TEXAS HOLDEM

WITH MVC AND OBJECTS WITH JAVASCRIPT AND PHP

1. PROJECT GOAL

Create Texas Holdem game

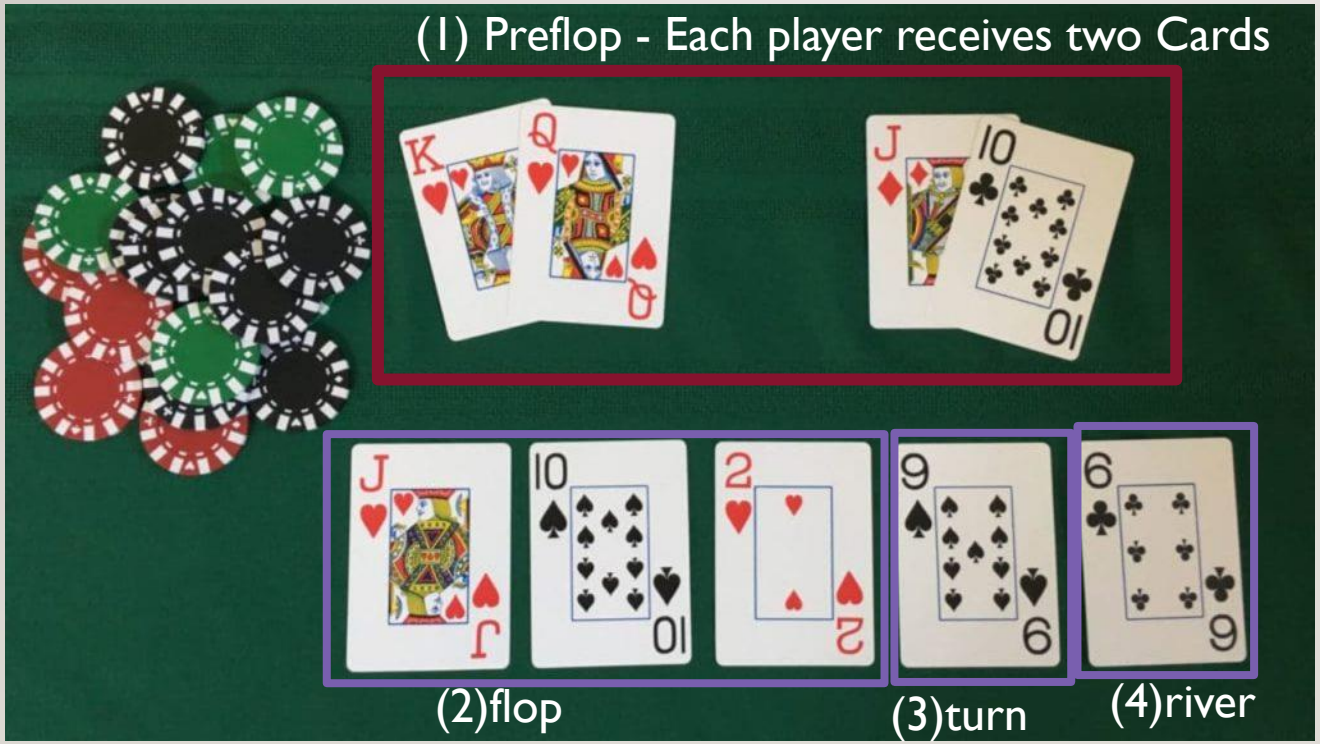
By using MVC or Model-View-Controller

Using Objects of JavaScript and php

2. STEPS

Step1	Make a program with C++ to figure out how to embody Texas Holdem card game into a computer program Analyze it to plan how to utilize JavaScript and php to our project
Step2	Understand MVC concept throughout comparison between sample game, BattleShip and Texas Holdem
Step3	Apply objects created by C++ to JavaScript and php with the concept of MVC

UNDERSTANDING OF TEXAS HOLDEM 1
- HOW TO DISTRIBUTE CARDS



Each Player	Preflop (2Cards)	Bet 1
	◆	
Community	Flop (3Cards)	Bet 2
	◆	
	Turn (1Card)	Bet 3
	◆	
	River (1Card)	Bet 4

◆ - Dealer throw away a card of deck

UNDERSTANDING OF TEXAS HOLDEM 2 - RANK OF HANDS

- Texas Holdem is one of Poker game which has a unique rank of hands created by **combinations of pairs and sequences** of face and suit with five cards

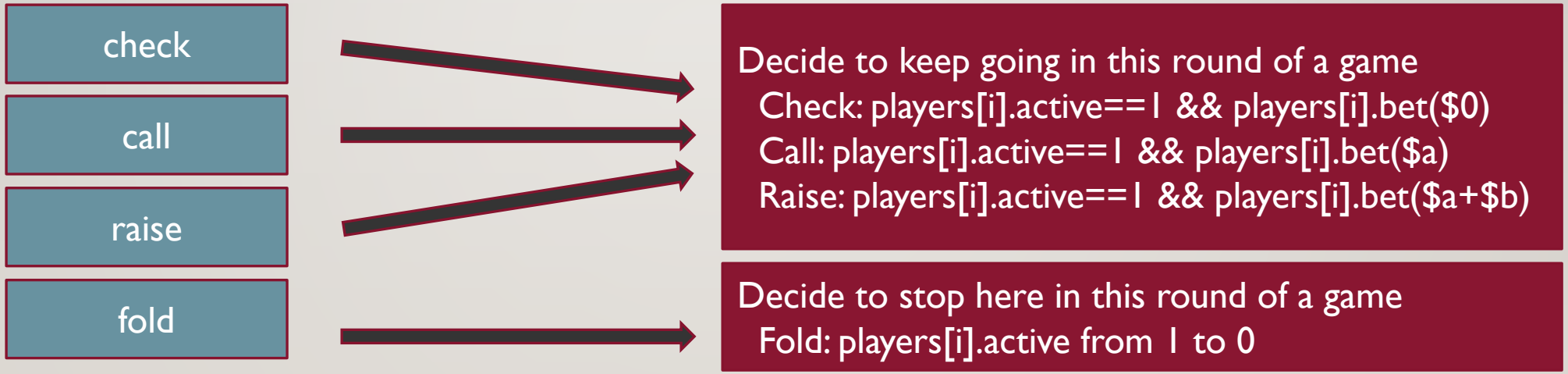
Rank of Hands		Examples
(Royal) Straight Flush	Straight + Flush	
Four of a kind		
Full House	Three of a kind + a pair	
Flush	Five cards with same kind of suit	
Straight	Five cards in a sequence	
Three of a kind, Two pairs, One Pair		

UNDERSTANDING OF TEXAS HOLDEM 3 - GIVEN OR PREDESTINED CONDITION

- The number of players
- The result of shuffle
- Game Rule e.g. Big Blind should initially put money into the pot mandatorily (Small Blind ~ ½)
  Dealer distributes cards with a clock-wise order
  Procedure ~ "preflop", "flop", "river", "turn"
  After each procedure, dealer throw away a card
- Betting strategy of another players (Bluffing vs Reasonable Decision) – That is, it's not only about a calculation of probability. Rather, it's about a figuring out another player's betting strategy and its pattern. In this reason, our project is limited with human players.

UNDERSTANDING OF TEXAS HOLDEM 4 - WHAT PLAYERS CAN DECIDE

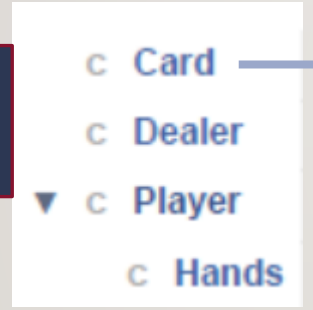
- Join/quit/just watch a game
- Betting strategy: check / call / raise / fold



($a= amount that previous player bet, $b=additional amount current player bet)

OBJECTS IN C++
- (1) CARD

Class Hierarchy

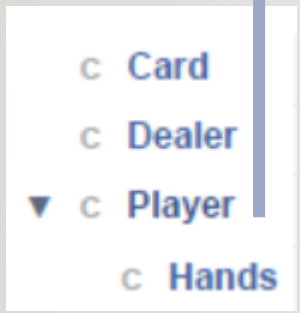


(1) Card Membership List

Card(int num)	Card	
getFace() const	Card	inline
getFaceName() const	Card	inline
getNumber() const	Card	inline
getPict() const	Card	inline
getSuit() const	Card	inline
getSuitName() const	Card	inline
setFace()	Card	
setPict()	Card	
setSuit()	Card	
toString()	Card	

OBJECTS IN C++
- (2) PLAYER, (3) HANDS : PLAYER

Class Hierarchy



(2) Player Membership List

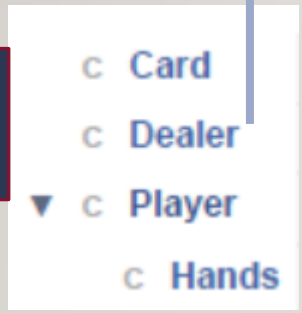
addBal(int pot)	Player	inline
addMyCards(Card card)	Player	
bet(int amount)	Player	inline
getActStatus()	Player	inline
getBalance() const	Player	inline
getMyCards()	Player	inline
getName() const	Player	inline
getPlayerAct()	Player	inline
Player()	Player	
putInThePot()	Player	inline
resetMyCards()	Player	inline
setInThePot()	Player	inline
setPlayerAct()	Player	inline
setPlayerBal()	Player	inline
setPlayerInact()	Player	inline
setPlayerName(string name)	Player	inline
~Player()	Player	inline

(3) Hands Membership List

addBal(int pot)	Player	inline
addMyCards(Card card)	Player	
bet(int amount)	Player	inline
checkStraight()	Hands	
getActStatus()	Player	inline
getBalance() const	Player	inline
getFaces()	Hands	inline
getHands()	Hands	
getHandsName()	Hands	
getMyCards()	Player	inline
getName() const	Player	inline
getPlayerAct()	Player	inline
getSuits()	Hands	inline
Hands()	Hands	
Hands(const Hands &orig)	Hands	
Player()	Player	
putInThePot()	Player	inline
resetHands()	Hands	inline
resetMyCards()	Player	inline
setFaces()	Hands	
setInThePot()	Player	inline
setPlayerAct()	Player	inline
setPlayerBal()	Player	inline
setPlayerInact()	Player	inline
setPlayerName(string name)	Player	inline
setSuits()	Hands	
~Hands()	Hands	inline
~Player()	Player	inline

OBJECTS IN C++
- (4) DEALER

Class Hierarchy



(4) Dealer Membership List

bettingPrompt1()	Dealer	
bettingPrompt2(int num, int amount)	Dealer	
bettingPrompt3(int num, int amount)	Dealer	
calBal()	Dealer	
Dealer(int num)	Dealer	
decideWinner()	Dealer	
displayPlayersInfo(int num)	Dealer	
flop()	Dealer	
getBigBlind()	Dealer	inline
getDeck()	Dealer	inline
getNumAct()	Dealer	
getNumPlayers() const	Dealer	inline
getPlayers() const	Dealer	inline
getPotAmount()	Dealer	inline
getRound() const	Dealer	inline
getRound()	Dealer	inline
getSmallBlind()	Dealer	inline
nextRound()	Dealer	inline
preflop()	Dealer	
resetGame()	Dealer	
resetPot()	Dealer	inline
river()	Dealer	
setBlind()	Dealer	
setCards()	Dealer	inline
setIniCont(int num)	Dealer	inline
setPlayers(int num)	Dealer	
setRound()	Dealer	inline
shuffle()	Dealer	
turn()	Dealer	
~Dealer()	Dealer	inline

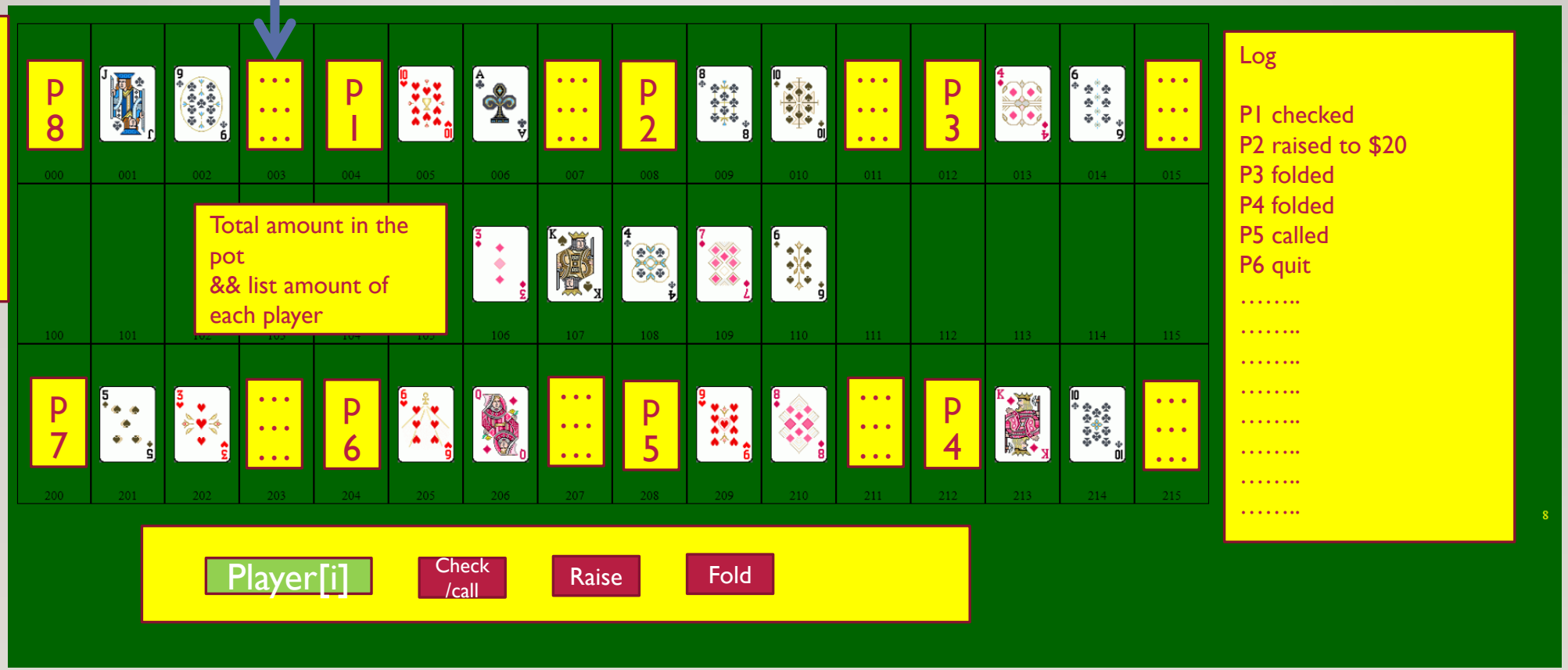
MVC
- BATTLESHIP VS. TEXAS HOLDEM 1

	Battle Ship	Texas Holdem
Player's Discretion	Guess = the location of a table	Decide to keep going or Stop the game with betting strategy
Given Condition	Locations of ship created randomly	Order of cards throughout shuffling
View::displayMessage function (msg)	**1. Model::fire function(guess)** (1)"Oops, you already hit that location" (Input validity) (2)"You sank my battleship!" (3)"HIT!" and "You missed!" **2. Controller::processGuess(guess)** (1)"You sank all my battleships …"	**1. Deal::preflop(), flop(), river(), turn()** (1) "player[i] checked" (2) "player[i] called $x" (3) "player[i] raised $x" (4) "player[i] folded" **2. Controller::decideWinner(Hands);** (1) "Player[i] won $x in 10th round" **3. Controller::checkPlayers(Deal);** (1) "player[i] joined/quited" (2) "player[i] won and earned total $X" (ini & terminate)

ATTACHMENT 1 - CSS & HTML OF TEXAS HOLDEM

* - *plan*

Name: …
CurrentBalance: $
Money in the Pot: $



MVC
- BATTLESHIP VS. TEXAS HOLDEM 2

	Battle Ship	Texas Holdem
View::display[picture]	Model::fire function(guess) calls view::displayHit(location) and view::displayMiss(location)	Deal::preflop, flop, river, turn function() calls view::displayCard(location, players[i].myCards[j].picture)
Initiate a game	HandlersInit.js with guess input	- HnadlersInit.js with number of players input - After initiating the game, we have to handle betting buttons similarly (waiting next players decision or dealer's action)
Model	1. Locate ships randomly to cells of the table 2. Reflect player's guess to the location 3. Show the result of player's guess	Classes : Card => 2. Player => 3. Deal => 4. Hands

MVC
- BATTLESHIP VS. TEXAS HOLDEM 3

	Battle Ship	Texas Holdem
Controller.js	1. It says when this game is done such as the case that player hits all the ships. 2. Translates user's input of guess ("A2") to array location of table("02")	1. Round control (static NROUND ++) => Card distribution order change => rotates the order of big and small blind => As NROUND increases, base amount goes up 2. In the betting process, it changes the status of the player (when they fold, players[i].active = 1 ➔0) 3. It Decides a Winner and verifies the changes of balance for each player (players[i].bal >0, betting amount <players[i].bal) 4. When a player quit, removes the player[i]'s object 5. When a player took all the balance of other players or all the players quit, then it terminates the game. 6. It asks if players want to do another game and prepare the next game with resetting players.

IMPROVEMENTS
- TO DO LIST FOR FINAL

- Figuring out a way to communicate between php and JavaScript
- Players can log in and have their own different view. They also can see other players game without joining the game.
- Exceptional case study, e.g. all-in situation(call betting amount > players[i].balance), immediate winning decision when all other players folded
- Display various message depending on each event
- Do…while memory clearing issue

```
Do{
        play game (preflop() → flop() → turn() → river() →decideWinner() → resetRound()
        nRounds++;
        rotate distributing cards order;
}while(all players not quit || more than two players still play game)
```