Gioco di carte con intelligenza artificiale

Titolo del progetto: Gioco di carte con intelligenza artificiale
Alunno: Alessandro Colugnat
Classe: Info 4AC
Anno scolastico: 2018/2019
Docente responsabile: Ugo Bernasconi

1 Introduzione

1.1 Informazioni sul progetto

- Alessandro Colugnat – Allievo
- Ugo Bernasconi – Docente Responsabile
- Scuola Arti e Mestieri Trevano, Informatica
- Data inizio: 08.01.2019
- Data fine: 10.04.2019

1.2 Abstract

The job of this project is to create a robotic arm who plays card game called UNO and he plays with its own intelligence.

1.3 Scopo

Lo scopo del progetto è quello di creare un'intelligenza artificiale che gioca a UNO e avrà una intelligenza che trova tutte le mosse possibili basandosi sulla carta che si trova nel campo da gioco e infine in maniera casuale sceglierà la mossa da utilizzare, si deve implementare anche una videocamera che permette di riconoscere le carte di gioco e verificare che tutte le regole siano state applicate correttamente, per fare in modo che quando si gioca il computer riconosce le carte ed elabora la mossa che permette di battere l'avversario, verrà anche implementato un braccio robotico per poter fare in modo che il computer riesca a giocare con le proprie carte a livello fisico.
Questa documentazione serve a introdurre gli utenti alla creazione del programma, insieme anche al suo scopo e anche del motivo per cui è stato creato.
Serve anche a introdurre come è stata programmata l'intelligenza artificiale e come la macchina riesca ad auto apprendere le varie mosse che servono a vincere, come fa a scegliere autonomamente quale strategia da utilizzare durante la partita, la documentazione mostra anche come è stato reso possibile il riconoscimento delle carte tramite computer e come verificare che la carta riconosciuta sia la stessa che è stata messa nel campo da gioco, fa vedere anche come la macchina riesca a gestire un braccio robotico che gestisce i movimenti scelti dall'intelligenza artificiale.

2 Analisi

2.1 Analisi del dominio

Questo progetto verrà utilizzato ogni volta che si vuole fare una partita a carte, un prodotto del genere esiste già e sono dei robot che giocano a scacchi oppure qualche altro gioco di carte, ma su internet non ho visto nessuno che ha costruito un robot con intelligenza artificiale che gioca a UNO.

2.2 Analisi dei costi

Componenti	Prezzo
1 Lavoratore	32 CHF/ ora

Prezzo totale per 160 ore: 5120 CHF

2.3 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	Creazione riconoscimento telecamera per carta in gioco
Priorità	1
Note	Si necessità di una buona videocamera con alta risoluzione per avere migliori prestazioni.
Sotto requisiti	
001	Si deve collegare la telecamera al computer
002	Prendere la carta e metterla davanti alla telecamera
003	Riconoscere il colore della carta
004	Riconoscere il numero della carta

ID: REQ-02	
Nome	Creazione riconoscimento telecamera per carta in possesso
Priorità	1
Note	Si necessità di una buona videocamera con alta risoluzione per avere migliori prestazioni.
Sotto requisiti	
001	Si deve collegare la telecamera al computer (oppure la telecamera del pc)
002	Vedere il colore della carta pescata
003	Vedere il numero della carta pescata
004	Salvare la posizione della carta

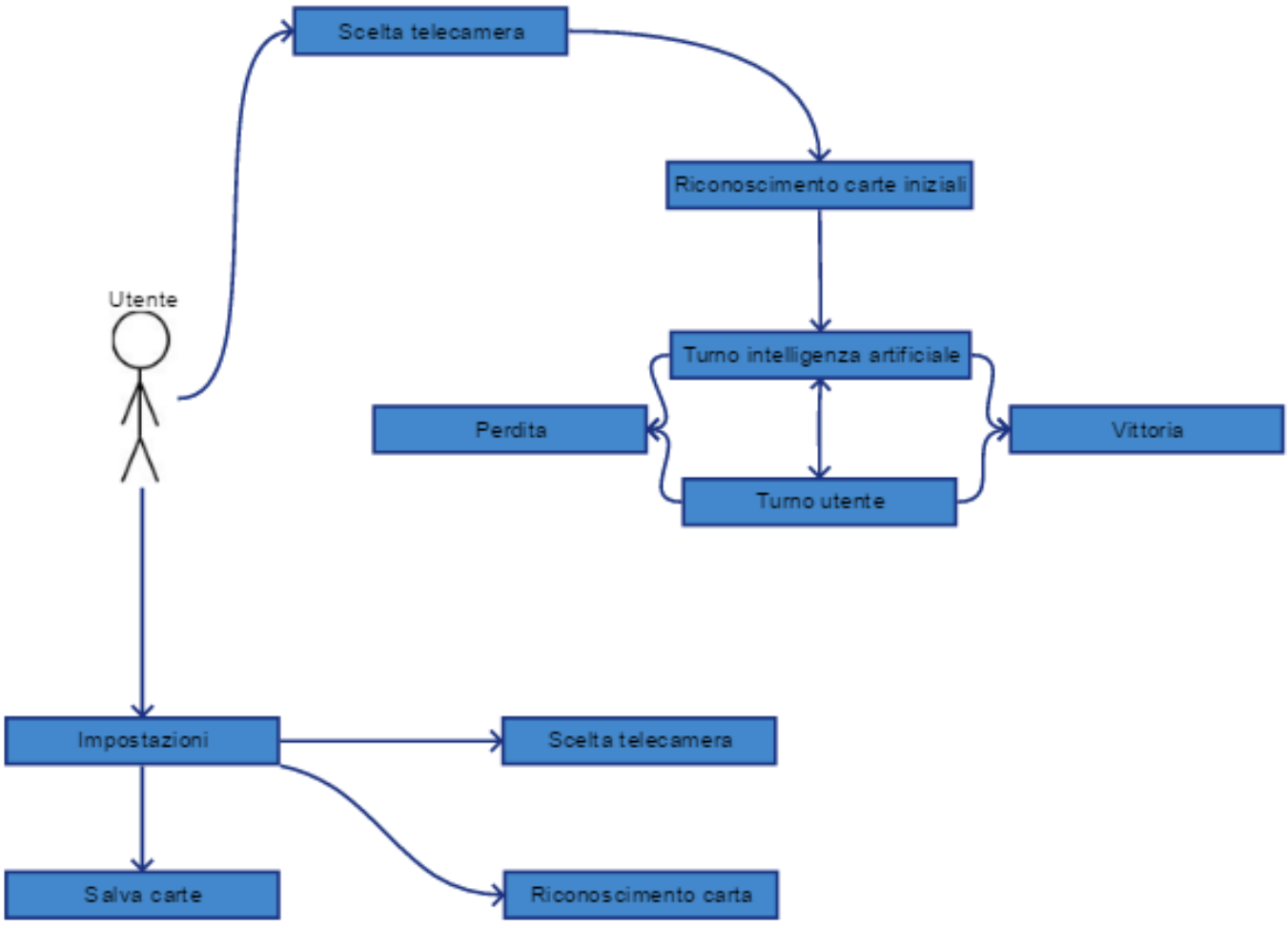
ID: REQ-03	
Nome	Creazione IA
Priorità	1
Sotto requisiti	
001	Controllare numero o colore delle carte IA siano uguale alla carta in gioco
002	Controllare se si possono fare altre combinazioni con le carte rimanenti
003	Scelta della mossa migliore tramite una percentuale di successo
004	Eseguire la mossa

ID: REQ-04	
Nome	Movimento braccio elettronico per mossa della carta
Priorità	1
Note	Si necessità di un braccio con minimo 5DOF
Sotto requisiti	
001	Aspettare la scelta della mossa dell'IA
002	Avere le coordinate della posizione della carta
003	Muovere il braccio nella posizione corretta e prendere la carta
004	Trasportare carta nella sezione della carte in gioco
005	Lasciare la carta e tornare in posizione iniziale

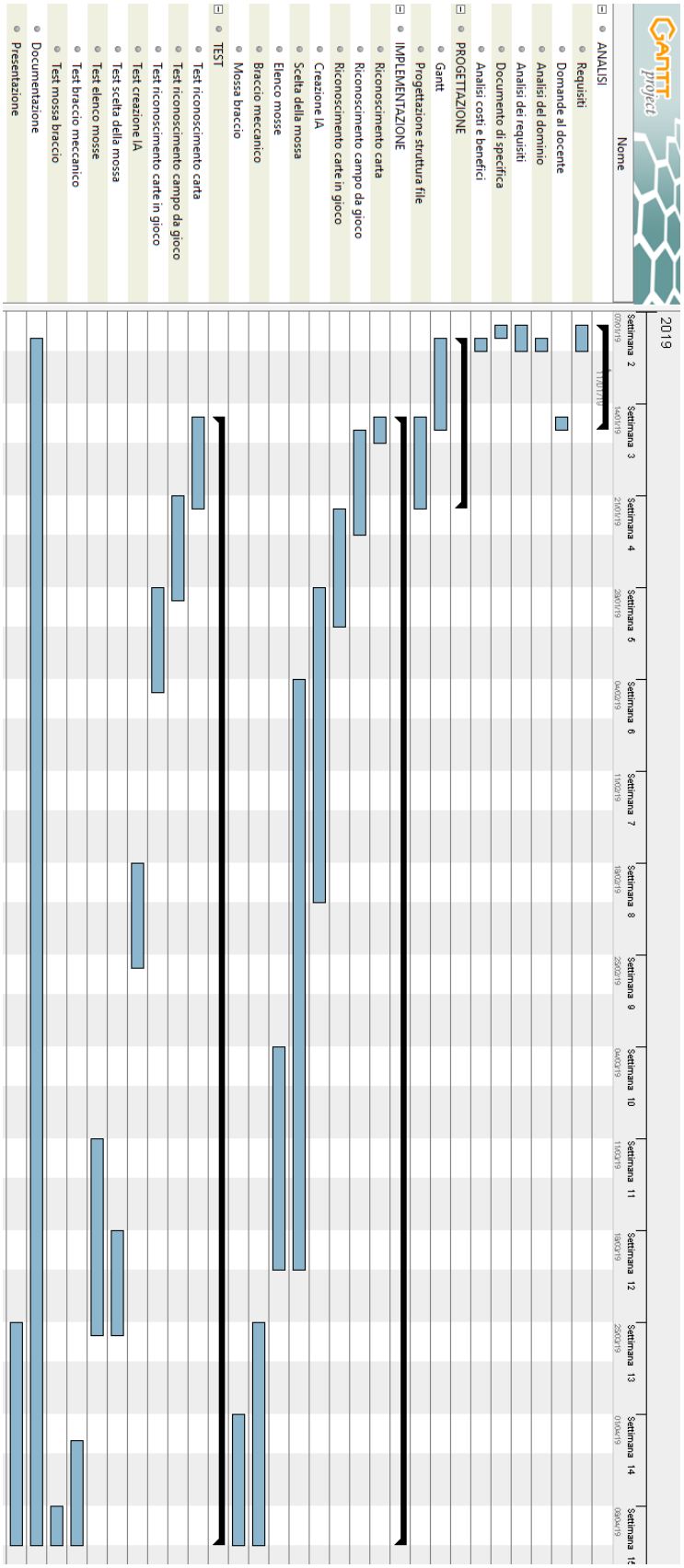
ID: REQ-05	
Nome	Movimento braccio elettronico per pescare la carta
Priorità	1
Note	Si necessità di un braccio con minimo 5DOF
Sotto requisiti	
001	Prendere le coordinate del mazzo di carte
002	Avvicinare il braccio elettronico al mazzo
003	Appiccicare la carta alla mano
004	Riconoscere la carta con la telecamera
005	Mettere la carta in posizione
006	Salvare le coordinate della carta nuova

2.4 Use case

Qui rappresentato sotto e lo schema di cosa può fare un'utente, per questo progetto non esistono amministratori o responsabili, esiste solamente l'utente che vuole fare una partita. L'utente può anche fare manutenzione delle carte che sono registrare nel database, per cercare di aggiungere o migliorare le carte che sono registrate nel database.



2.5 Pianificazione



Nella parte dell'implementazione ho scelto di mettere più importanza al riconoscimento delle carte e della intelligenza artificiale perché sono le parti principali del progetto, invece il braccio meccanico è secondario.

2.6 Analisi dei mezzi

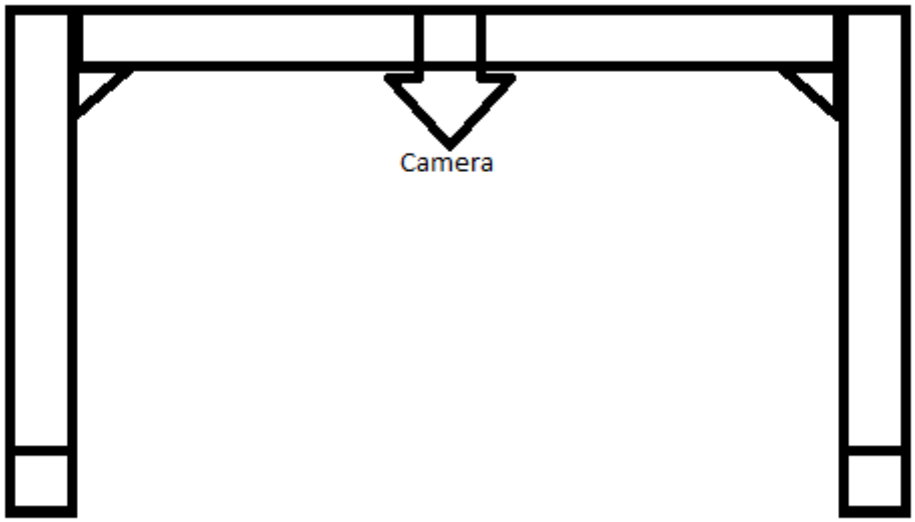
2.6.1 Software

- Visual studio 2017
- EmGu CV / AForge.NET

2.6.2 Hardware

- 1 PC portatile (Windows 10)
- Braccio con 5 DOF Arduino
- Raspberry 3B
- Telecamera

In questo progetto sono a disposizione un braccio meccanico e un raspberry ma nel proseguimento dell'implementazione non sono riuscito a mettere in funzione il braccio e il raspberry per motivi di tempo, la complicazione è avvenuta con il riconoscimento delle carte con la telecamera ma senza nessun successo, per quanto riguarda il software non sono stati implementati le librerie per il riconoscimento delle carte essendo inutili allo scopo perché non funzionavano correttamente, per risolvere il problema ho implementato un nuovo programma che non utilizza librerie esterne e quindi non sono stati utilizzati librerie esterne richieste dal progetto.
Per la telecamera è stata costruita una struttura che permettesse di tenere la telecamera attaccata verso la l'alto che punta in basso, per fare in modo che si possano mettere le telecamere nel campo da gioco. La struttura è la seguente:



La struttura è fatta in ferro e sono stati utilizzati componenti forniti dalla scuola d'arti e mestieri di Trevano, i pezzi sono stati montati insieme.

3 Progettazione

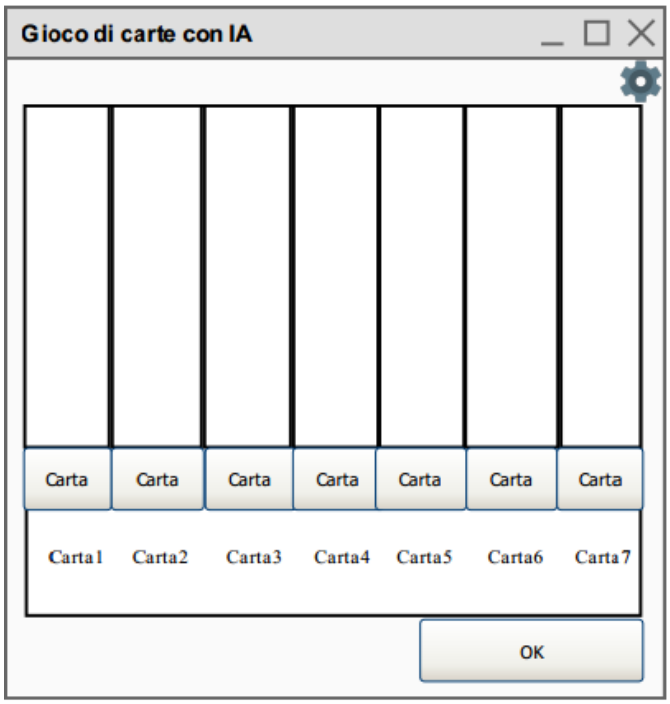
3.1 Design delle interfacce

Per le interfacce è stato utilizzato il programma Pencil per fare in modo di creare delle interfacce semplici, che siano comprendibili da tutti gli utenti, sono state utilizzate 5 interfacce in totale, per andare in avanti nelle interfacce si può trovare in basso il bottone “OK” che permette di andare nella pagina seguente, ma facendo che tutti i requisiti siano soddisfatti nei controlli delle singole interfacce. Esiste una interfaccia uguali per tutti che è la finestra principale:

3.1.1 Design scelta della telecamera

In questa interfaccia si farà la scelta di quale telecamera si vorrà utilizzare per il riconoscimento delle carte. Verrà mostrata l’immagine di ciò che la telecamera sta vedendo per fare in che se la telecamera è sbagliata si possa selezionarne una nuova.

3.1.2 Design della scelta delle prime carte

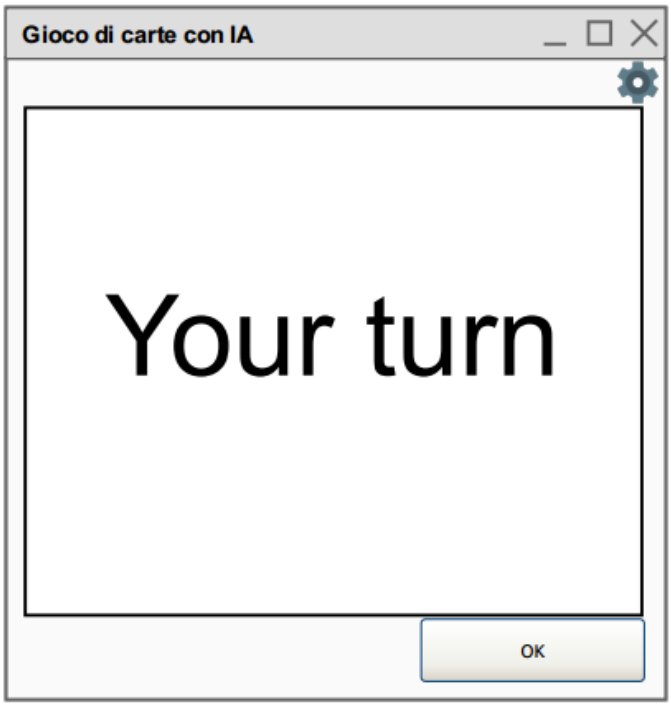


La scelta delle carte serve per riconoscere con quale mazzo iniziale ha l'intelligenza artificiale per quando si inizia un gioco, le carte da mettere sono 7, seguendo il regolamento di UNO.

3.1.3 Design della pagina di gioco dell'IA

In questa sezione si vede quando è il turno dell'intelligenza artificiale, fa vedere la carta che ha riconosciuto e poi con il tasto che si trova sotto si può fare un'altra verifica della carta che è stata trovata, per fare in modo che il riconoscimento delle carte finisca con il risultato giusto.

3.1.4 Design pagina di gioco del giocatore

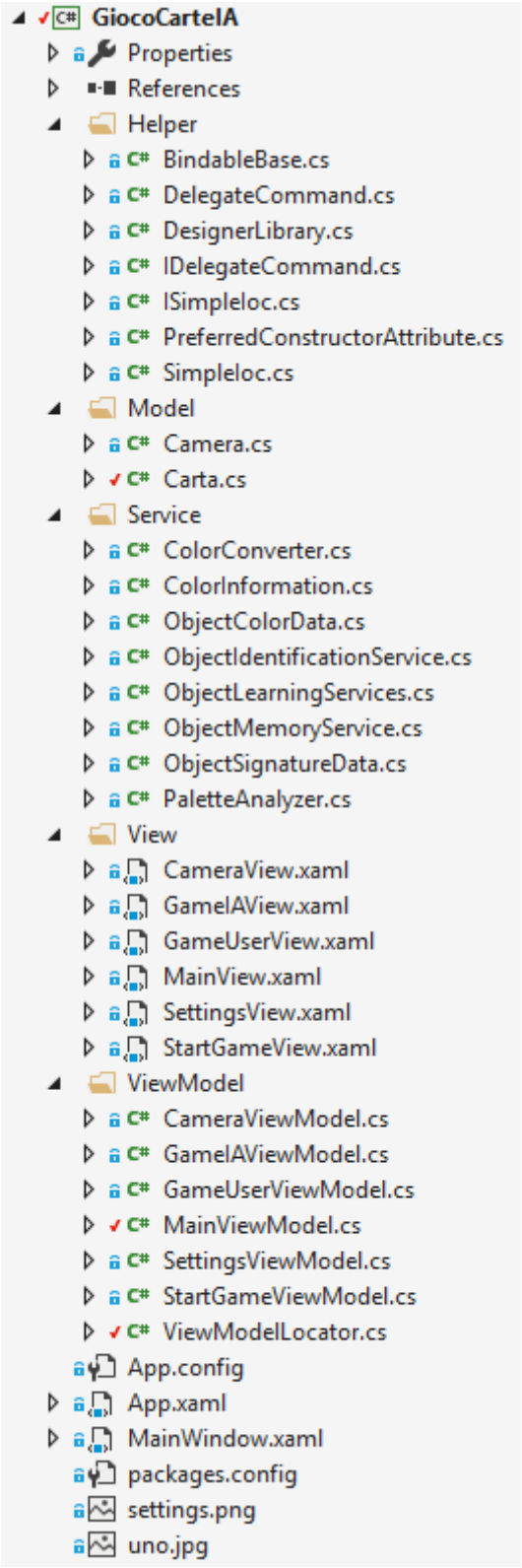


Questo è la pagina del turno del giocatore, non fa niente di speciale, serve per notificare che il giocatore deve fare la sua mossa, per poi passare di nuovo alla pagina di gioco dell'intelligenza artificiale.

3.1.5 Design pagina delle impostazioni

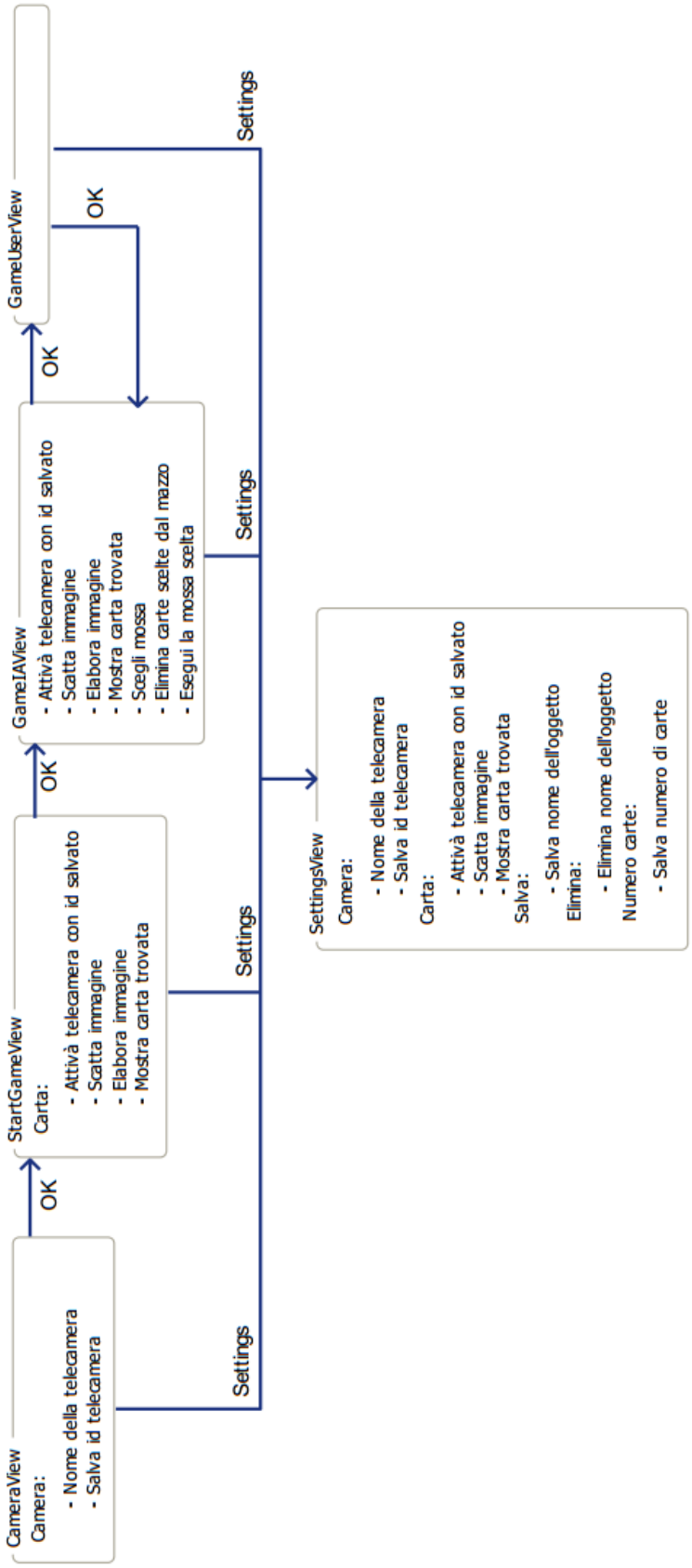
La pagina delle impostazioni contiene la telecamera che si può cambiare e anche la sezione per riconoscere nuove carte oppure eliminare le carte precedentemente salvate.

3.2 Design delle cartelle



Le cartelle utilizzate sono le stesse per MVVM ma viene aggiunta la cartella Helper e Service per contenere tutte le impostazioni riguardanti hai codici di altri utenti, alla fine i file di viewmodel non sono serviti totalmente, perché sono serviti solamente da collegamento per il passaggio delle pagine. Ma tutto il codice si trova direttamente nelle View e non è corretto ma per motivi che verranno spiegato nella parte di implementazione, sono stati scelti altri metodi.

3.3 Design delle classi



Questo rappresenta lo schema di funzionamento delle varie classi e ciò che devono fare, viene indicato anche quale passaggio si deve fare per proseguire di classe, le frecce indicano dei pulsanti, le immagini sono la parte fondamentale per il funzionamento totale del progetto.

4 Implementazione

4.1 Intelligenza artificiale

Come prima parte dell'implementazione verrà spiegata la logica di come funziona l'intero programma dell'intelligenza artificiale, il codice è corto ma da comprendere è molto complesso perché per fare questo pezzo di codice serviva pensare a ciò che fa il codice nei vari possibili casi, ho creato un algoritmo che permette di vedere tutte le possibili mosse:

```
private void SelectGame(String selectedCard, String selectedNum, bool nextIsMy)
{
 for(int i = 0; i < cardAICopy.Length; i++)
 {
  if(((cardAICopy[i][0] == selectedCard) && !nextIsMy) || (cardAICopy[i][1] == selectedNum))
  {
  Console.WriteLine(AddTab(true)+cardAICopy[i][0]+" "+cardAICopy[i][1]+Environment.NewLine);
   cardInGameCopy[0] = cardAICopy[i][0];
   cardInGameCopy[1] = cardAICopy[i][1];
   combinations.Add(cardInGameCopy[0] + " " + cardInGameCopy[1]);
   cardAICopy[i][0] = "";
   cardAICopy[i][1] = "";
   SelectGame(cardInGameCopy[0], cardInGameCopy[1], true);
   cardAICopy[i][0] = cardAI[i][0];
   cardAICopy[i][1] = cardAI[i][1];
   AddTab(false);
  }
 }
 Save();
 combinations.RemoveRange(0, combinations.Count);
}
```

Questo mostrato qui sopra è l'intero codice che permette di far mostrare tutte le mosse possibili con le carte che si ha in mano. Ora spiegherò la logica del funzionamento del codice.
Ci sono delle variabile che sono degli Array multidimensionali che vengono utilizzati in questo modo:
cardAICopy[][] → la prima parte dell'array e utilizzato per la carta, invece il secondo contiene il colore nell'indice 0 e invece il numero nell'indice 1.
Ci sono anche degli Array:
cardInGameCopy[] → nel primo indice si trova il colore della carta e invece nel secondo indice si trovava il numero della carta.
Ci sono anche delle variabili all'interno del codice:
selectedCard → il colore della carta.
selectedNum → il numero della carta.
nextIsMy → una variabile booleana.
Tutti queste variabili verranno spiegate nella documentazione sottostante e del motivo per qui sono state create.

4.1.1 Ciclo

La prima parte che voglio introdurre è ii ciclo for, viene utilizzato semplicemente per far passare tutte le carte al controllo che verrà usato nel passaggio successivo, ho preferito utilizzare il ciclo for perché come ciclo verrà creata la variabile i che sarà molto importante alla fine del ciclo, il for verrà richiamato più volte a se stesso perché come verrà mostrato nella documentazione ci sarà il metodo stesso che verrà richiamato, quindi tutto ciò sarà necessario per controllare delle carte del proprio mazzo con altre carte che provengono dallo stesso mazzo.

4.1.2 Condizione

Nella condizione subito dopo il for vengono controllate due condizioni molto importanti, che sono quelli del numero della carta e del colore della carta che viene utilizzata nel primo turno, tutto questo viene fatto per ogni carta che si possiede, tutto questo grazie al ciclo, e molto importante perché se non si gestisce bene questa condizione si rischia di creare della chiamate infinite ed inutili a se stesso e facendo così si creerebbe un ciclo infinito da bloccare il programma e far lavorare il computer in maniera inutile e pesante, ma con l'if che ho pensato riesce a gestire questi problemi e far riuscire a trovare una soluzione per il programma. Nelle condizione si può prendere il primo pezzo:

```
(cardAICopy[i][0] == selectedCard)
```

In questo pezzo di codice si fa il controllo delle due carte che siano dello stesso colore, questa regola del colore può essere applicata solamente con la carta che si trova in mezzo al campo. Infatti davanti a questo codice si può vedere che c'è stata l'aggiunta di una condizione molto importante:

```
((cardAICopy[i][0] == selectedCard) && !nextIsMy)
```

Si può notare la variabile che si chiama nextIsMy questa variabile booleana significa che la prossima carta è la mia, per spiegare meglio significa che se la carta in gioco è quella della IA, nella prossima mossa può decidere di fare un'altra mossa ma senza l'utilizzo del colore, perché secondo le regole, se vuoi mettere più di una carta per terra si dovrà prendere solamente il segno uguale e non il colore, e questa variabile fa questo tipo di controllo, infatti nella prossima condizione non si verrà aggiunta nessuna variabile nextIsMy:

```
(cardAICopy[i][1] == selectedNum)
```

In questo pezzo di codice viene fatto solamente il controllo dei numeri della carta in gioco con la carta posseduta. Tutto questo viene sommato con una porta OR per fare in modo che nel caso nextIsMy sia a false, la condizione possa comunque venire assolta grazie al numero della carta:

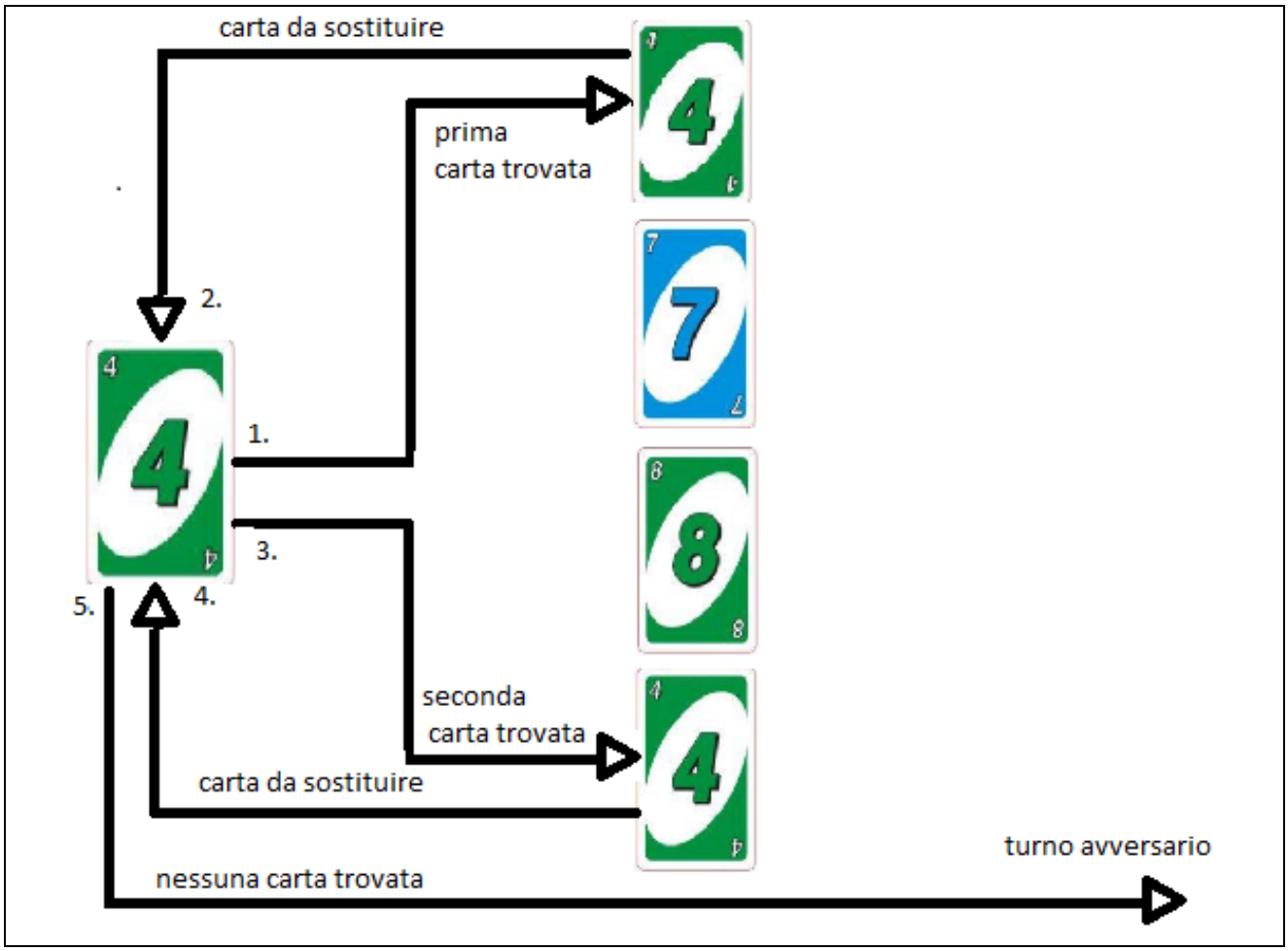
```
if(((cardAICopy[i][0] == selectedCard) && !nextIsMy) || (cardAICopy[i][1] == selectedNum))
```

4.1.2.1 Condizione falsa

Nel caso in cui i requisiti non soddisfacessero la condizione del metodo, non sarà causato alcun cambiamento nel metodo, ma passera alla carta successiva, fino a quando non ci saranno più carte da scegliere e in quel caso, finirà il ciclo delle carte.

4.1.2.2 Condizione vera

Se la condizione soddisfa i criteri che sono stati spiegati soprastante, allora inizierà, l'operazione più impegnativa che è quella di mostrare la carta scelta e richiamare il metodo stesso per fare di nuovo il controllo se si possono aggiungere nuove carte alla mossa. La logica del codice permette alla IA di poter scegliere tutte le mosse possibili, così facendo potrà scegliere la mossa migliore per fare in modo che l'avversario rimanga bloccato per poi essere sconfitto, nel codice viene richiamata lo stesso metodo che permette di controllare più volte la carta giocata, potrebbe crearsi un loop infinito se non si gestisce bene questa chiamata del metodo, in parte il problema è risolto grazie alla condizione ma il problema e soprattutto risolto grazie alla rimozione della carta trovata, così facendo non si creano continuamente chiamate del metodo con la stessa carta, per fare in questo modo, si deve togliere la carta prima della chiamata del metodo. Per questo motivo vengono utilizzate delle variabili che hanno copiato i dati dalla variabile madre, per fare in modo che venga cancellata la carta sul momento, ma il mazzo originale non viene toccato, appena si richiama il metodo vengono richieste tre valori, il colore, il numero e se la carta è quella che gioca la IA, così cambia la condizione dell'if. La logica del metodo è spiegata in maniera semplificata nello schema riportato qui sotto:



In questo schema viene mostrato il caso in cui ci fossero dei numeri uguali nella propria mano, si potrebbe fare la combinazione con il numero 8 verde, ma non ci sarebbe la possibilità di aggiungere il 4, per fare una combinazione maggiore.

4.1.3 Salvataggio combinazione di carte

Appena finisce il ciclo delle carte, verrà richiamato un metodo che si chiama **Save()**, questo metodo permette di salvare le combinazioni trovate precedentemente, contiene queste informazioni:

```
private void Save()
{
 if(combinations.Count !=  0)
 {
  addCard += 1;
  saveCombinations[addCard - 1] = combinations.ToList();
 }
}
```

Questo metodo è molto semplice, perché viene aggiunta una combinazione ogni volta che finisce il metodo **SelectGame()**, perché se il metodo finisce significa che la combinazione è arrivata al termine, nel codice si può trovare la variabile **combinations** che serve a salvare la combinazione ogni volta che se ne trova una, e alla fine verrà salvata in un array, che poi sarà utilizzata in futuro per scegliere la mossa in maniera completamente casuale. Viene fatto un controllo nel metodo perché nel caso non ci fossero combinazione non si dovrebbe riempire inutilmente l'intero array. Alla fine la lista verrà completamente pulita per poi contenere una nuova combinazione:

```
combinations.RemoveRange(0, combinations.Count);
```

All'interno della lista verrà salvata come stringa, per poi alla fine quando si dovrà svolgere la mossa, si dovrà dividere la stringa e prendere i due valori principali che sono il colore e il numero:

```
combinations.Add(cardInGameCopy[0] + " " + cardInGameCopy[1]);
```

Una volta che sono state salvate tutte le combinazioni, dovrà essere scelte una di quelle mosse per farlo ho scelto di fare un numero casuale per scegliere la mossa così potrà fare tutti i tipi di mossa è non solo quelle in cui puoi mettere più carte alla volta:

```
Random num = new Random();
int value = num.Next(0, saveCombinations.Length);
saveCombinations[num.Next(0, saveCombinations.Length)];
```

Quando si trova la mossa verrà fatto in modo che le carte scelte vengano eliminate dall'array che contiene tutte le carte possedute dalla IA. Si utilizza un numero casuale che si trova dallo 0 fino al numero masso delle combinazioni che si possono fare. Infine la lista che contiene tutte le combinazioni verrà pulita per fare in modo che potrà venire utilizzato un'altra volta:

```
saveCombinations = null;
```

4.1.4 Aggiunta tabulazione

Per semplicità, nel codice viene aggiunta una tabulazione per vedere in maniera più veloce le varie combinazione trovate dalla IA, questa parte serve ai programmatori per vedere se il codice funziona come si deve. I dati vengono stampati a terminale:

```
Console.WriteLine(AddTab(true)+cardAICopy[i][0]+" "+cardAICopy[i][1]+Environment.NewLine);
```

Nel caso in cui il richiamo del metodo ha come valore true, verrà aggiunta una tabulazione, invece se è false verrà tolta. Così non verrà aggiunta continuamente una tabulazione. Il metodo è composto in questo modo:

```
private String AddTab(bool remove)
{
 tab = "";
 if (remove)
  countTab += 1;
 else
  countTab -= 1;
 for(int i = 0; i < countTab; i++)
 {
  tab += " L ";
 }
 return tab;
}
```

Questo fa vedere come aggiungere una tabulazione, come tabulazione è stata scelta una lettera che sembra una tabulazione e abbiamo utilizzato un l in maiuscolo, questo serve per mettere in risalto le sequenze delle carte scelte. Nel caso si aggiunge una tabulazione è perché dopo la cartella nel gioco si mette un'altra carta, questo fa un effetto a piramide:

```
Rosso 1
 L Blu 1
   L Giallo 1
 L Giallo 1
   L Blu 1
Rosso 4
 L Rosso 4
```

In questo caso si può vedere a cosa funziona la tabulazione, ora si può vedere che la prima combinazione di carte e composta da tre passaggi e invece la seconda possibilità di combinazione contiene due passaggi, questo farà in modo che l'utente riesca a vedere in maniera migliore cosa stia accadendo con l'algoritmo per fare una mossa. Questo codice può essere utile se si vuole mostrare le carte della combinazione scelta, per fare vedere al giocatore con quale carta la IA farà la mossa.

4.1.5 Eliminare le carte utilizzate

Dopo che la mossa è stata eseguita si possono eliminare le carte che erano nel mazzo della IA:

```
private void DeleteCard(string delete)
{
 for(int i = 0; i > Carta.CardAI.Length; i++)
 {
  if ((Carta.CardAI[i][0] + " " + Carta.CardAI[i][1]) == delete)
  {
   Carta.CardAI[i][0] = "";
   Carta.CardAI[i][1] = "";
   Carta.NumCard -= 1;
   break;
  }
 }
}
```

Il codice è molto semplice perché serve solamente a cancellare il dato all'interno dell'array che contiene tutte le carte che possiede l'intelligenza artificiale. Alla fine rimarrà un buco nell'array che questo permetterà di essere riempito di nuovo nel caso in cui si pesca una nuova carta. Nel for viene passato come lunghezza del ciclo e la grandezza dell'array di CardAI che contiene 76 spazi, il motivo per cui si hanno 76 spazi all'interno dell'array e per fare in modo che in ogni caso si possano tenere tutte le carte del mazzo in mano e sono state tolte le carte speciali perché il programma gestisce soltanto i numeri. Quindi il for farà 76 cicli per controllare ogni possibilità che si abbia una carta, nel ciclo verrà fatto un controllo che permette di vedere se la carta selezionata nel ciclo sia identica alla carta che si deve eliminare. Appena si entra nella condizione si verranno cancellati i dati all'interno dell'array e decrementare una variabile che si spiegherà più in avanti. E alla fine si esce dal ciclo perché è inutile continuare a cercare altre carte dello stesso valore se la carta giocata è una soltanto.
La variabile NumCard serve a vedere quante carte rimangono all'intelligenza artificiale per fare in modo di vedere se si deve chiamare UNO oppure vedere se si ha vinto:

```
foreach (string comb in saveCombinations[num.Next(0, saveCombinations.Length)])
{
 Thread.Sleep(1000);
 Move.Content = comb;
 DeleteCard(comb);
 if (Carta.NumCard == 1)
 {
  Thread.Sleep(2000);
  Move.Content = "UNO!";
 }
 else if (Carta.NumCard == 0)
 {
  Move.Content = "!!!!!! I WIN !!!!!!";
 }
}
```

In questo pezzo di codice si vede l'utilità della variabile NumCard e si vede anche che processi si fanno per quando si avvia una mossa. Come primo passaggio si deve fare un foreach che fa passare tutte le carte che sono state trovate per la mossa che si è scelto dall'algoritmo descritto in precedenza. Come secondo passo ho fatto in modo che il programma venga fermato per 1 secondo, perché ogni volta che fa il foreach si deve mostrare quale carta si utilizza per la mossa, dopo si chiama il metodo che permette di eliminare la carta dall'array, dopo che si è eliminata la carta si controlla che la prossima carta sia la penultima e nel caso si scrive UNO! oppure se è l'ultima si deve scrivere !!!!!! I WIN !!!!!! queste informazioni servono per notificare l'avversario su quante carte possiede ancora la IA.

4.1.6 Pescare una nuova carta

Per il metodo che serve per il pescaggio della carte si deve fare come prima cosa, un controllo che permette di verificare se la macchina deve pescare delle carte. Dato che non ci sono carte speciali che ti fanno pescare, l'unico modo per pescare una nuova carta e fare in modo che l'intelligenza artificiale non abbia più mosse disponibili:

```
private void IsDrawCard()
{
 Draw.IsEnabled = true;
}
```

Questo metodo viene richiamato nel caso in cui non ci fossero mosse disponibili:

```
if(saveCombinations.Length != 0)
{
 [...]
}
else
{
 Move.Content = "DRAW";
 IsDrawCard();
}
```

Il pulsante viene abilitato nel caso in cui fosse disponibile pescare, il pescaggio viene fatto una volta sola, e appena si pesca la carta si fa di nuovo il controllo se è possibile mettere una carta in cima al mazzo:

```
private void Draw_Click(object sender, RoutedEventArgs e)
{
 Carta.NumCard += 1;
 var card = IdentifierObject().Split(' ');
 Carta.CardAI[Carta.NumCard][0] = card[0];
 Carta.CardAI[Carta.NumCard][1] = card[1];
 Carta.CardAICopy = (String[][])Carta.CardAI.Clone();
 SelectGame(Carta.CardInGameCopy[0], Carta.CardInGameCopy[1], false);
}
```

Come procedimento è lo stesso che si è utilizzato per fare il pescaggio delle carte iniziali, ma a differenza del metodo utilizzato precedentemente, viene aggiunto anche il fatto che le carte devono essere utilizzate per fare la mossa un'altra volta, come si può vedere viene richiamato il metodo per fare la mossa, nel caso in cui non si potesse fare ancora la mossa, l'utente può scegliere in base alle proprie regole se l'utente vuole fare in modo che la carta pescata sia solo una per turno, oppure se si vuole pescare fino a quando si riesce a fare una mossa.

4.2 Riconoscimento delle carte

Per riconoscere le carte si utilizza un codice offline perché non si vogliono fare richieste verso l'esterno, per fare in modo che si possa utilizzare anche se non si ha una connessione verso l'esterno, per fare in modo che il codice del riconoscimento delle carte sia senza connessione verso l'esterno si devono prendere dei file che contengono le funzioni per il riconoscimento da internet, quindi il codice per le classi non verrà spiegato, invece verrà spiegato il codice per utilizzare quelle classi, sono semplici ma devono avere comunque un controllo per fare in modo che funzionino:

```
webCameraControl.StartCapture(Camera.CameraId);
identificationImage = webCameraControl.GetCurrentImage();
webCameraControl.StopCapture();
if (identificationImage == null)
{
 MessageBox.Show("Please select an image to analyze");
}
else
{
 IList<string> identifiedObjects =
                    ObjectIdentificationService.AnalyzeImage(Convert(identificationImage));
 if (identifiedObjects.Count == 0)
 {
  FoundedCard.Content = "Not work";
 }
 else
 {
  var objectCard = identifiedObjects[0].Split(' ');
  Carta.CardInGame[0] = objectCard[0];
  Carta.CardInGame[1] = objectCard[1];
  FoundedCard.Content = objectCard[0] + " " + objectCard[1];
  Carta.CardInGameCopy = (String[])Carta.CardInGame.Clone();
  SelectGame(Carta.CardInGameCopy[0], Carta.CardInGameCopy[1], false);
  [...]
 }
 saveCombinations = null;
}
```

Questo è il codice che si utilizza per fare il riconoscimento delle carte e c'è anche il codice per l'utilizzo della telecamera e anche il codice per prendere la carta trovata e metterla nel algoritmo per sapere quale sia la mossa migliore.

4.2.1 Utilizzo videocamera

Per utilizzare la telecamera servono solamente quattro comandi importanti che sono quelli per accendere, spegnere, catturare l'immagine e la funzione per prendere tutte le telecamere disponibili dal computer:

```
webCameraControl.StartCapture(Camera.CameraId);
identificationImage = webCameraControl.GetCurrentImage();
webCameraControl.StopCapture();
```

Il primo comando serve per iniziare a far partire la telecamera, per farlo si deve passare un'id che permette di sapere quale telecamera può essere attivata. Invece il secondo comando serve per prendere una immagine di cosa sta riprendendo la telecamera nel momento in cui si chiama il metodo. Invece il secondo codice serve fermare la telecamera e si deve mettere ogni volta, perché se si vuole accendere la telecamera di nuovo si deve prima spegnere la sessione precedente se no da errori.

4.2.2 Riconoscere una carta

Per riconoscere la carta serve un metodo che permette di analizzare un'immagine e controllare in un database tra i le varie carte quale è la carta più identica a ciò che si trova nel gioco:

```
IList<string> identifiedObjects =
ObjectIdentificationService.AnalyzeImage(Convert(identificationImage));
```

Come si può notare il risultato tirato fuori dal metodo viene messo in una lista di stringa, il motivo e perché quando si fa partire il metodo vengono salvate tutte le carte possibili per la carta che si trova in gioco, ma alla fine viene vista solamente la prima carta che è stata trovata, se è sbagliato nel programma è stato messo un bottone che permette di rifare il controllo delle varie macchine.

4.2.3 Conversione immagine

Nel metodo per riconoscere una carta viene richiamato un altro metodo che permette di convertire un'immagine bitmap in una immagine bitmapImage, perché nel metodo viene richiesto di utilizzare una variabile di quel tipo, con la telecamera possono essere salvate immagine di tipo bitmap e non è ciò che serve al metodo per riconoscere le carte:

```
private BitmapImage Convert(System.Drawing.Image img)
{
 using (var memory = new MemoryStream())
 {
  img.Save(memory, ImageFormat.Png);
  memory.Position = 0;
  var bitmapImage = new BitmapImage();
  bitmapImage.BeginInit();
  bitmapImage.StreamSource = memory;
  bitmapImage.CacheOption = BitmapCacheOption.OnLoad;
  bitmapImage.EndInit();
  return bitmapImage;
 }
}
```

Come prima operazione si deve creare una nuova memoria in cui si salverà l'immagine all'interno per poi poterla lavorare, come secondo passo si può andare a salvare l'immagine che si vuole convertire. Poi si deve mettere la memoria in prima posizione così si è sicuri che si andrà a lavorare sul dato corretto, e adesso arriva la parte in cui si deve creare un oggetto di tipo bitmapImage che verrà utilizzata per metterci all'interno la nuova immagine convertita, nei prossimi comandi servono per prendere l'immagine nella memoria e mettere l'immagine all'interno della variabile precedentemente creata. E alla fine c'è un ritorno della nuova immagine convertita.

4.2.4 Comandi per salvare carta da riconoscere

Quando si ha la carta riconosciuta si dovrà trasformare un array dividendo il numero e il colore, per farlo si può utilizzare un semplice split che divide gli spazi:

```
var objectCard = identifiedObjects[0].Split(' ');
Carta.CardInGame[0] = objectCard[0];
Carta.CardInGame[1] = objectCard[1];
SelectGame(Carta.CardInGameCopy[0], Carta.CardInGameCopy[1], false);
```

Dopo che si è diviso il numero dal colore si può chiamare il metodo che permette di far partire l'algoritmo per trovare le combinazione per la mossa che deve fare la IA.

4.3 Utilizzo delle interfacce

Per questo progetto sono state utilizzate 5 interfacce, per passare ad ogni interfaccia si devono fare dei controlli e si devono passare vari dati, in questo capitolo verranno spiegate le tecniche utilizzate per fare in modo che il programma funzioni correttamente. Tutte le funzioni sono state fatte all'interno delle interfacce perché i model non funzionano correttamente, quindi come soluzione e stata trovata la possibilità di utilizzare le classi delle interfacce senza utilizzare i model, viene comunque utilizzato il ViewModelLocator per fare in modo che le pagine possano essere cambiate.

4.3.1 Passaggio dei dati

Il passaggio di dati viene attraverso una classe esterna per fare in modo che i dati si visualizzabili da tutti, non sono stati fatti controlli per la sicurezza dei dati, essendo il prodotto senza connessione verso l'esterno, questi sono i dati all'interno del file per quanto riguarda la telecamera:

```
public static class Camera
{
 public static WebEye.Controls.Wpf.WebCameraId CameraId { get; set; }
}
```

Invece questa è la classe che è stata utilizzata per tenere i dati delle carte:

```
public static class Carta
{
 public static String[] CardInGame { get; set; }
 public static String[] CardInGameCopy { get; set; }
 public static String[][] CardAI { get; set; }
 public static String[][] CardAICopy { get; set; }
 public static int NumCard { get; set; }
}
```

Come si può notare sono state utilizzate delle classi statici per il passaggio di dati, questo per fare in modo che quando si utilizzano le classi vengano presi i dati senza dovere creare un nuovo oggetto che non contiene lo stesso tipo di dati all'interno delle variabili, invece con le classi statici posso passare i dati da pagina all'altra senza perdere alcun tipo di dato.
Sono utilizzati vari tipi di attributi che ora verranno spiegati:

```
public static WebEye.Controls.Wpf.WebCameraId CameraId { get; set; }
```

Questo attributo serve per tenere salvato quale telecamera è stata scelta, per fare in modo che quando si cambi pagina non si debba scegliere ogni volta quale telecamera si vuole utilizzare invece facendo in questo modo si riesce a prendere la telecamera che è stata scelta nella prima interfaccia.

```
public static String[] CardInGame { get; set; }
```

Serve a tenere salvato quale carta il si trova nel gioco e con quale il computer si deve riferire per poter fare la nuova mossa, verrà cambiato ogni inizio turno e anche quando la IA dovrà vedere quale carta è da mettere come prossima.

```
public static String[] CardInGameCopy { get; set; }
```

Questo è una copia dell'attributo spiegato prima, serve per tenere salvato la prima carta che si trova in cima nel mazzo all'inizio del turno perché nel caso la IA abbia qualche problema nel riconoscere la carta e l'utente gli dice di ripetere l'operazione.

```
public static String[][] CardAI { get; set; }
```

Questo attributo contiene tutte le carte che possiede l'intelligenza artificiale, viene aggiornato ogni volta che la IA fa una mossa e si devono togliere oppure aggiungere carte all'array multidimensionale.

```
public static String[][] CardAICopy { get; set; }
```

Invece come gli attributi spiegati precedentemente contiene una copia delle carte che possiede l'intelligenza artificiale, nel caso in cui si voglia rifare la mossa perché la telecamera non ha riconosciuto bene la carta si devono riprendere i dati precedentemente utilizzati.

```
public static int NumCard { get; set; }
```

Contiene il numero di carte che possiede la IA, solitamente all'inizio possiede 7 carte e durante il gioco verranno aggiunte oppure tolte, questo attributo serve a tenere traccia di quante carte rimangono, nel caso in cui rimangono 0 carte oppure con ancora 1 carta, l'utente deve essere avvisato di tale sconfitta oppure del avvicinamento alla sconfitta.

4.3.2 Controlli per cambiare pagina

Per cambiare pagina, non ci sono controlli, l'unico controllo che si fa, ed è quello più importante, il controllo se la telecamera è stata scelta per la partita, e il controllo è questo:

```
if (Camera.CameraId == null)
{
 MessageBox.Show("Insert a camera for continue the game.");
 Carta.ChooseView = -1;
}
```

Questo controllo viene fatto, quando si vuole passare alla prossima pagina dopo la scelta della telecamera, viene cliccato il bottone OK. Nel caso in cui la variabile che contiene la telecamera scelta è vuota, nel caso in cui la telecamera è vuota viene fatto apparire un messaggio che contiene l'informazione del fatto che l'utente deve selezionare una telecamera per continuare con il programma, verrà reimpostata la pagina della telecamera fino a quando non si scegliere quale sia da utilizzare.

5 Test

5.1 Protocollo di test

Test Case:	TC-01	**Nome:**	Riconoscimento telecamera per carta in gioco
Riferimento:	REQ-01		
Descrizione:	Questo test serve per controllare che la telecamera riesce a riconoscere la carta che si trova in gioco in maniera corretta.		
Procedura:	1. Avviare il programma 2. Scegliere la telecamera 3. Andare in avanti nel programma per fare il riconoscimento 4. Guardare che la carta sia stata riconosciuta		
Risultati attesi:	Quando si riconosce la carta, deve venire scritto nel programma che il numero e il colore della carta siano corretti.		

Test Case:	TC-02	**Nome:**	Riconoscimento telecamera per carta in possesso
Riferimento:	REQ-02		
Descrizione:	Questo test serve per controllare che la telecamera riesce a riconoscere la carta che si trova in mano all'IA in maniera corretta.		
Prerequisiti:	REQ-01		
Procedura:	1. Avviare il programma 2. Scegliere la telecamera 3. Andare in avanti nel programma per fare il riconoscimento 4. Guardare che la carta sia stata riconosciuta 5. Rifare il processo per le sette carte		
Risultati attesi:	Quando si riconosce la carta, deve venire scritto nel programma che il numero e il colore della carta siano corretti.		

Test Case:	TC-03	**Nome:**	Utilizzo IA
Riferimento:	REQ-03		
Descrizione:	Questo test serve per vedere se la IA riesce a fare la mossa corretta, scegliendo tra le carte che ha a disposizione.		
Prerequisiti:	REQ-01, REQ-02		
Procedura:	1. Avviare il programma 2. Scegliere la telecamera 3. Andare in avanti nel programma per fare la mossa alla IA 4. Controllare sequenza di mosse corretta 5. Ripetere fino a vittoria oppure sconfitta		
Risultati attesi:	Vedere la sequenza delle mosse che l'intelligenza artificiale a scelto, e vedere che le carte mostrate siano le stesse che ci sono in gioco.		

5.2 Risultati test

Test	Risultati
Riconoscimento telecamera per carta in gioco	Il risultato è positivo, il riconoscimento avviene, ma non è preciso, riconsce bene il colore, ma non riconosce bene il numero.
Riconoscimento telecamera per carta in possesso	Il risultato è positivo, il riconoscimento avviene, ma non è preciso, riconsce bene il colore, ma non riconosce bene i numeri.
Utilizzo IA	L'intelligenza artificiale funziona bene, riesce a scegliere le mosse possibili e selezionare la mossa che vuole utilizzare.

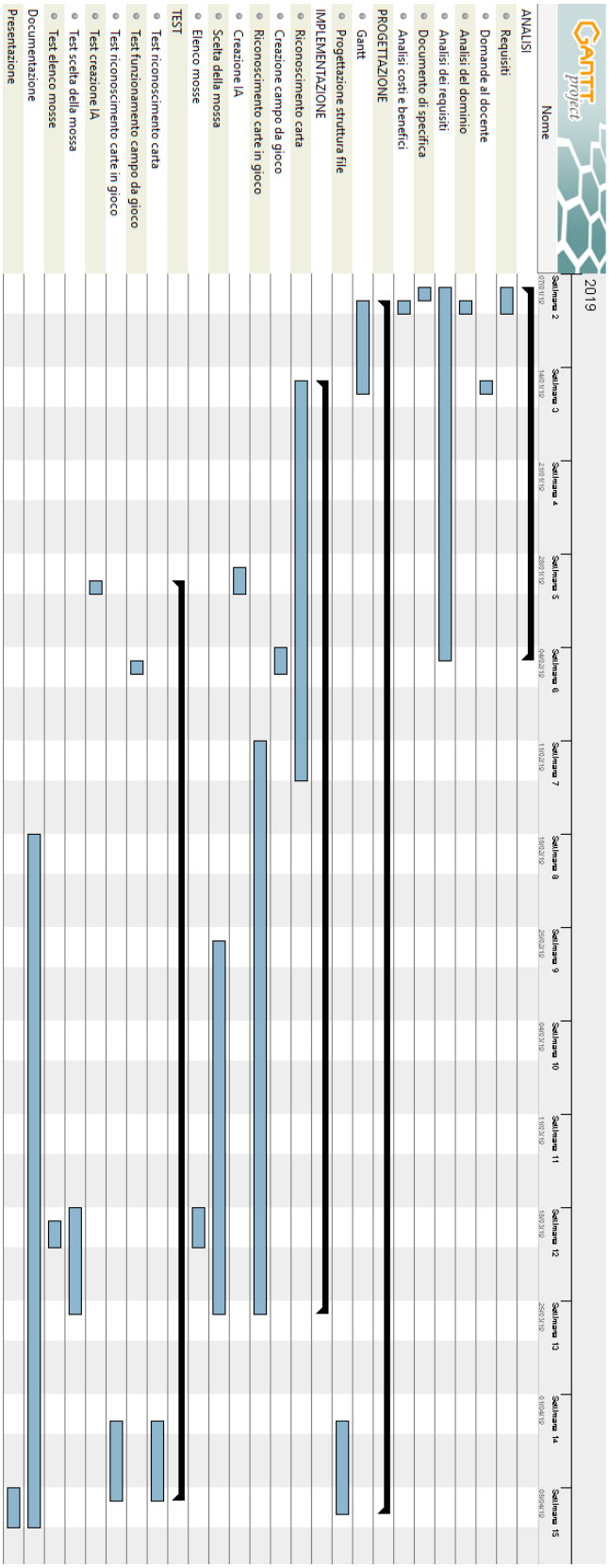
5.3 Mancanze

Nel progetto non state implementate, due requisiti mostrati in cima alla documentazione, non sono stati eseguiti per motivi di tempo, perché la telecamera e il suo riconoscimento hanno messo troppo tempo nell'implementazione e quindi non sono riuscito a fare l'implementazione del braccio meccanico. Quindi non è stato scritto il loro test perché non esistono.

6 Glossario

Parola	Significato
AI	Artificial intelligence
IA	Intelligenza artificiale
UNO	Gioco di carte che funziona con numeri e colori

7 Consuntivo



Sono state cambiate un bel po' di cose durante lo svolgimento del progetto, come ad esempio il riconoscimento della telecamera e durato molto più tempo di quanto avevo previsto, invece il braccio meccanico non è stato implementato perché non avevo abbastanza tempo per implementarlo.

8 Conclusioni

Questo progetto può essere utilizzato per fare una partita a UNO, per quando non si ha nessuno con qui giocare, oppure giocare in un gruppo è mettere all'interno una IA che permetta di giocare solamente con le carte basilari, può essere migliorata, ma in questo stato può fare una partita ma non si deve essere pignoli per quanto riguarda la qualità, perché durante l'implementazione del progetto ci sono stati problemi per quanto riguarda il riconoscimento delle carte e quanto riguarda MVVM.

8.1 Sviluppi futuri

In futuro si possono mettere molte migliorie come per esempio un riconoscimento delle carte che funzioni correttamente con una libreria oppure un software che permetta un riconoscimento funzionante al massimo delle prestazioni, mettere il programma in una struttura MVVM cercando di risolvere il problema del trasferimento dei dati, come altra aggiunta e quella di mettere un braccio meccanico che permette di fare la mossa da solo senza che l'utente gestisca le carte della intelligenza artificiale, così facendo le partite diventano ancora più interessanti perché l'utente non sa quale carte abbia la IA.

8.2 Considerazioni personali

Questo progetto mi è piaciuto per quando riguarda la programmazione della IA, perché la mia idea iniziale era quella di creare una IA che pensasse da sola le mosse che si vogliono utilizzare, ma poi e stata aggiunta la telecamera e il riconoscimento delle carte che continuavano a darmi dei problemi e alla fine mi ha preso troppo tempo e non sono riuscito a fare tutte le cose che volevo fare, come per esempio una struttura grafica migliore, la telecamera mi ha preso tutto il tempo perché non riuscivo a trovare niente su internet che mi permettesse di riconoscere le carte in gioco, ho pensato di creare un algoritmo tutto mio per il riconoscimento ma avrebbe preso ancora più tempo e non avrei avuto le conoscenze necessarie per creare un programma che riconosce le carte.

9 Bibliografia

9.1 Sitografia

- https://www.codeproject.com/Articles/125478/Versatile-WebCam-C-library Utilizzo della telecamera e prendere un'immagine 16.01.2019
- http://hemant-srivastava.blogspot.com/2012/11/image-color-detector-in-c.html Riconoscimento del colore 16.01.2019
- http://www.pixel-technology.com/freeware/tessnet2/ Codice Tesseract 22.01.2019
- https://coredump.one/questions/18418150/badimageformatexception-in-c Soluzione non funzionante sul mio computer 22.01.2019
- https://stackoverflow.com/questions/11944778/tessnet2-error-in-c-sharp Soluzione del codice di Tesseract 23.01.2019
- https://tesseract.patagames.com/help/html/baa0aa10-7805-4ae6-b6e9-9df777c4678c.htm Possibile codice funzionante 23.01.2019
- https://stackoverflow.com/questions/13755007/c-sharp-find-highest-array-value-and-index Per controllare il valore più alto 28.01.2019
- https://stackoverflow.com/questions/3975290/produce-a-random-number-in-a-range-using-c-sharp Utilizzo di valori casuali per le carte in gioco 29.01.2019
- https://stackoverflow.com/questions/12567329/multidimensional-array-vs Creazione array multidimensionale 29.01.2019
- https://stackoverflow.com/questions/599369/array-of-an-unknown-length-in-c-sharp Codice per creare un array con il numero di campi indeterminati 30.01.2019
- https://stackoverflow.com/questions/1222117/multidimensional-lists-in-c-sharp Creare un array che contiene un altro array 30.01.2019
- https://github.com/openalpr/openalpr/wiki/Integrating-OpenALPR Possibile codice per implementare la nuova libreria 05.02.2019
- https://ironsoftware.com/csharp/ocr/ Libreria OCR 05.02.2019

- https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/ Riconoscimento con Azure 05.02.2019
- https://stackoverflow.com/questions/54549225/image-recognition-in-c-sharp Domanda che ho posto su internet 06.02.2019
- https://cloud.google.com/vision/?hl=it Libreria di riconoscimento fornita da Google 06.02.2019
- http://eyesbot.com/blog/?preload=object_recognition.txt Progetto in cui si fa il riconoscimento di oggetti 11.02.2019
- http://james-ramsden.com/c-convert-image-bitmapimage/ Codice per convertire Bitmap in BItmapImage 11.02.2019

10 Allegati

- Diari di lavoro
- Abstract