Briscola a chiamata

# Descrizione del programma

Dati 4 giocatori reali, che vogliono giocare a Briscola a Chiamata, con carte da gioco fisiche, il programma svolgerà il ruolo di quinto giocatore. Ricevendo in ingresso, a ogni turno, tutte le informazioni, il sistema giocherà esattamente come un quinto giocatore, mostrando a schermo a ogni suo turno la carta giocata.

Il programma è utilizzabile da PC, per giocatori di livello medio, con uno stile di gioco conservativo.

Si è scelto un programma PC dopo una analisi attenta dell’utilizzo dei giochi di carte online in Italia: le piattaforme maggiormente utilizzate sono burraconline e boardgamearena, entrambe si concentrano prevalentemente sulle loro versioni per PC per due motivi:

* L’utilizzatore medio ha una età per la quale l’utilizzo del PC viene preferita all’utilizzo di smartphone, sia per una questione di semplicità sia per una questione visiva, lo schermo dello smartphone è spesso troppo piccolo.
* La visualizzazione di tante carte su uno smartphone è complessa e spesso risulta scomoda, in giochi come briscola tra le carte in mano e quelle in gioco si arriva a un totale di 12 carte contemporaneamente.

Sulla base di queste informazioni quindi si è optato per un software PC

# Analisi del gioco

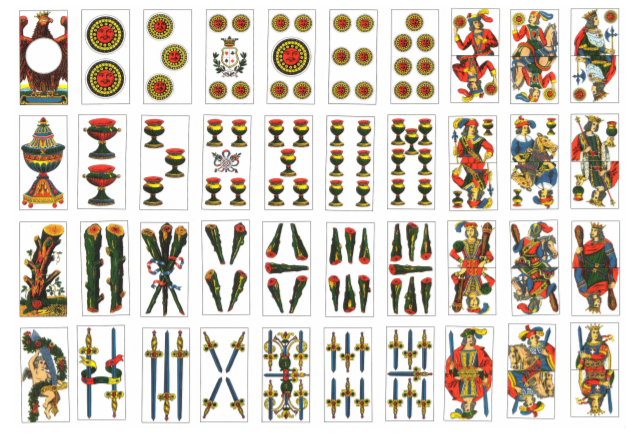
Per la Briscola si utilizza il classico mazzo italiano da 40 (Fig. 1.1), L’ordine di valore delle carte prevede, dalla più alta alla più bassa: Asso, Tre, Re, Cavallo, Donna, 7, 6, 5, 4 e 2. Come viene riassunto nulla Tabella 1.1, a ogni carta è associato un valore. Asso e Tre sono detti Carichi e valgono rispettivamente 11 e 10 punti; Re (4 punti), Cavallo (3 punti) e Donna (2 punti) sono detti Carte Vestite, mentre le carte rimanenti, dette Scartine, non hanno valore. Sommando i punteggi delle singole carte, il totale del mazzo `e di 120 punti.

Figure .1

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Carta | 1 | 3 | re(10) | Cavallo(9) | Fante(8) | 7 | 6 | 5 | 4 | 2 |
| punteggio |  |  |  |  |  |  |  |  |  |  |

tabella .1

# Fasi del gioco:

## Asta:

Il sistema **non** partecipa all’asta

* lettura da terminale della carta chiamata, del giocatore chiamante e del punteggio da raggiungere.

## Mano:

Ogni partita è formata da 8 mani consecutive.

* Lettura da terminale della carta inserita dal singolo giocatore
* Assegnazione del valore della carta
* Decisione dalla carta da giocare in base alle due azioni precedenti
* Presentazione a schermo della carta giocata
* Salvare il valore carta-punteggio-giocatore
* Sommare i valori delle carte uscite
* Assegnare al giocatore vincente la somma dei valori
* Eliminare le carte uscite dalle carte possibili

# Svolgimento del gioco:

* Si sceglie randomicamente il ruolo del mazziere alla prima partita
* Il giocatore alla sua destra inizia l’asta e sarà anche il primo a giocare ad asta conclusa
* Ogni giocatore, a turno, chiama o passa
* L’asta continua tra i giocatori che non hanno passato. I punti sono chiamati in ordine crescente
* L’asta si conclude quando un ultimo giocatore è rimasto a chiamare, esso diventa il “chiamante” della partita
* Il “chiamate” dichiara la carta scelta, definendo così il seme di briscola
* Il giocatore che ha in mano la carta chiamata diventerà così il “chiamato” che sarà in squadra con il “chiamante”.
* La divisione delle squadre non è nota ai giocatori
* La partita inizia dal primo giocatore a destra del mazziere
* Il vincitore di ogni mano è decretato seguendo le regole della briscola
  + La prima carta giocata definisce il seme dominante della mano
  + Qualsiasi carta del seme di briscola vince sul seme dominante della mano
  + La gerarchia delle carte è definita dal valore, in ordine decrescente: asso-tre-re-cavallo-fante-sette-sei-cinque-quattro-due
* Il vincitore di ogni mano gioca per primo la mano seguente finché non si esauriscono le carte (8 mani totale di gioco, quante le carte distribuite a ogni giocatore ad inizio partita).
* A fine partita ogni giocatore somma il punteggio delle carte che ha vinto con quelle dei giocatori della propria squadra
* Si confrontano i punteggi e si decretano i vincitori. Per poter vincere la squadra composta da chiamate e chiamato devono arrivare almeno al punteggio deciso durante l’asta.

# Giocatore

* nome
* posizione
* flagChiamante
* puntiVittoria
* carte[8] (viene aggiornato ogni turno dalle carte che gioca)
* prese[8] (vettore delle mani di gioco)
* mano[5] (vettore delle carte giocate ad ogni mano)
* probabilitaChiamato[8] (probabilità aggiornata di turno in turno)

# Funzionamento del programma:

## Inserimento dei giocatori e delle carte:

* Inserire i nomi dei giocatori e la posizione nel tavolo
* Ogni giocatore a turno inserisce le proprie carte nel sistema
* Il sistema riconosce le proprie carte sottraendo le 32 carte dei giocatori dalle 40 totali

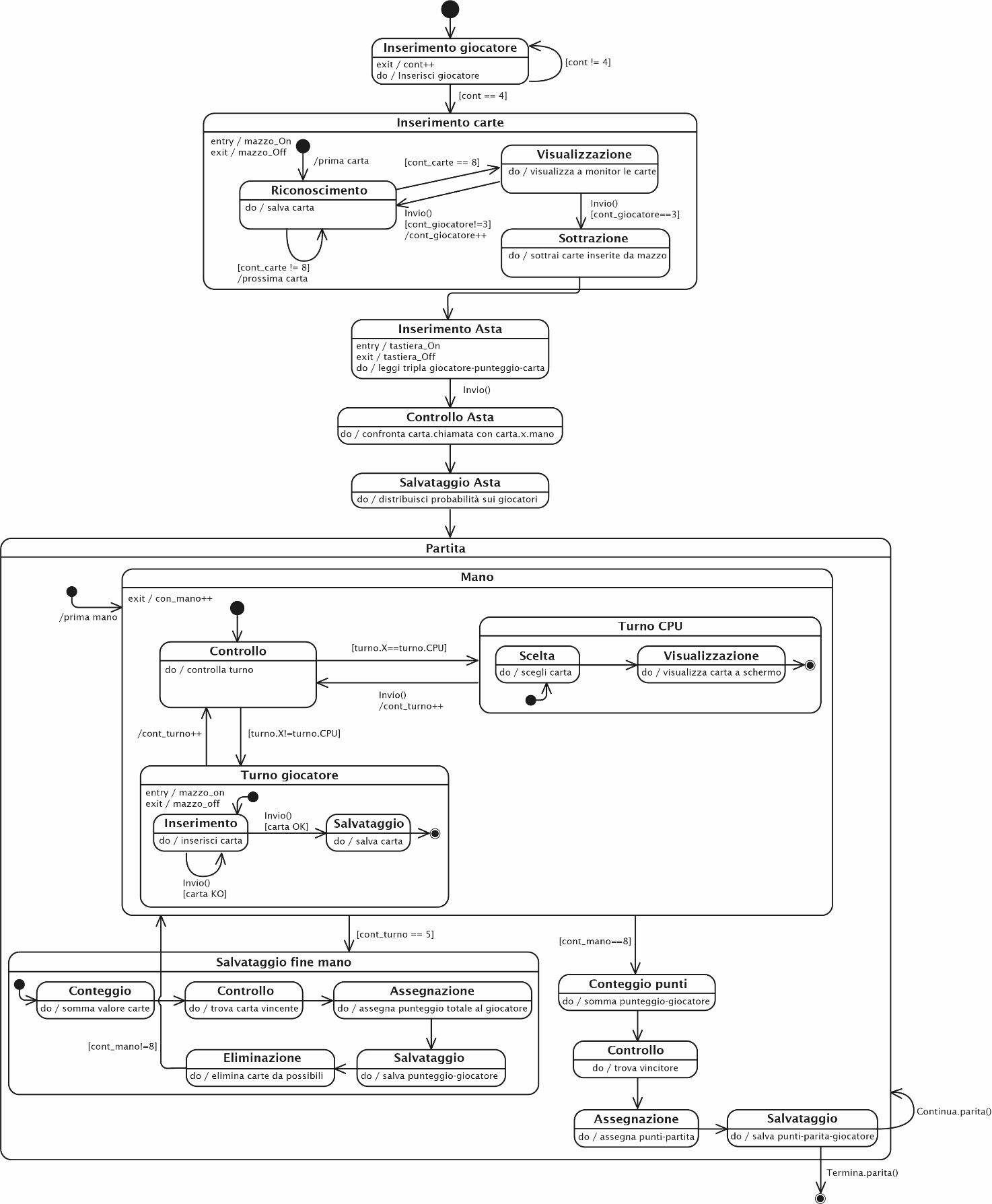
## Asta:

* Inserire il nominativo del giocatore chiamante: flag\_chiamante aggiornato a 0 o 1 per ogni giocatore
* Inserire il punteggio e la carta chiamati, tripla giocatore -carta-punt\_vittoria
* Il sistema controlla di possedere o meno la carta chiamata.
* Definisce la modalità di gioco e imposta il prob[1] per ogni giocatore

## Mano:

* Se tocca al sistema, gioca una carta ed aumenta il contatore del turno
* Se tocca al giocatoreX, il sistema legge la carta giocata (i.e. assegna il valore a carte[i]) ed aumento il contatore del turno
* Quando il contatore del turno è arrivato a 5 (i.e. tutti i giocatori hanno giocato) il sistema trova la carta vincente, e aggiorna la riga della matrice prese con le 5 carte sul tavolo Sommando i punteggi della riga della matrice calcola i punti complessivi della mano
* Ogni fine turno il contatore delle mani viene aumentato di uno
* Ad ogni mano il sistema aggiorna i pesi di ogni carta e la probabilità che i singoli giocatori siano compagni
* Quando la carta chiamata compare sul tavolo da gioco la probabilità di essere compagno viene settata ad 1 sul giocatore che ha giocato la carta
* Quando il contatore delle mani arriva a 8 il sistema si ferma e sommando i vari punteggi dei giocatori decreta la squadra vincente con i relativi punti

## Statechart:



# Requisiti:

## Requisiti funzionali

|  |  |
| --- | --- |
| No. | Descrizione |
| 1 | Il sistema deve permettere l’inserimento di stringhe da terminale |
| 2 | Il sistema deve permettere l’inserimento di interi da terminale |
| 3 | Il sistema deve automaticamente riconoscere le carte durante la partita |
| 4 | Il sistema deve automaticamente definire le sue carte in funzione di quelle inserite |
| 5 | Il sistema deve essere fornito di tastiera alfanumerica |
| 6 | Il sistema deve essere fornito di schermo |
| 7 | Il sistema deve avere un’interfaccia interattiva |
| 8 | Nessun giocatore deve poter vedere le carte del sistema |
| 9 | Il sistema non deve memorizzare le carte dei giocatori |
| 10 | Il sistema è sempre il giocatore No.5 |
| 11 | Il numero del giocatore equivale all’ordine di gioco in ordine crescente |
|  |  |

## Requisiti non funzionali

|  |  |
| --- | --- |
| No. | Descrizione |
| 1 | Quando è il suo turno, il sistema deve giocare una carta in 10 secondi |
| 2 | Il sistema deve avere due modalità di gioco: “non-chiamato” e “chiamato” |
| 3 | Il sistema deve essere compatibile con Windows |
| 4 | Il sistema deve avere una gerarchia di scelte |
| 5 | Il sistema deve avere una lista di condizioni di decisione |

## Vincoli

|  |  |
| --- | --- |
| No. | Descrizione |
| 1 | Il sistema non può giocare più di una carta |
| 2 | Ci devono essere 4 giocatori |

# Architettura: Model-View-Controller (MVC)

I component dell’architettura MVC sono programmati per gestire diversi aspetti di una applicazione in sviluppo. L'obiettivo di MVC è quello di separare la business logic [logica di funzionamento] dalle considerazioni relative all'interfaccia utente. In particolare:

1. Model:

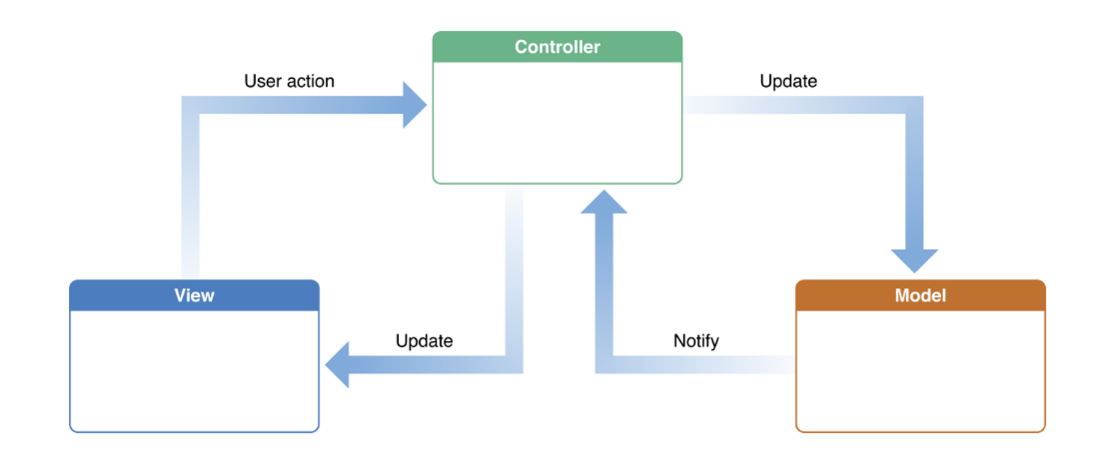
È il componente centrale del modello. Contiene i metodi di accesso ai dati. Gestisce direttamente i dati, la logica e le regole dell’applicazione.

1. View:

Si occupa di visualizzare i dati all’utente e gestisce l’interazione fra quest’ultimo e l’infrastruttura sottostante.

1. Controller:

Riceve i comandi dell’utente attraverso il View e reagisce eseguendo delle operazioni che possono interessare il Model e che portano generalmente ad un cambiamento di stato del View.



### Model:

Il “database” dell’applicazione sarà in parte predefinito e in parte aggiornato in varie fasi del gioco. La parte statica definirà le carte, il loro valore e le regole del gioco assieme alle strategie. La parte dinamica riguarderà i giocatori e sarà aggiornata durante la partita. In particolare, la classe giocatore sarà aggiornata in diversi momenti della partita:

* nome: aggiornato ad inizio partita con il nome di ogni giocatore
* posizione: aggiornata ad inizio partita con l’ordine dei giocatori
* flagChiamante: aggiornato una sola volta ad inizio partite per tenere traccia del giocatore che ha vinto l’asta.
* puntiVittoria: aggiornato dopo la conclusione dell’asta tra i giocatori. È un valore compreso tra 1 e 120 che indica i punti necessari di ogni giocatore per vincere.
* carteInMano[8]: aggiorna di mano in mano le carte giocate
* mano[5]: ogni mano viene aggiunta la carta giocata
* prese[8]: vettore aggiornato delle mani
* probabilitaChiamato: ogni turno viene aggiornata il valore di probabilità di ogni giocatore di essere compagno. Inizialmente se il giocatore CPU non è il chiamato la probabilità è distribuita tra i restanti giocatori.

### View:

L’interfaccia con l’utente richiede l’inserimento delle informazioni necessarie (nome giocatori, posizione, carte) e visualizza la carta giocata quando richiesto.

### Controller:

Ad ogni interazione con il giocatore, chiama le funzioni necessarie per salvare i dati e, nel caso fosse il turno giocatoreCPU, confronta la situazione della mano (carta giocata, giocatore) e controlla tra le condizioni di strategia per scegliere la carta corretta da giocare.

Per lo sviluppo è stato scelto

## Funzioni e modelli utilizzati:

funzioni globali:

* inserisci\_carte: visualizza a schermo le carte, permette di selezionare le carte della mano del giocatore, salva le carte nell’array giocatore->carte, restituisce l’array giocatore->carte
  + Visualizza: visualizza carta a schermo
  + Mazzo\_on: visualizzazione del mazzo a schermo
  + Carte\_CPU: sottrazione tra carte del mazzo e le carte dei giocatori, reset dell’array giocatore->carte.
* Int calcolo\_punti(carta, punteggio): data in ingresso una carta ne trova il valore e lo somma a un intero dato in ingresso, restituisce un intero

Struttura:

Modulo 1- Inserimento dati:

* Inserisci\_dati: legge una stringa a schermo e la salva nel database come giocatore->nome
* Inserisci\_carte: tramite due cicli for viene reiterata la funzione inserisci\_carte fino ad aver assegnato 8 carte ai 4 giocatori, per un totale di 32 cicli, per ogni ciclo la carta uscente viene assegnata alla variabile giocatoren->carte[i]
* Carte\_CPU: sottrae alle 40 carte totali le 32 appena assegnate per trovare le 8 carte rimanenti ed assegnarle a CPU->carte[i]

Modulo 2- Asta:

* Asta: legge a schermo il giocatore chiamante, la carta chiamata e il punteggio di vittoria ed aggiorna
  + Int Check\_compagno: confronta giocatoreCPU->carte con la carta chiamata, ritorna 1 se compagno, 0 se non compagno
  + Aggiorna\_prob: aggiorna la probabilità di essere compagno tra i giocatori:
    - se check\_compagno==1, tutte le probabilità sono=0 tranne quella di CPU che è =1
    - se check\_compagno==0, le probabilità dei tre giocatori vengono inizializzate a 0.33 (ovvero la probabilità che ognuno di loro ha di essere il compagno) mentre quella di CPU = 0

Modulo 3- Svolgimento mano:

* + controllo\_turno: confronto int turno\_corrente con int CPU->posizione
    - if turno\_corrrente==CPU->posizione:
      * turno\_CPU:
        + scelta\_carta: utilizza le informazioni salvate in giocatore e ritorna la carta da giocare
        + calcolo\_punti
        + visualizza: visualizza carta a schermo
    - if turno\_corrente!=CPU->posizione:
      * turno\_giocatore:
        + inserisci\_carta:
        + aggiorna\_prob: aumenta o diminuisce giocatoren->prob in base a diversi criteri.
        + calcola\_vincitore\_mano: date in ingresso 5 carte e il seme della briscola, le confronta e aggiorna la matrice giocatore->prese

Modulo 4- conclusione\_partita:

* + calcolo\_punti
  + char trova\_vincitore: somma i punti dei giocatori giocatore->flag==1 e li confronta con punti\_per\_vincere. Se somma>=punti\_per\_vincere allora ritorna giocatore->nome
  + int assegna\_punteggio: aggiorna con il punteggio di ogni giocatore la matrice
  + Inizio nuova partita

Immagine che contiene testo, dispositivo, metro

Descrizione generata automaticamente

Analisi funzioni complesse:

* aggiorna\_prob: la funzione aggiorna\_prob permette di aggiornare le probabilità dei 3 giocatori rimasti, escludendo CPU e Chiamante, partendo dal valore di 0.33, inizializzato precedentemente: essendo 3 i giocatori rimasti e 1 la carta chiamata le probabilità che il giocatore n la abbia sono 1/3.

Partendo da 0.33 vengono aggiunti o sottratti dei valori compresi tra -0.3 e 0.3. I seguenti valori sono stati calcolati in maniera statistica.

I punteggi dipendono interamente dalle mosse fatte dai restanti tre giocatori, non vengono influenzati dai comportamenti di Chiamante e CPU

Criteri:

|  |  |
| --- | --- |
| Mossa | Punteggio |
| Giocatore gioca briscola ≥ 8 prima del chiamante | -0.3 |
| Giocatore gioca briscola ≥ 8 dopo il chiamante | +0.1 |
| Giocatore gioca briscola ≤ 8 prima del chiamante | -0.2 |
| Giocatore gioca briscola ≤ 8 dopo il chiamante | +0.2 |
| Giocatore gioca briscola > di una briscola giocata dal chiamante | -0.3 |
| Giocatore gioca carta con valore ≥ 10 prima del chiamante | +0.3 |
| Giocatore gioca carta con valore ≥ 10 dopo il chiamante | -0.2 |
| Giocatore gioca carta con valore ≥ 10 su una briscola del chiamante | +0.3 |
| Giocatore gioca carta con valore ≥ 10 dopo chiamante | -0.3 |

Questi criteri si basano su mosse che vengono compiute dai tre giocatori rimanenti che avvantaggiano il chiamante o lo svantaggiano, ognuna per i suoi motivi, ovviamente tenendo anche conto della componente di bluff che sta alla base di qualsiasi gioco di carte e, soprattutto, contando che non sempre un giocatore ha una alternativa valida ad una particolare giocata che potrebbe metterlo in cattiva luce agli occhi degli altri giocatori. Per questo motivo i valori dati a ogni mossa non sono definitivi ma spostano le probabilità che un giocatore sia o meno il compagno del chiamante in maniera più o meno netta.

I valori di questi criteri sono stati estratti sulla base di 20 partite con sempre gli stessi giocatori, uno sviluppo interessante sarebbe quello di analizzare ogni partita e pesare i valori in maniera più precisa, tenendo però sempre conto del fatto che ogni giocatore gioca in maniera diversa, chi più aggressiva, chi più conservatrice.

È poi importante sottolineare che il numero dei criteri non è casuale, se ne potrebbero aggiungere numerosi altri; tuttavia, questa scelta non sarebbe coerente con lo scopo del software, ovvero creare un supporto per giocatori di livello medio di briscola, non professionisti. Dei giocatori di medio livello danno importanza a questi nove criteri per la scelta della carta e quindi per capire chi è il compagno, aggiungere altri criteri significherebbe dare importanza a mosse che non sono state fatte con l’intento di avvantaggiare il chiamante o meno. La probabilità verrebbe quindi sporcata da questi comportamenti, che invece non avvengono a giocatori di livello alto che ragionano ogni mossa.

Uno dei motivi alla base della nostra architettura è anche la possibilità di tenere uno storico delle mosse attraverso il database, cosa che può poi essere sfruttata in un eventuale algoritmo di machine learning.

* scelta\_carta: la funzione scelta carta è il cuore del nostro software nonché il momento più importante del gioco, è la funzione che maggiormente verrà testata nella fase di testing e che influenzerà in maniera decisiva il funzionamento.

La scelta dipende dalle carte che CPU ha in mano

Ci sono tre modalità di funzionamento diverse che dipendono dalla fase del gioco e dalla distribuzione delle carte:

* + CPU ha in mano la carta chiamata: CPU->probabilitàChiamato==1

In questo caso CPU è il compagno del chiamante, la strategia è quella di tenere la carta chiamata per penultima, in modo da non farsi scoprire dagli altri giocatori. Verranno quindi giocate le carte di più basso valore per prime, poi le briscole e infine le carte con valore ≥ 10. Come penultima carta verrà giocata la briscola. Questa è la struttura della strategia di gioco, che sarà quindi gerarchica, dalle carte con minor valore a quelle col valore maggiore. C’è solo una variante a questa strategia ed è nel caso in cui la carta chiamata sia un asso, in quel caso verrà giocata come ultima carta e non penultima.

* + CPU non ha in mano la carta chiamata: CPU->probabilitàChiamato==0

Questa seconda modalità di gioco avrà due fasi:

* + - Il chiamato non si è ancora scoperto: giocatore->probabilitàChiamato≠1

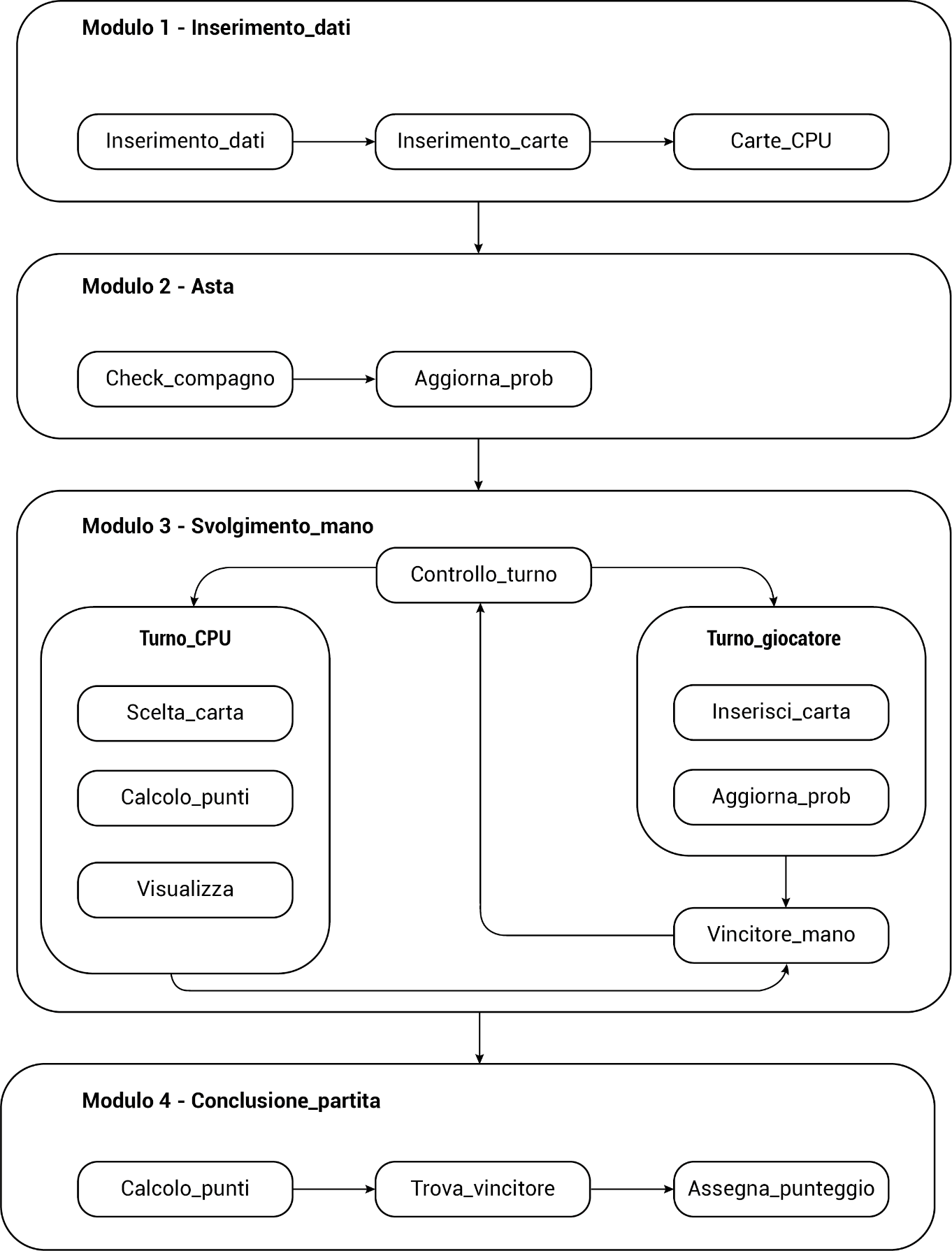
In questa fase la CPU gioca con una struttura gerarchica diversa: inizia giocando le briscole sotto la carta numero 7 se nessun altro giocatore ha messo una briscola di livello superiore sul campo, in quel caso, se il giocatore che ha giocato la briscola è il Chiamante o uno dei due giocatori con probabilitàChiamato maggiori, la CPU gioca la carta di valore più basso, altrimenti gioca la carta con valore più alto.

Quando la CPU ha finito le briscole basse e non ci sono briscole per terra gioca la carta di valore minore che ha.

Quando contatoreTurno≥5 CPU gioca le briscole di valore superiore a 7, se nessun altro giocatore ha messo una briscola di livello superiore sul campo, in caso non ne avesse continua come sopra.

* + - Il chiamato si è scoperto: giocatore->probabilitàChiamato≠1

Nel momento in cui viene scoperto il giocatore chiamato, CPU gioca la carta più alta che ha nel caso ci sia una briscola degli altri due giocatori, in caso non ci fosse gioca la carta più bassa.



# Implementazione:

## Linguaggio

I linguaggi di programmazione maggiormente utilizzati nell’ambito dei giochi sono: C#, Java e Python, c’è, tuttavia, un’ampia possibilità di scelta per quanto riguarda la tecnologia da usare. La scelta in questo caso è ricaduta su C#, in quanto particolarmente indicato per la creazione di giochi in 2D.

## Interfaccia grafica

Grazie alla scelta di C# è possibile utilizzare Microsoft Forms, che mette a disposizione parecchi strumenti per la creazione di interfacce grafiche. L’interfaccia grafica è pensata in maniera che sia semplice ed intuitiva per tutti e per tutte le età,