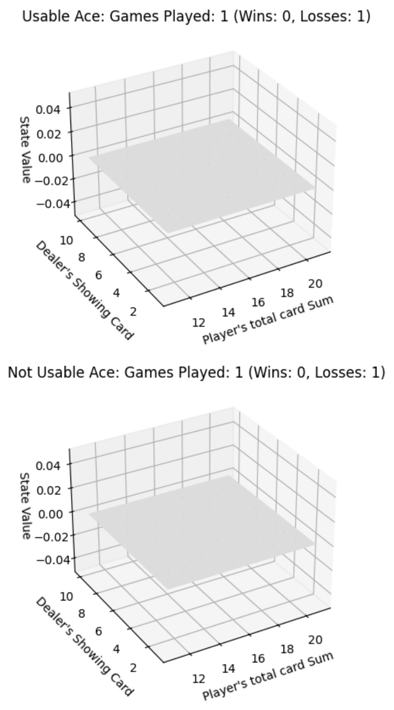

Figure 1: The AI has only played one game. It is not familiar with the game of Blackjack yet, which can be seen with how flat the plot is and how gray it is.

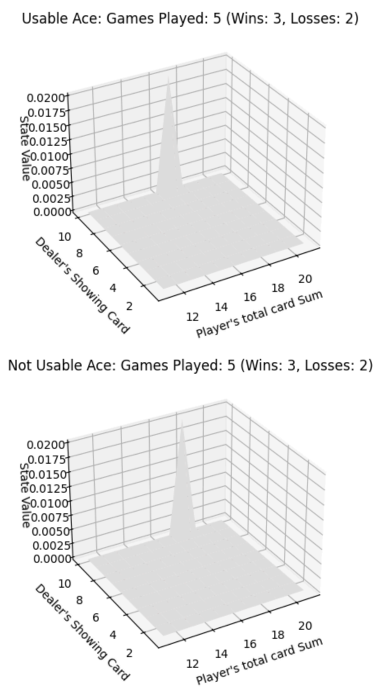

Figure 2: The AI only played 5 games and is still not familiar with Blackjack.

The plot shows a spike for playing more than 1 game and gray because it has not won any games and has not learned Blackjack yet.

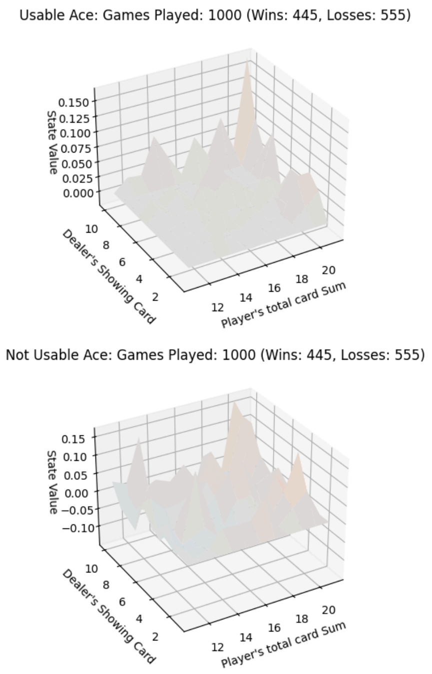

Figure 3: The AI has played 1000 games. It is learning how to play Blackjack better. Plot shows spikes for playing more than 1 game and a slight orange tint shows how much it learns.

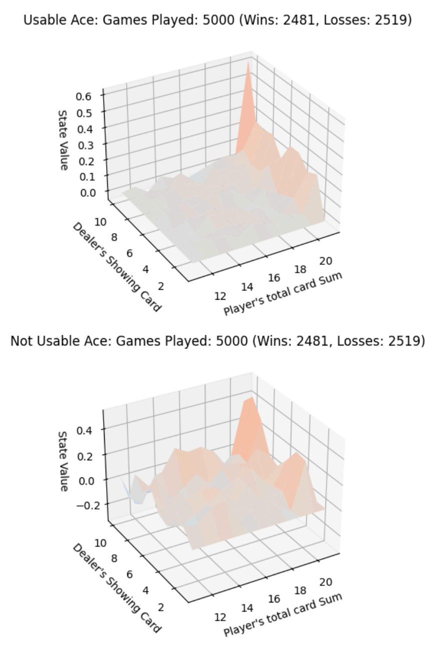

Figure 4: The AI has played 5000 games. It is learning how to play Blackjack better. Plot shows spikes for playing more than 1 game and a slight orange tint shows how much it learns.

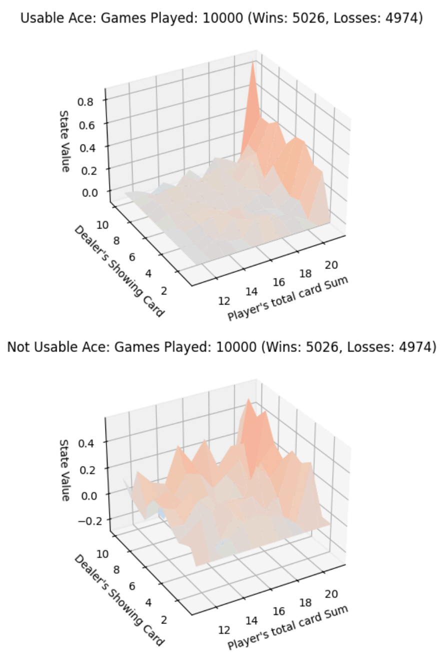

Figure 5: The AI has played 10000 games. It is learning how to play Blackjack better Plot shows spikes for playing more than 1 game and a slight orange tint shows how much it learns.

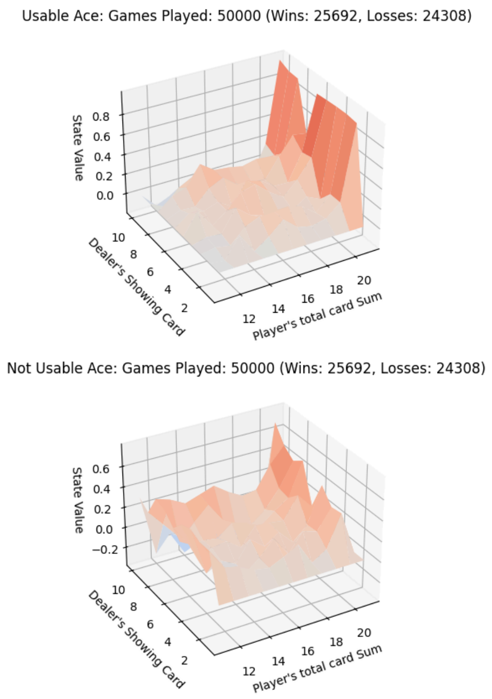

Figure 6: The AI has played 50000 games. It is learning how to play Blackjack better. Plot shows spikes for playing more than 1 game and a slight orange tint shows how much it learns.

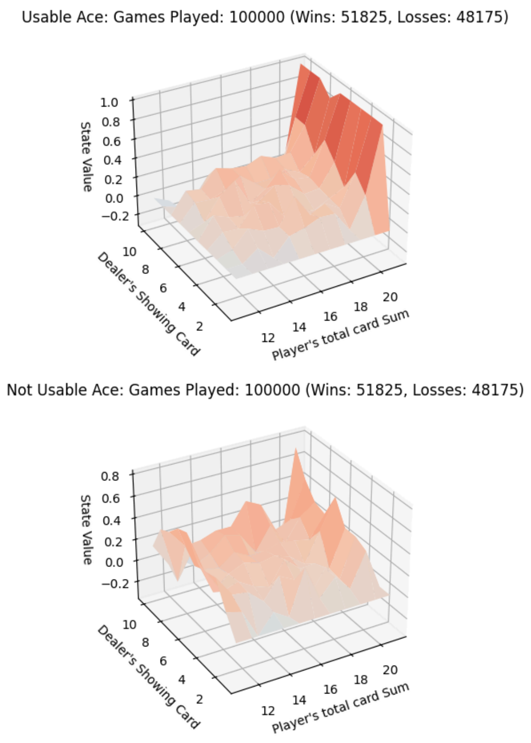

Figure 7: The AI has played 100000 games. It is learning how to play Blackjack better. Plot shows spikes for playing more than 1 game and more orange tint shows how much it's learning.

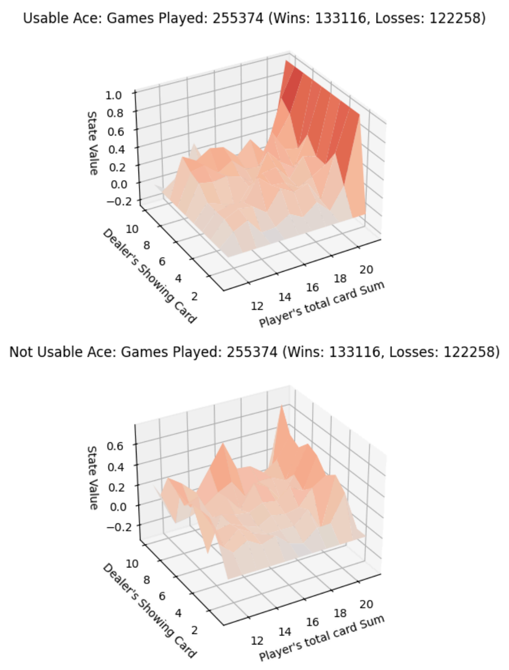

Figure 8: The AI has played 250000 games. It is learning how to play Blackjack better. Plot shows spikes for playing more than 1 game and more orange tint shows how much it's learning.

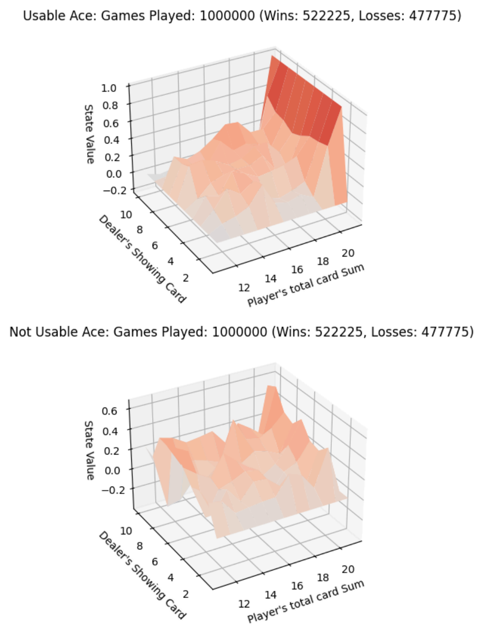

Figure 9: The AI has played 1000000 games. It is learning how to play Blackjack better. Plot shows spikes for playing more than 1 game and more orange tint shows how much it's learning. The AI is winning a lot more rounds using Aces.

Conclusion and Future Work

This project offers an opportunity to develop Python programming skills while creating an interactive and visually appealing Blackjack game with a GUI. It involves understanding the principles of GUI programming, implementing game logic within the GUI environment, and enhancing the user experience with visual and auditory elements. Some things that can be improved and implemented in the future is a settings option so the user can set up certain values through the GUI, counting cards, allowing the computer AI to bet, implement more players, casino-based simulation, and letting the game run without the use of a terminal.

References

Byrne, D. (n.d.). *Learning To Win Blackjack With Monte Carlo Methods*. https://towardsdatascience.com/learning-to-win-blackjack-with-monte-carlo-methods-61c90a52d53e

[Coding web forum]. (n.d.). Stackoverflow. https://stackoverflow.com/

Kamal, T. (2021, May 25). *Blackjack Monte Carlo Reinforcement Learning - Part 1* [Video]. Youtube. https://www.youtube.com/watch?v=NeusGkowXR4

[Lecture]. (n.d.). Udacity. https://www.udacity.com/catalog?promo=year_end&coupon=MAY40&utm_source=gsem_brand&utm_medium=ads_r&utm_campaign=19723941495_c_individuals&utm_term=154856657988&utm_keyword=udacity%20certification_p&utm_source=gsem_brand&utm_medium=ads_r&utm_campaign=19723941495_c_individuals&utm_term=154856657988&utm_keyword=udacity%20certification_p&gad_source=1&gclid=Cj0KCQjwxeyxBhC7ARIsAC7dS3-TghA__lZpV9O_Q5gfsub4qyo97Dz49gGvJghdiVz0g4sNC40gIBsaAsicEALw_wcB

Mr. P Solver. (2021, November 8). *How to: Monte Carlo Simulation in Python (Introduction)* [Video]. Youtube. https://www.youtube.com/watch?v=U00Kseb6SB4