Dopo tutto questo tempo di silenzio (chiedo perdono!) incominciamo a considerare alcune funzionalità che riguardano più da vicino la programmazione di avventure testuali: cominciamo finalmente a vedere come si utilizzano i verbi e le variabili.

[Nota: dal momento che i verbi preimpostati in Inform sono diversi, ne verrà caricata una lista, tratta dal manuale di Scarpa, sul mio sito personale: pian piano la esamineremo tutta quanta]

Vediamo il codice sorgente:

```
Constant Story "Esempio utilizzo verbi/variabili^";
Constant Headline "Parte 1^";

Include "Parser";
Include "Verblib";
Include "Replace";

Global punti_vita = 10;

Object stanza "Una stanza"
with initial [; style bold; print "^Una stanza^^"; rtrue; ],
before [;
Look:
   style bold; print "^Una stanza^^";
   style roman; print "Ti trovi in una stanza";
   if (children(stanza) == 1) {
      print " spoglia.^"; rtrue; }
   else {
      print " in cui puoi vedere:^";
      if (fungo in self) print "- un fungo^";
      if (mela in self) print "- una mela^";
      if (torta in self) print "- una torta^";
      print "e nient'altro.^";
      rtrue;
      };
],
has light;

Object -> fungo "fungo"
with name 'fungo',
description "Un fungo dall'aspetto decisamente rivoltevole.",
before [;
Eat:
   print "Il suo aspetto avrebbe dovuto farti presagire qualcosa di brutto.^";
   style underline; print "Perdi due punti vita.^"; style roman;
   punti_vita = punti_vita-2;
   rtrue;
],
has edible;

Object -> mela "mela"
with name 'mela',
```

```
description "Ricorda vagamente quella di biancaneve...",
before [;
Eat:
   print "Non avresti proprio dovuto mangiarla.^";
   style underline; print "Perdi tre punti vita.^"; style roman;
   punti_vita = punti_vita-3;
   rtrue;
],
has edible female;

Object -> torta "torta"
with name 'torta',
description "Una torta dall'aspetto dolcissimo!",
before [;
Eat:
   print "Decisamente gustosa!^";
   style underline; print "Guadagni cinque punti vita.^"; style roman;
   punti_vita = punti_vita+5;
   rtrue;
],
has edible female;

[ Initialise;
location = stanza;
];

[ DiagnosiSub;
print "Hai ", punti_vita, " punti vita.^";
];

Verb 'diagnosi' * -> Diagnosi;

Include "ItalianG";
```

Ho proposto un sorgente con alcune cose che dovreste ricordare per rispolverare quello che è stato detto tanto tempo fa e ricominciare gradualmente gli articoli su I6.

La prima cosa che si vede dopo i tre *Include* è la **variabile*** *punti_vita*, preceduta nella sua dichiarazione dalla parola **Global**; vediamo anche che le abbiamo assegnato il valore di partenza *10*: questo d'ora in poi sarà un valore che potremo andare a modificare tramite opportune istruzioni che possono essere compiute dal giocatore.

Vediamo l'oggetto *Stanza*: ne ho modificato leggermente la struttura per mettere in evidenza l'istruzione **children()**. Normalmente tutto quello che è scritto nel corpo dell'oggetto *Stanza* viene eseguito automaticamente dal programma, così possiamo vedere praticamente cosa viene fatto di solito. Appena si entra in una locazione, il programma esegue in automatico l'istruzione **look** (guarda) riferita proprio alla locazione in cui è entrato il giocatore, visualizzando il nome della stanza (che, nell'esempio, è quello scritto nell'istruzione *Initialise*) e la sua descrizione. Attenzione: per le locazioni **viene fatta una distinzione tra look ed examine**, nonostante concettualmente siano praticamente lo stesso verbo: in questo esempio l'aver scritto *examine* al posto di *look* avrebbe portato ad un errore.

Come già sappiamo, la modifica del verbo *look* è inserita nell'istruzione *before*, nel senso che dopo aver ricevuto l'input (ossia dopo che il giocatore ha digitato *> guarda*, o, in questo caso, dopo che il programma ha capito che bisogna "guardare la stanza"), ma *prima* di accedere al messaggio di default corrispondente al verbo *look (*che *non* è quello da noi specificato, ma uno già preimpostato nelle librerie), il programma viene deviato a considerare la risposta allo stesso verbo nella istruzione *before* (dell'oggetto su cui si agisce). Se questa descrizione è oscura, accenniamo all'istruzione *after*, che in un certo senso è complementare: se i verbi da modificare vengono inseriti in quest'ultima istruzione, allora viene svolto ogni comando come di default (fornendo anche le risposte di default) e *dopo* vengono prodotti gli effetti del verbo modificato.
Normalmente, tranne per i casi in cui è specifica, si utilizza l'istruzione *after* solo per modificare le conseguenze di un verbo, tenendo come buone le risposte di default senza doverne scrivere ogni volta una apposta.
Nel caso fosse ancora oscuro il procedimento (probabilmente lo è), aspettate di leggere l'articolo in cui si parlerà dell'istruzione *after*, e per il momento andate di before.

Sull'oggetto *Stanza* nient'altro da dire (l'unica particolarità è l' *if else* che abbiamo già spiegato), a parte l'istruzione **children()**. Ricordate l'albero degli oggetti? Se l'oggetto A contiene l'oggetto B, con Inform ci si può riferire ad A come **parent(B)**, e a B come **child(A)**, dove *parent* e *child* possono essere interpretati genealogicamente come "colui che 'sta (immediatamente) su'" e "colui che 'sta (immediatamente)giù'" rispettivamente. Questo è utile quando un oggetto produce effetti diversi a seconda della locazione in cui si trova: è evidente che se le locazioni cambiano l'unico punto fisso resta l'oggetto, e quindi le locazioni vengono identificate tramite l'istruzione *parent()*. Analogamente, un oggetto può produrre effetti diversi a seconda degli oggetti che contiene: è evidente che gli oggetti contenuti variano, mente non cambia il contenitore, e quindi è utile riferirsi al generico oggetto contenuto come *child()*. [Notare che se C è contenuto in B, l'unico "parent" di C è B e l'unico "child di A rimane B!]
Tutto bello, ma l'istruzione non è *child()*, bensì *children()*; la differenza sta nel fatto che quest'ultima "conta" gli oggetti che stanno nell'oggetto indicato tra le parentesi, quindi a quest'ultima corrispondono valori numerici (da 0 a n), mentre alle prime due corrispondono sempre nomi di oggetti (o, nel caso si voglia indicare il niente, l' "oggetto" *nothing"*).
[Domandone: l'istruzione allora significa: "Se nella stanza c'è uno ed un solo oggetto allora..."; perché ho indicato il valore 1 e non 0 come ci si dovrebbe aspettare, ovvero nel caso in cui il giocatore abbia raccolto tutti e tre gli oggetti?]
Al termine delle due condizioni dell'*if else* come usuale un *rtrue* per evitare che il programma continui ad eseguire le azioni di default, ovvero valutare se ci sono oggetti nella stanza e comunicarcelo (cosa da evitare, dal momento che l'abbiamo già inclusa noi nella descrizione della stanza!)

Veniamo agli altri tre oggetti, e prendiamo in considerazione, ad esempio, il fungo. Vediamo che anche in questo caso abbiamo deciso di modificare il verbo "mangia" e, essendo quest'ultimo preimpostato nelle librerie, ci riferiamo ad esso col corrispettivo inglese, come tutti quelli preimpostati.
L'unica cosa degna di nota, a parte che fornisce una risposta specificata da noi diversa da quella di default, è vedere che questo comando modifica il valore della variabile *punti_vita*, nel semplice modo che viene indicato.
Vediamo anche che è presente un nuovo attributo: **edible**, che vuol proprio dire "edibile": questo attributo si attacca agli oggetti che possono essere mangiati per farli riconoscere come tali; in particolare la sua utilità sta nel fatto che ci evita di dover aggiungere l'istruzione *remove OGGETTO* ad un'eventuale modifica del verbo *Eat*, altrimenti potremmo incorrere nell'eventualità che il giocatore mangi qualcosa e continui poi a ritrovarselo nell'inventario!

D'accordo, ma come fa un giocatore a sapere quanti punti vita gli rimangono?

Per questo c'è bisogno di un verbo, nella fattispecie un qualcosa che non è verbo ma che viene interpretato come tale, ovvero la parola "diagnosi".

Ogni volta che si crea un verbo bisogna innanzitutto creare una **routine** in cui specificare in che modo il verbo corrispondente agisce, e una **dichiarazione** in cui indicare

- con quale/i parola/e il giocatore può riferirsi al verbo
- la routine corrispondente

(più altre opzioni accessorie, come per i verbi più complicati, che vedremo in seguito)

La routine è del tutto simile a quella che già conosciamo: l'I*nitialise*:

- si apre una quadra
- si scrive il nome della routine **a cui va aggiunto il suffisso -Sub**
- si aggiunge un punto e virgola per terminare la dichiarazione del nome della routine
- si scrivono le istruzioni di quello che ci si aspetta che faccia la routine (nella fattispecie dirci quanti punti vita ci rimangono*)
- si chiude la quadra
- ed infine di nuovo punto e virgola.

Per quanto riguarda la dichiarazione di un verbo, come per gli oggetti era necessaria la parola *Object*, qui si utilizza la parola **Verb**, seguita dal nome che il giocatore deve digitare inserito tra singoli apici. Completano questa dichiarazione, in questo preciso ordine, un asterisco ed una freccia (capiremo a cosa servono più avanti), il nome della routine a cui il verbo si riferisce (senza il suffisso -Sub!) ed il classico punto e virgola.

[Nota di game design: chiaramente, per tutti i verbi non preimpostati, e che quindi non possono essere conosciuti dal giocatore, va indicato più o meno chiaramente come possono essere scoperti.
L'esempio della diagnosi mi è venuto da Zork: esso veniva indicato addirittura nel depliant delle istruzioni, data la sua importanza.
Ma come abbiamo visto, con questo procedimento, non si dichiarano solamente verbi veri e propri: così facendo si possono dichiarare anche espressioni, parole... Nella mia ultima at, ad esempio, mi sono rifatto all'episodio tolkeniano in cui Gandalf deve aprire le porte di Moria, e ho dichiarato come "verbo" l'esclamazione "amico" che, digitata nella locazione giusta, permetteva l'accesso ad un luogo.
La ricerca di un verbo o di una parola (potrebbe ad esempio essere un verbo, ma in una lingua aliena!) può costituire un valido enigma (attenzione: sto parlando di "ricerca del verbo" e non di "caccia al verbo": una di quelle noiosissime situazioni in cui il programmatore ha concesso pochi sinonimi per una parola che i giocatori hanno dovuto buttare tempo per trovare quelli giusti!), quindi possono anche essere date scarse indicazioni su come trovare quel preciso verbo o quella parola!
Oppure può non essere dichiarato affatto, nel caso si tratti di un easter egg (vedi numerose avventure della Infocom)]

Attenzione: TUTTI i verbi devono essere dichiarati DOPO le routines corrispondenti, e generalmente verbi e routines sono posti DOPO la *Initialise*, pena possibili malfunzionamenti (se non addirittura mancate compilazioni) del programma.

*Vediamo ora di spiegare bene come si dice al programma di scrivere il valore di una variabile nel bel mezzo della stampa di un testo, evento che era già occorso in passato e sul quale ho sorvolato.
Siccome si deve stampare del testo si procede nel solito modo: *print "TESTO...* , nel momento in cui bisogna che venga stampato *il valore* di una variabile si chiudono le virgolette, si aggiunge una virgola,

si scrive il nome della variabile di cui vogliamo venga scritto il valore, si mette un'altra virgola, si riaprono le virgolette e si continua col testo normale che si stava scrivendo.