Texas Hold'em Poker Game

1. Project Overview

Poker game with deep-learning hand evaluation.

2. Project Review

Study and then present a relevant existing project, outlining the specific improvements you plan to make. Highlight the key areas where your project enhances or builds upon the original.

This project originally does not have a duration tracker, by adding so would be something that I would do. Additionally, this project evaluates the winner in just a general way, however, I am thinking of using deep learning to evaluate the hand instead.

https://github.com/notaSWE/pokerhands.git

3. Programming Development

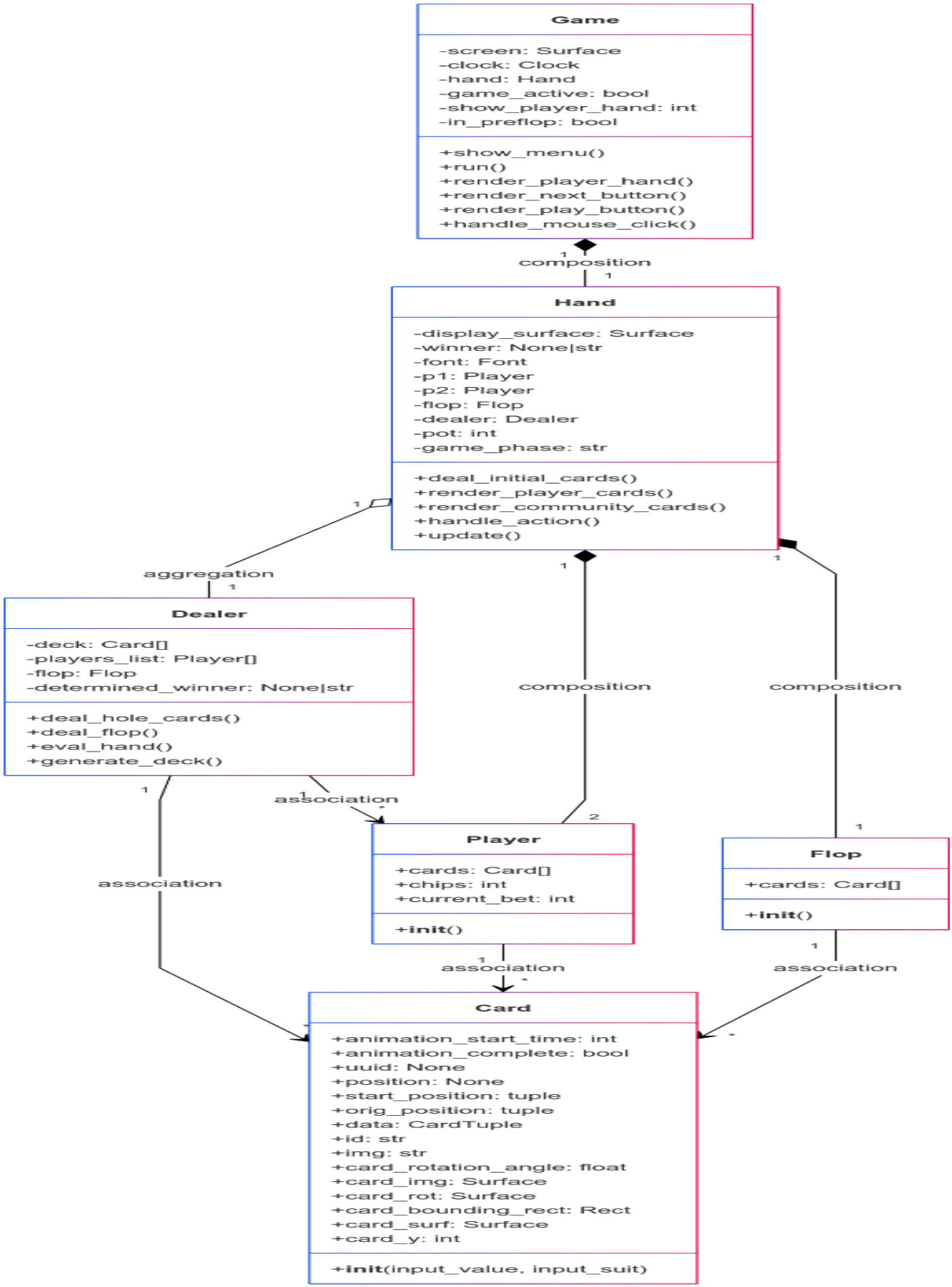
3.1 Game Concept

Every player draws 2 cards, in each player's turn, determining the action: fold, bet, or raise. After, drawing a Flop, 3 community cards can be evaluated with the player's hand, then determine the action again. After drawing a River, the 4th card in the community card, repeat the action. Then, draw a Turn, 5th, or last community card, and repeat the action. If there is still a player competing, each needs to show their card, then determine the score based on the combination of the card (Straight-Flush,

Flush, Straight, Full-House, Four of a kind, Three of a kind, Two pair, One pair, High card). This project will simulate a Texas Hold'em Poker game paired with Hand evaluation, to determine the winner of each round.

3.2 Object-Oriented Programming Implementation

Card - Simulate each card with value and suit
Player - The array for keeping the Card
Hand - Keep track of all cards of the player in the game
Game - Simulate the poker game, generating all necessary components


Game
-screen: Surface
-clock: Clock
-hand: Hand
-game_active: bool
-show_player_hand: int
-in_preflop: bool
+show_menu()
+run()
+render_player_hand()
+render_next_button()
+render_play_button()
+handle_mouse_click()
1
composition
1
Hand
-display_surface: Surface
-winner: None|str
-font: Font
-p1: Player
-p2: Player
-flop: Flop
-dealer: Dealer
-pot: int
-game_phase: str
+deal_initial_cards()
+render_player_cards()
+render_community_cards()
+handle_action()
+update()
1
aggregation
1
1
composition
1
composition
Dealer
-deck: Card[]
-players_list: Player[]
-flop: Flop
-determined_winner: None|str
+deal_hole_cards()
+deal_flop()
+eval_hand()
+generate_deck()
1
association
*
2
1
association
Player
+cards: Card[]
+chips: int
+current_bet: int
+init()
Flop
+cards: Card[]
+init()
1
association
*
1
association
Card
+animation_start_time: int
+animation_complete: bool
+uuid: None
+position: None
+start_position: tuple
+orig_position: tuple
+data: CardTuple
+id: str
+img: str
+card_rotation_angle: float
+card_img: Surface
+card_rot: Surface
+card_bounding_rect: Rect
+card_surf: Surface
+card_y: int
+init(input_value, input_suit)

3.3 Algorithms Involved

Hand evaluation using CFR.

4. Statistical Data (Prop Stats)

4.1 Data Features

1. *Duration of each game*
2. *Most drawn card/ hand*
3. *Winning percentage*
4. *Player's behavior based on winning percentage*
5. *Player's win rate*

Feature	Why it is good to have this feature data? What can it be used for?	How will you obtain 50 values of this feature data?	Which variable (and which class) will you collect this from?	How will you display this feature (via summarization statistics or via graph)?
Feature 1: Duration of each game	Helps analyze game pacing, balance, and player engagement. Useful for optimizing game design (e.g., adjusting phases like preflop/flop).	Through playing for 50 games, each game the statistic will be collected automatically.	Game Class	Graph: Histogram of game durations. Stats: Mean/median duration.
Feature 2: Most Drawn Card/ Hand	Identifies card distribution biases or popular hands (e.g., pairs, flushes). Useful for debugging RNG or balancing card weights.	Through playing for 50 games, each game the statistic will be collected automatically.	Card Class	Graph: Bar chart of card/hand frequencies. Stats: Mode (most frequent card/hand).

Feature 3: Winning Percentage for each player	Measures fairness and skill impact. Helps adjust AI difficulty or detect imbalances in starting hands.	Through playing for 50 games, each game the statistic will be collected automatic ally.	Hand Class	Graph: Pie chart or stacked bar. Stats: Win rate (%) per player.
Feature 4: Player's Behavior Based on Winning Percentage	Correlates actions (bet/fold) with outcomes to model strategies (e.g., aggressive vs. cautious play).	Through playing for 50 games, each game the statistic will be collected automatic ally.	Hand Class	Graph: Scatter plot (win rate vs. action frequency). Stats: Action ratios per win/loss.
Feature 5: Player's Win rate	Evaluates long-term performance and skill progression. Useful for leaderboards or adaptive AI.	Through playing for 50 games, each game the statistic will be collected automatic ally.	Hand Class	Graph: Line chart over time. Stats: Cumulative win rate.

3.2 Data Recording Method

I would like to store all the information in the form of a CSV file, as it is easy to read, and modify, and can be accessed easily by Python libraries such as pandas.

3.3 Data Analysis Report

I think using the tables would be the best way to suit all the data recorded.

3.3.1 Specify which feature you will use to present in the table

Feature	Statistical Values to Present in Table
Duration of each game	Mean, Median, Min, Max, Standard Deviation (SD)
Most Drawn Card/Hand	Mode (most frequent card/hand), Frequency Count
Winning Percentage	Average win rate (%), Win rate per player (%), SD of win rates

3.3.2 Specify which features you will use to create 3 graphs. Construct and include the table as shown in the bottom right corner of the attached image below.

Feature Name	Graph Objective	Graph Type	X-axis	Y-axis
Duration of each game	Show the distribution of game durations to analyze pacing and player engagement.	Histogram	Time	Frequency of games
Most Drawn Card/Hand	Compare the frequency of top 10 most drawn cards/hands for RNG validation.	Bar Chart	Card/Hand	Count
Player's Win Rate	Track win rate trends over 50 games to evaluate performance progression.	Line Graph	Game number	Cumulative win rate (%)

4. Project Timeline

Week	Task
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	Full proposal submission
3 (24 March)	Finished coding all Classes

4 (31 March)	Deep learning hand-evaluating algorithm
5 (7 April)	Bringing all together into one project
6 (14 April)	Submission week (Draft)

4.1 Provide weekly planning: what you plan to do each week.

Date	Plan
26 March-2 April	Game Menu and Initial Page of the Game.
3 April-9 April	Designing Game Logic, such as betting, calling, raising, and folding.
10 April-16 April	Designing Game Logic and making progress with the algorithm.
17 April-23 April	Complete the algorithm and Start working on Data part
24 April-11 May (Note that 28 April-9 May are final exam weeks)	Finish the data part

4.1.2 List 50% of the tasks that you expect to complete by 16 April.
All the game models must be complete, with animation, and game logic should be working perfectly.
Start working on the CFR algorithm.

4.1.3 List 75% of the tasks that you expect to complete by 23 April.
Complete doing the algorithm and start on doing data science part

4.1.4 List the remaining 25% of the tasks that you expect to complete by 11 May.
Finish doing the data science part

5. Document version

Version: *4.0*

Date: *31 March 2025*