

# Realizzazione di un motore scacchistico in **Visual Basic 6**

## PARTE 1

*In questo articolo illustreremo passo passo la realizzazione di un software che “gioca a scacchi”, utilizzando come ambiente di sviluppo il Visual Basic 6.*

di **Andrea Lanza** > [alanza@infomedia.it](mailto:alanza@infomedia.it)

**L**a prima tra le tante domande che si pone colui che per la prima volta si avvicina alla programmazione di un motore scacchistico è: “Quale linguaggio è migliore o necessario per realizzare un programma di questo tipo?”.

La risposta è molto semplice: “Utilizzate il linguaggio che a voi è più familiare!”.

Chi è bravo a programmare in Visual Basic potrà ottenere grosse soddisfazioni dal suo “motore” senza dover investire parecchio tempo per imparare ad utilizzare un nuovo linguaggio.

Certo, il Visual Basic è meno adatto in termini di prestazioni rispetto al Delphi o al C++, dato che un fattore fondamentale nella realizzazione di programmi di questo tipo è la velocità di elaborazione e di calcolo.

Ritengo comunque che sia importante utilizzare il linguaggio a ciascuno di noi più congeniale, per imparare e conoscere tematiche quali la rappresentazione della scacchiera, gli algoritmi di ricerca e di valutazione, come evitare le ripetizioni e lo stallo, che cos'è la quiescenza e tante altre problematiche che un programmatore di software scacchistico si troverà a dover affrontare.

Una volta realizzato il nostro motore saremo sempre in tempo a fare il porting in un altro linguaggio.

In questo speciale saranno utilizzati come esempio parti di codice di “Matilde”, il motore scacchistico da me realizzato. La versione cui si riferiscono tali esempi è la 2.5.5 e i sorgenti completi sono disponibili all'indirizzo “<http://www.cds.pc.it/Matilde/Archive.htm>”.

Ma ora basta con i preamboli ed entriamo nel vivo della nostra sfida.

## L'interfaccia Winboard

Il nostro motore altro non è che un file eseguibile!

Per renderlo utilizzabile è necessario interfacciarlo ad un software grafico (per intenderci, una scacchiera con i pezzi bianchi e neri, dove l'utente interagisce con essi tramite il mouse) che supporti il protocollo *Winboard*.

Esistono diversi tipi di protocolli di comunicazione per motori scacchistici, noi utilizzeremo lo standard *Winboard* essendo uno dei più popolari (all'indirizzo potrete scaricare la versione 4.2.7 che comprende il motore “GNU Chess 5.07”). La Figura 1 illustra come interfacciare due motori scacchistici tramite questa interfaccia:

All'avvio, l'interfaccia invia una serie di comandi di inizializzazione (che vedremo più avanti), il motore bianco dopo avere elaborato la mossa la invierà all'interfaccia *Winboard*, la quale si occuperà di rappresentarla graficamente sulla scacchiera e la invierà al motore avversario.

A questo punto il motore nero aggiornerà la sua scacchiera interna, elaborerà una contromossa, la invierà all'interfaccia e così via fino al termine della partita.

La comunicazione motore/interfaccia e viceversa avviene tramite due canali, uno di ingresso e uno di uscita che verranno inizializzati all'avvio del nostro programma:

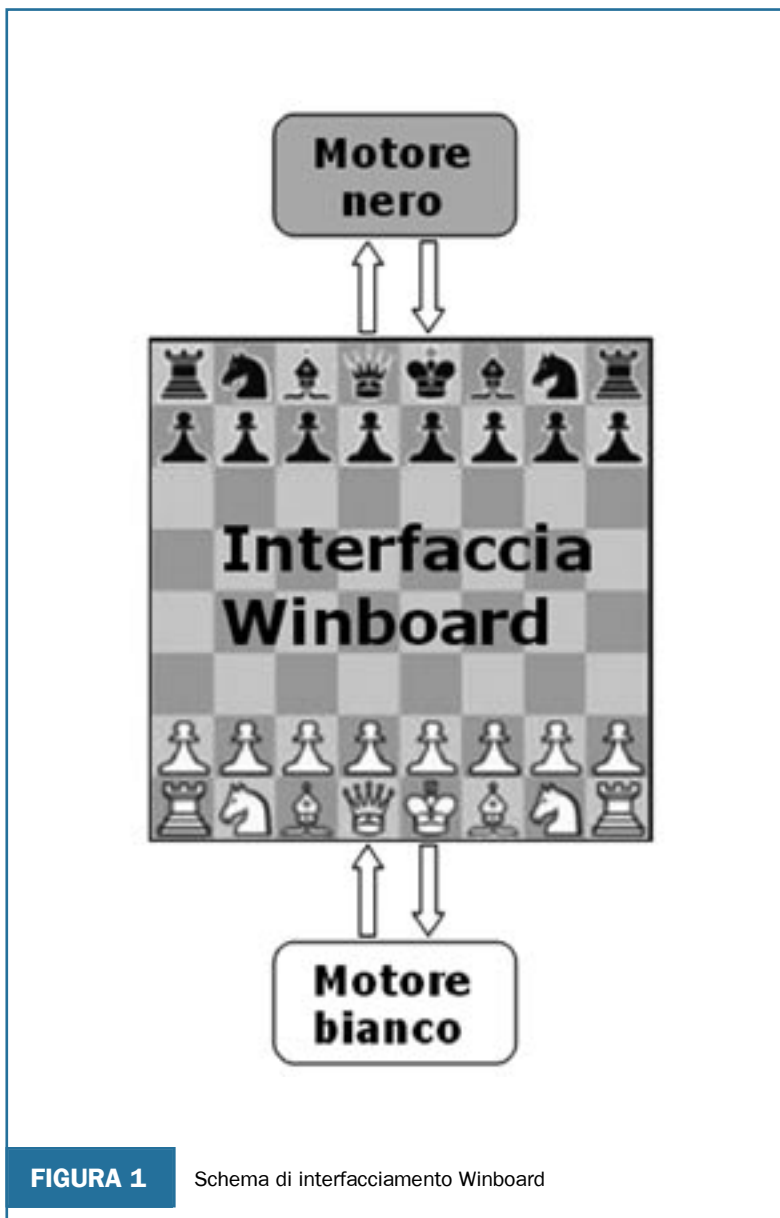
'Dichiarazione dei canali di ingresso e di uscita.

Public InputChannel As Long

Public OutputChannel As Long

'Impostare questa subroutine come oggetto di avvio per

'richiamarla all'avvio del programma.



```
Private sub Main()
    InputChannel = GetStdHandle(-10)
    OutputChannel = GetStdHandle(-11)
End sub
```

Ricordiamoci per sicurezza di chiudere i canali quando usciamo dalla nostra applicazione:

```
'=====
'Chiusura dei canali e fine dell'applicazione.
'=====
Public Sub ExitProgram()
    CloseHandle InputChannel
    CloseHandle OutputChannel
End
End Sub
```

## Letture ed interpretazione dei programmi

Per poter leggere i dati in arrivo dall'interfaccia *Winboard* è necessario realizzare un ciclo che, per tutta la durata della nostra applicazione, esegue un controllo sul canale in ingresso.

A tale scopo aggiungeremo un controllo "Timer" nell'unica form del nostro programma impostando la proprietà Interval a 100 mS e la proprietà Enabled a False.

Il timer sarà avviato durante la fase di inizializzazione del nostro motore:

```
'=====
'Impostare questa subroutine come oggetto di avvio per
'richiamarla all'avvio del programma.
'=====
Private sub Main()
    InputChannel = GetStdHandle(-10)
    OutputChannel = GetStdHandle(-11)
    MainForm.MainLoop.Enabled = True
End sub
```

Quindi ogni decimo di secondo si attiverà l'evento Timer:

```
'=====
'Ciclo principale.
'Legge continuamente i comandi in arrivo dalla Winboard.
'=====
Private Sub MainLoop_Timer()
    Dim S As String
    If PoolCommand = True Then
        S = ReadCommand
        ParseCommand S
    End If
End Sub
```

Se sono presenti dei dati in arrivo dalla *Winboard* (PoolCommand) sono letti (ReadCommand) ed interpretati (ParseCommand).

La lista seguente illustra solamente i comandi principali trasmessi dall'interfaccia *Winboard* al motore; un elenco completo lo potrete trovare all'indirizzo[3]:

- "e2e4" Mossa in arrivo dalla *Winboard* nel formato Algebraic Notation (mossa di esempio). La mossa sarà convertita nel formato interno del nostro motore. Aggiungeremo la nostra scacchiera interna. Avvieremo la funzione di ricerca e di valutazione.
- "new" Comando inviato ad ogni nuova partita. Resettiamo la nostra scacchiera. Ripristiniamo tutte le nostre variabili allo stato iniziale, sta per cominciare una nuova partita.
- "go" Informa il motore che è il suo turno di fare la mossa. Avviamo la nostra funzione di ricerca. Una volta esaurito il tempo a disposizione giochiamo la mossa.

- “time” Il tempo residuo a nostra disposizione in centesimi di secondo.  
Memorizziamo il tempo a nostra disposizione.  
Lo utilizzeremo per interrompere la funzione di ricerca e giocare la mossa.
- “edit” Ci informa sullo stato della scacchiera.  
Utile ad esempio quando si carica una partita salvata in precedenza.
- “quit” Utilizziamo questo comando per uscire dall’applicazione.  
Arrivederci, grazie per la partita...

## Le operazioni di scrittura

Il motore utilizzerà il canale di uscita per inviare i comandi alla *Winboard*:

```

'=====
Invia il comando alla Winboard.
'=====
Public Function SendCommand(ByVal sCommand As String) _
    As String
    Dim lBytesWritten As Long
    Dim lBytes As Long
    Dim RC As Long
    sCommand = vbCrLf & sCommand & vbCrLf
    lBytes = Len(sCommand)
    RC = WriteFile(OutputChannel, ByVal sCommand, lBytes, _
        lBytesWritten, ByVal 0&)
    SendCommand = sCommand
End Function
    
```

La funzione `SendCommand` può ad esempio essere richiamata nel seguente modo:

```
SendCommand "move " & WinboardMove
```

Il comando `move` informa la *Winboard* che è in arrivo una mossa dal motore e la variabile *WinboardMove* contiene la mossa in formato algebraic notation (ad esempio: e2e4). La lista seguente illustra una parte dei comandi che è possibile inviare alla *Winboard*.

- `move WinboardMove`  
L’interfaccia *Winboard* eseguirà la mossa contenuta nella variabile *WinboardMove*.
- `result {comment}`  
Quando la partita termina inviamo un commento alla *Winboard* in formato PGN (1-0, 0-1 o 1/2-1/2) seguita da un commento:  
0-1 {Black mates}  
1-0 {White mates}  
1/2-1/2 {Draw by repetition}  
1/2-1/2 {Stalemate}
- `tellics Message`  
Questo comando è utilizzato nelle partite sui server Internet per comunicare con il giocatore avversario.

## Conclusioni

In questa prima parte dello speciale abbiamo trattato l’argomento *Winboard*, un protocollo di comunicazione molto diffuso per interfacciare due motori scacchistici.

Abbiamo visto come leggere ed interpretare i comandi in arrivo e come inviarli in maniera corretta all’interfaccia *Winboard*.

Le parti di codice di “Matilde” inseriti in questo articolo sono già sufficienti per realizzare le fondamenta per l’interfacciamento alla nostra *Winboard*.

Supponiamo di avere installato il programma *Winboard* nel seguente path “c:\programmi” e di utilizzare come motore avversario “GNU Chess 5.07”; se vogliamo che il nostro motore sia il giocatore bianco eseguiremo questo file batch:

```
c:\programmi\winboard\WinBoard /cp /
fcp="MioMotore.exe" /fd=" c:\PercorsoMioMotore"
/scp="GNUChes5.exe" /sd="c:\programmi\Winboard".
```

Se al contrario, il nostro motore sarà il giocatore nero, eseguiremo quest’altro file batch:

```
c:\programmi\winboard\WinBoard /cp /
fcp="GNUChes5.exe" /fd=" c:\programmi\Winboard "
/scp="MioMotore.exe" /sd=" c:\PercorsoMioMotore".
```

## Bibliografia

- [1] Tim Mann, “XBoard and Winboard”, <http://www.tim-mann.org/xboard.html>
- [2] <http://www.tim-mann.org/xboard.html>
- [3] “<http://www.tim-mann.org/xboard/engine-intf.html>”

### Download del progetto

Matilde versione 2.5.5  
Motore scacchistico realizzato in Visual Basic 6

### Andrea Lanza

Lavora da 18 anni come programmatore in un’azienda che si occupa di analisi ambientali. Utilizza il Visual Basic 6, il Delphi e il Visual C++ per realizzare programmi per la gestione di sistemi di rilevamento ambientali e Gas Serra in ambienti Windows e Linux. È membro del GSei (Gruppo Scacchi e Informatica), dedicato allo studio e sviluppo di software scacchistico.