Come scrivere (e giocare) delle avventure testuali in Inform e Glulx

Vincenzo Scarpa

Seconda edizione: gennaio 2010

Come scrivere (e giocare) delle avventure testuali in Inform e Glulx

Autore: Vincenzo Scarpa (scarvin@libero.it)

Copertina: Little Falls (sfondo) di Alessandro Schillaci e Roberto Grassi, 2005
Enigma (in alto a destra) di Marco Vallarino, 2001-2003
Adventure (in basso a destra) di Will Crowter e Don Woods – traduzione e adattamento di Giovanni Riccardi, 1997-2004

Prima edizione: gennaio 2006
Seconda edizione: gennaio 2010

INDICE

APPENDICI

INTRODUZIONE

Se state cercando un libro illustrato di Alessia Marcuzzi o di Sabrina Ferilli, avete proprio sbagliato strada; tutto quello che qui potete trovare è un manuale su Inform[1], il linguaggio di programmazione più usato per scrivere le avventure testuali.

Un'avventura testuale può essere paragonata a una sorta di "storia interattiva" dove il lettore (o meglio, il giocatore) è chiamato in causa nello svolgere le opportune azioni (digitate attraverso la tastiera di un computer e seguite dalla pressione del tasto di Invio) che gli consentano di proseguire nella storia stessa, come ad esempio spostare determinati oggetti, parlare con delle persone, o magari fare un po' più di attenzione ai piccoli dettagli che a prima vista sfuggono.
Inform, dicevo, è attualmente lo strumento più potente ed efficace per poter realizzare un'avventura testuale, ma anche uno dei più difficili da usare; infatti, a meno che non si abbia una precedente esperienza nell'ambito della programmazione (in particolare sul linguaggio C), alcuni concetti tipicamente "informiani" e non come ad esempio le funzioni, le classi e gli oggetti, risultano essere veramente ostici da apprendere per chi è alle prime armi. Questo però non vuol dire che sia impossibile capirli; un po' di pazienza, tenacia e un buon manuale sono gli ingredienti magici per entrare in questo fantastico mondo, popolato da tutte le creature che volete: elfi, maghi e orchetti, nel caso di un'avventura fantasy, oppure demoni, zombie e vampiri in una bella storia horror.

Il manuale da me scritto è strutturato in diversi capitoli e appendici, e affronta una serie di argomenti molto importanti ma spesso poco documentati. Leggendolo, non diventerete di certo dei geni della programmazione, ma riuscirete a capirne abbastanza da poter poi essere voi in grado di scrivere una vostra avventura (almeno si spera). A chi conosce la lingua inglese e ha intenzione di approfondire questo straordinario linguaggio di programmazione, mi permetto di consigliare l'INFORM BEGINNER GUIDE di Roger Firth (tradotto in italiano all'indirizzo http://www.inform-italia.org/docs/gip e scaricabile anche qui) e la quarta edizione dell'INFORM DESIGNER'S MANUAL (il manuale ufficiale di Inform scritto da Graham Nelson, scaricabile anche qui).

Desidero inoltre ringraziare tutti coloro che mi hanno aiutato, in particolare Tommaso Caldarola, Francesco Cordella, Giancarlo Niccolai, Alessandro Schillaci, Tommaso Percivale, Luca Bertaiola, e Paolo Lucchesi per i loro preziosi consigli; Raffaello Valesio per la traduzione di Ruins; Giovanni Riccardi per aver scritto INFIT; Graham Nelson per avermi autorizzato a tradurre in italiano Ruins e per aver creato Inform.

Per consigli, insulti, opinioni, maledizioni e altro ancora la mia e-mail è scarvin@libero.it, ma potete anche contattarmi sul newsgroup it.comp.giochi.avventure.testuali. Buon divertimento…

Vincenzo Scarpa
Settimo Torinese (TO), 10 gennaio 2010

[1] Il manuale qui presente fa riferimento alla versione 6 di questo linguaggio. Esiste anche la versione 7 di Inform che permette di creare un'avventura testuale in una modalità molto più vicina al modo di pensare "umano" (con un linguaggio, cioè, naturale). Altre informazioni potete trovarle su http://inform7.com/, http://milleuna.sourceforge.net/ e http://inform7.com/extensions/translations/#Italian.

CAPITOLO 1
COSA OCCORRE PER INIZIARE

§1.1 Compilatore, interprete e altro ancora...

Ora che avete letto (almeno mi auguro) l'introduzione, all'interno della quale è stata spiegata cos'è a tutti gli effetti un'avventura testuale, possiamo allora passare al lato più "pratico", ovvero a cosa occorre per crearla. Se in qualità di utente dovessi scaricare i soli file di Inform, mi ritroverei con le librerie di sistema in inglese e un compilatore (ovviamente anch'esso in inglese); poco, davvero troppo poco per poter cominciare a lavorare con un linguaggio di programmazione. Qualcuno più esperto potrebbe iniziare ad aprire, per esempio, il blocco note di Windows per scrivere il sorgente e magari la finestra del prompt di MS-DOS per usare il compilatore. Già, ma non è scomodo lavorare su delle finestre che devo tutte le volte attivare/disattivare per poter effettuare le varie operazioni? E poi… come faccio a scrivere un'avventura testuale in lingua italiana se le librerie di sistema sono in inglese? E come faccio a giocarla, dal momento che il file compilato non è un eseguibile (.exe) ma nel formato Z-code (.Z5 o .Z8)?
Allora… procediamo con ordine, facendo una breve rassegna su tutto quello che ci serve[1]:

1. LE LIBRERIE DI SISTEMA (versione 6.11, sulla quale è basato il manuale, o successiva), composte di nove file che includono la grammatica e il parser;
2. LE LIBRERIE IN ITALIANO (versione 2.5, sulla quale è basato il manuale, o successiva), composte di tre file e scritte da Giovanni Riccardi;
3. IL COMPILATORE (versione 6.30, sulla quale è basato il manuale, o successiva), disponibile per diversi sistemi operativi tra cui ovviamente Windows;
4. L'INTERPRETE (WINDOWS FROTZ v1.08 o successivo), che esegue i file .Z5 e .Z8 per poter giocare alle avventure testuali create con Inform;
5. UN EDITOR DI TESTO, ovvero il bellissimo WIDE di Alessandro Schillaci e Paolo Lucchesi.

§1.2 L'installazione

La prima cosa da fare è la più banale di tutte: creare sotto Windows una directory (o cartella) denominata inform sulla partizione C: (o su un'altra lettera nel caso che il vostro disco fisso sia suddiviso in più partizioni). Create poi al suo interno le seguenti sottodirectory: libraries (per le librerie di sistema e Infit), interpreter (per l'interprete Inform, ovvero Windows Frotz 2002) e infine wide. Decomprimete ora con il programma Winzip i seguenti file:

- inform_library611.zip e infit25.zip nella directory libraries;
- inform_compiler630.zip nella directory inform;
- WindowsFrotz2002.zip nella directory interpreter;
- wide099beta.zip nella directory wide.

L'ultima cosa che rimane da fare è rinominare le librerie di sistema che ora si trovano in "C:\inform\libraries" (nell'ordine English, Grammar, infix, linklpa, linklv, Parser, parserm, VerbLib, verblibm) con l'estensione .h (English.h, Grammar.h e così via).
Fine dell'installazione. Come vedete non è poi così difficile...[2]

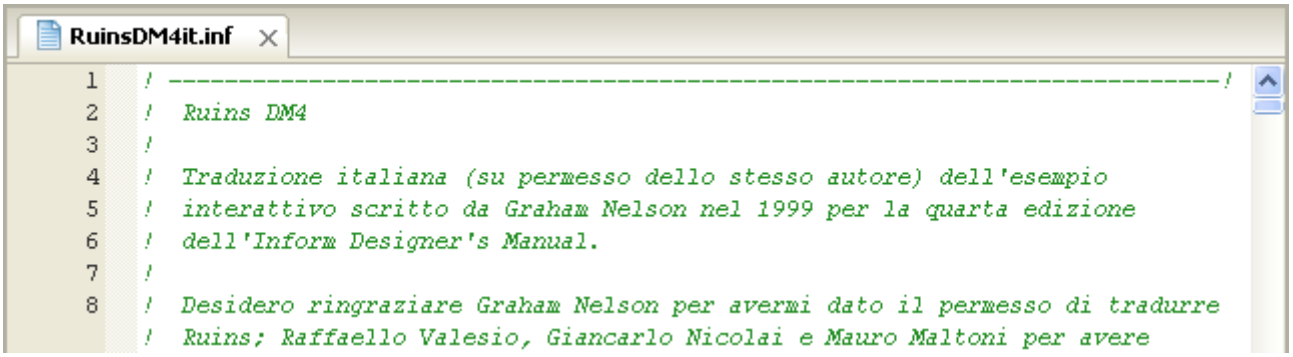
[1] Per scaricare tutto l'occorrente cliccate, con il tasto sinistro del mouse, sulle parole sottolineate tra parentesi. In alternativa, potete sempre scaricare l'Inform Pack (http://slade.altervista.org/downloads.html), un pacchetto che contiene tutto (o quasi) l'occorrente per iniziare a programmare con Inform, organizzato in directory secondo una comoda struttura.
[2] Se proprio non riuscite a fare quanto fin qui detto, date allora un'occhiata al contenuto della directory Capitolo1 del file esercizi_manuale.zip.

§1.3 Perché WIDE?

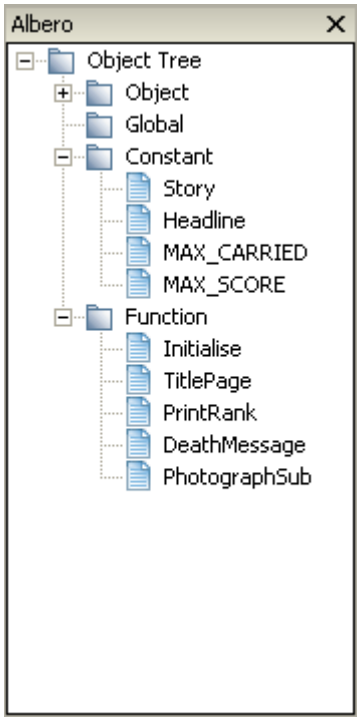
È presto detto: WIDE non funge solo da editor di testo, ma anche da compilatore e visualizzatore di oggetti, funzioni, costanti e molto altro ancora. È, insomma, un vero e proprio ambiente integrato di programmazione, che ci permette di svolgere da una sola finestra tutte le principali azioni richieste da Inform per la creazione di un'avventura testuale. Esse sono nell'ordine:

1. LA CREAZIONE DEL CODICE SORGENTE;
2. LA COMPILAZIONE DEL CODICE SORGENTE NEL FORMATO Z-MACHINE (.Z5 O .Z8);
3. L'ESECUZIONE DEL FILE COMPILATO MEDIANTE L'APPOSITO INTERPRETE.

Apriamo ora il programma. La finestra principale di WIDE è suddivisa in tre ulteriori finestre (chiamate anche aree di lavoro). Quella in alto a destra è di fatto l'editor di testo, all'interno della quale potete creare e salvare il codice sorgente (contraddistinto dall'estensione .inf):

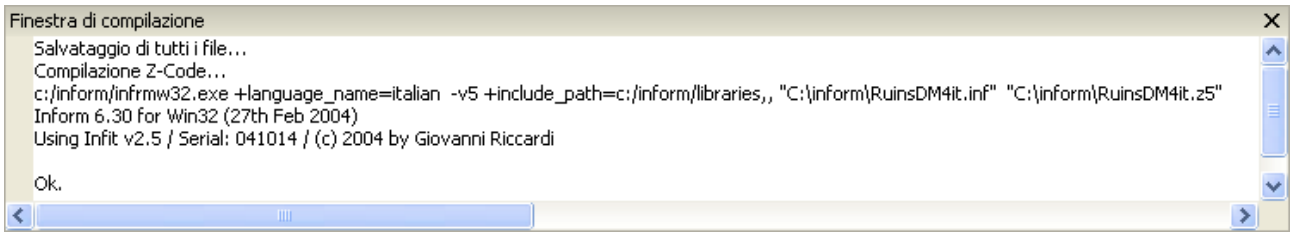


La finestra in alto a sinistra visualizza (se presenti) tutti gli oggetti, le costanti e le funzioni relative ad ogni listato aperto:



ma è comunque possibile, dal menu Albero, scegliere di visualizzare (sempre se presenti) anche le classi, gli include e i verbi.

La finestra in basso è indispensabile durante la fase di compilazione, perché segnala la presenza o meno di errori all'interno del codice sorgente che la maggior parte delle volte non permettono la creazione del file eseguibile:



Resta infine da vedere come CREARE, APRIRE, SALVARE, COMPILARE ed ESEGUIRE il codice sorgente. Basta premere i seguenti pulsanti sulla barra degli strumenti, nell'ordine:

 per creare un nuovo codice sorgente;

 per aprire un codice sorgente precedentemente salvato;

 per salvare il codice sorgente;

 per compilare il codice sorgente;

 per eseguire il file compilato con l'apposito interprete (nel nostro caso Windows Frotz 2002).

Il file compilato (l'avventura testuale vera e propria) ha di default l'estensione .Z5. A volte però, può capitare che l'avventura sia più lunga della norma, e a quel punto occorre compilarla con l'estensione .Z8. WIDE gestisce anche quest'ulteriore opzione dal menu Zcode e i due formati sono pienamente supportati da Windows Frotz 2002.

§1.4 Per chi non vuole (o non può) usare WIDE e Windows Frotz 2002

Il mondo è bello perché è vario. È per questo motivo che Tommaso Caldarola e Francesco Cordella mi hanno segnalato l'esistenza per Windows di TEXTPAD, un magnifico editor di testo completo ed efficiente prodotto dalla Helios Software e scaricabile dal sito http://www.textpad.com/. Un articolo di Roger Firth (http://www.onyxring.com/InformGuide.aspx?article=14) spiega poi come configurarlo correttamente per Inform.
Paolo Lucchesi ha invece adattato CRIMSON EDITOR (un editor di testo gratuito che non ha nulla da invidiare a Textpad) a Inform creando un apposito file di evidenziazione della sintassi come spiegato nel file crimson_inform.txt.
Impossibile poi, non citare lo stratosferico JIF (versione 2.0 - Megapack Edition, sulla quale è basato il manuale, o successiva), lo strumento di programmazione più completo per questo linguaggio. Scritto in Java da Alessandro Schillaci, questo programma multipiattaforma[3] ricalca sotto certi aspetti le orme di WIDE, ma è decisamente più completo e potente di quest'ultimo; l'unica nota "dolente" è che richiede, per essere eseguito, l'installazione del JRE (v1.4 o superiore) che può rallentare il sistema sui computer più vecchi[4].
Per quanto riguarda invece gli utenti Linux e Macintosh, le rispettive versioni di Frotz sono Frotz 2.42 per il primo, e MaxZip 1.78[5] per il secondo. Gargoyle (così come Splatterlight per il

[3] JIF funziona infatti su Windows, Linux e Linux-MacOs e supporta appieno sia Inform che Glulx. Ulteriori informazioni riguardo alla sua installazione, potete trovarle al paragrafo 5.2.
[4] Acronimo di Java Runtime Environment, è un file per Windows lungo circa 13 megabyte (la versione 1.4 o successiva per Linux la potete scaricare dal sito ufficiale della Sun), che permette di utilizzare qualsiasi programma scritto in Java sul vostro computer.
[5] Poiché questo programma può dare a volte dei problemi di visualizzazione delle lettere accentate, è vivamente consigliato usarlo da finestra terminale impostando il set dei caratteri Windows.

Macintosh), invece, è un programma che è in grado di eseguire delle avventure testuali scritte in vari formati (tra cui, ovviamente, lo Z-code); il risultato è davvero straordinario, grazie soprattutto all'utilizzo di un font proprietario caratterizzato da un'alta leggibilità[6].
Altri programmi possono essere reperiti all'indirizzo http://www.ifarchive.org/ e comunque, dal momento che Inform sotto questo aspetto è universale (è uguale a livello di codice su tutte le piattaforme che lo supportano), tutto quello che verrà detto dal secondo capitolo in poi andrà bene per tutti.

[6] Esiste anche Parchment, un interprete Z-code per i browser. Ulteriori informazioni potete trovarle sul sito ufficiale (http://code.google.com/p/parchment/) o sull'ottima guida d'utilizzo scritta dal bravissimo Paolo Lucchesi e liberamente consultabile all'indirizzo http://www.ifitalia.info/pmwiki/pmwiki.php?n=Guide.Parchment.

CAPITOLO 2

LE BASI DI INFORM

§2.1 A cosa serve questo capitolo?

Quando si vuole scrivere un'avventura testuale, bisogna prendere in considerazione la storia in sé ma anche il codice vero e proprio. Infatti, la sequenza di testi, domande e risposte che il vostro computer vi mostra a mano a mano che proseguite nel gioco, è il risultato di una sequenza d'istruzioni Inform che nel suo insieme costituisce il LISTATO o CODICE SORGENTE. Per scrivere un listato occorre però apprendere un insieme di regole relative alla programmazione (in questo caso specifiche su Inform, ma il discorso vale per qualsiasi linguaggio di programmazione). Ecco il motivo per cui ho scritto questo capitolo, che vi darà quindi le basi per scrivere poi un'avventura testuale vera e propria (argomento che sarà trattato nel terzo capitolo con la creazione della mini-avventura "Ruins").

OCCORRE TUTTAVIA CONSIDERARE CHE SI PUÒ COMINCIARE A SCRIVERE UN'AVVENTURA TESTUALE SENZA NECESSARIAMENTE SAPERE TUTTO QUELLO CHE VIENE QUAI DETTO (ne è un esempio l'INFORM BEGINNER GUIDE di Roger Firth e Sonja Kesserich, tradotto in italiano in versione stampabile o direttamente online all'indirizzo http://www.inform-italia.org/docs/gip); in poche parole, POTETE SALTARE DIRETTAMENTE AL CAPITOLO 3, RITORNANDO QUI SOLO PER APPRENDERE LE NOZIONI CHE VI SERVONO. A voi la scelta quindi: o prima (dal mio punto di vista il metodo migliore), o dopo (una volta che vi sarete impratichiti un po' di più con la programmazione delle avventure testuali con questo linguaggio).

§2.2 Benvenuto in Inform

Senza perdere tempo in ulteriori preamboli, direi che possiamo cominciare con il nostro primo programma[1]:

```
! Esempio sull'utilizzo dell'istruzione print - versione 1

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Stampa;
  ClearScreen(); ! pulisce lo schermo
  print "Benvenuto in Inform!";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Stampa();
     quit; ! fine del programma
];

Include "ItalianG";
```

Da WIDE o JIF, compilate il listato ed eseguitelo: se tutto è andato bene, sul video appare la seguente scritta:

```
Benvenuto in Inform!
```

il computer aspetta poi che l'utente abbia premuto un tasto qualsiasi per uscire e terminare il programma.

[1] Tutti i listati di questo capitolo si trovano nella directory Capitolo2 del file esercizi_manuale.zip.

Nonostante l'estrema semplicità di quest'ultimo (e non poteva essere altrimenti, visto che è il primo) si possono fare su di esso diverse considerazioni:

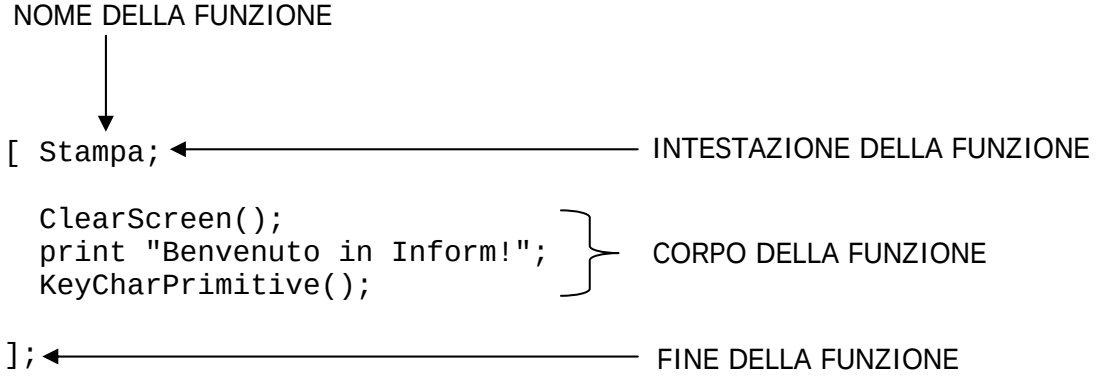
- in primis, possiamo dire che, tutte le volte che abbiamo un punto esclamativo seguito da un testo, siamo di fronte ad una riga di commento, molto utile per descrivere ad esempio cosa fa una determinata istruzione che possa in seguito servire a chi dovesse leggere il listato. I commenti, inoltre, vengono completamente ignorati dal compilatore e non hanno peso sul file eseguibile (potete scrivere tutti i commenti che volete, ma la dimensione del file eseguibile rimane sempre la stessa);
- l'istruzione print è, con molta probabilità, la più utilizzata nell'ambito della programmazione in Inform. In questo caso, la sua funzione è quella di stampare a video il testo racchiuso fra le virgolette, come d'altra parte è facilmente intuibile;
- il punto e virgola (";") finale è una regola basilare nella programmazione in questo linguaggio; infatti, salvo casi particolari, TUTTE LE ISTRUZIONI IN INFORM DEVONO TERMINARE CON UN PUNTO E VIRGOLA pena la comparsa di un messaggio d'errore durante la fase di compilazione;
- l'istruzione ClearScreen pulisce lo schermo, mentre l'istruzione KeyCharPrimitive legge un carattere dalla tastiera aspettando la pressione di un tasto qualsiasi da parte dell'utente;
- l'istruzione quit provoca l'immediata interruzione del programma e la conseguente chiusura di WinFrotz2002;
- per quanto riguarda la formattazione del codice, è possibile scrivere il listato nel seguente modo:

```
! Esempio sull'utilizzo dell'istruzione print – versione 1
Include "Parser"; Include "VerbLib"; Include "Replace";
[ Stampa; ClearScreen(); print "Benvenuto in Inform!";
KeyCharPrimitive(); ];[ Initialise; Stampa(); quit; ];
Include "ItalianG";
```

per il compilatore non cambia nulla, ma riuscite a capirci qualcosa da questa "macedonia"?
È meglio quindi mantenere un certo "ordine" all'interno di un nostro listato, in modo tale che risulti essere più comprensibile per noi stessi e per gli altri (soprattutto a distanza di tempo).

§2.3 Le funzioni

Sostanzialmente, possiamo definire una funzione come UNA SERIE D'ISTRUZIONI CONTENUTA ALL'INTERNO DI DUE PARENTESI QUADRE. Rifacendoci ancora una volta all'esempio precedente, nella funzione che segue possiamo distinguere un nome, un'intestazione, un corpo e una fine:



All'interno di un listato possono essere definite più funzioni, come avviene in questo secondo esempio:

```
! Esempio sull'utilizzo della funzione print – versione 2
```

```
Include "Parser";
Include "VerbLib";
Include "Replace";

[ Stampa1;
  ClearScreen(); ! pulisce lo schermo
  print "Benvenuto";
  Stampa2();
  Stampa3();
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Stampa2;
  print " in ";
];

[ Stampa3;
  print "Inform!";
];

[ Initialise;
     Stampa1();
     quit; ! fine del programma
];

Include "ItalianG";
```

il risultato è sempre lo stesso (la stampa a video della scritta “Benvenuto in Inform!”), ma cambia la modalità d’esecuzione con cui questa stampa avviene. In questo esempio il programma, partendo dalla funzione Initialise, esegue la chiamata alla funzione Stampa1 stampando a video la parola “Benvenuto”; segue poi la chiamata alla funzione Stampa2 (che stampa a video la parola “in” con tanto di spazi), il ritorno alla funzione Stampa, la chiamata alla funzione Stampa3 (che stampa a video la parola “Inform!”), il ritorno alla funzione Stampa, la lettura del carattere dalla tastiera e l’uscita dal programma. Possiamo quindi affermare che TUTTI I PROGRAMMI IN INFORM PARTONO DALLA FUNZIONE INITIALISE.

§2.4 Variabili numeriche e caratteri speciali

All’interno di un listato, vengono definiti dei valori che cambiano a seconda del verificarsi o meno di determinate condizioni: una variabile può essere paragonata ad una sorta di contenitore (per esempio una scatola) che contiene tutti questi valori. Ecco un esempio che illustra quanto detto:

```
! Esempio sull'utilizzo delle variabili numeriche

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Var_num();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];
```

```
[ Var_num numb;
   numb = 0;
   print "La variabile ~numb~ contiene il valore ", numb , ".^";
   numb = 1;
   print "Adesso la variabile ~numb~ contiene il valore ", numb ,".^";
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "ItalianG";
```

Nella funzione Var_num viene definita una variabile numb (che in questo caso funge anche da argomento della funzione) alla quale viene dapprima assegnato, tramite l'operatore =, il valore 0 e successivamente il valore 1.
In Inform, le variabili vengono usate in genere per la gestione dei numeri e dei singoli caratteri. Di default, il range per un valore numerico valido da assegnare ad una variabile varia da –32768 a 32767. Questo significa che È POSSIBILE ASSEGNARE UN VALORE COMPRESO TRA –32768 E 32767, MA NON MINORE DI –32768 E NON MAGGIORE DI 32767.
Ricordatevi inoltre, che PRIMA DI ESSERE UTILIZZATA, UNA VARIABILE DEVE SEMPRE AVERE ASSEGNATO UN VALORE INIZIALE (deve cioè essere INIZIALIZZATA) perché, nel momento in cui viene definita, il valore in essa contenuto può essere un numero qualsiasi che spesso e volentieri è al di fuori del range consentito. La conseguenza di tutto questo potrebbe essere davvero drastica, perché per il compilatore non ci sarebbe alcun tipo d'errore, ma il valore sarebbe completamente sballato.

▲Provate come esercizio a sostituire la riga numb = 1; con numb = 1234567; per vedere cosa succede.

In Inform esistono anche dei caratteri speciali che vengono stampati in un modo un po' diverso da quello usuale[2]. Nella riga print "Adesso la variabile ~numb~ contiene il valore ", numb ,".^"; ce ne sono ben due:

- la TILDE (~), che stampa a video le virgolette; si ottiene tenendo premuto il tasto ALT sinistro + i tasti 1, 2 e 6 del tastierino numerico premuti in rapida successione. Per stampare invece la tilde vera e propria occorre scrivere @@126 (ad esempio print "@@126";);
- il segno di RITORNO A CAPO o di NEWLINE (^); se non si dispone di una tastiera italiana, questo carattere è ottenibile tenendo premuto il tasto ALT sinistro + i tasti 9 e 4 del tastierino numerico premuti in rapida successione. Per stampare invece il carattere ^ vero e proprio occorre scrivere @@94.

Ne esistono ovviamente anche degli altri. I più usati sono[3]:

- @^ che pone l'accento circonflesso sulle lettere a, e, i, o, u, A, E, I, O, U (es.: print "@^e";);
- @c che pone la cediglia sulle lettere c o C;
- @: che pone la dieresi sulle lettere a, e, i, o, u, A, E, I, O, U;
- @' che pone l'accento acuto sulle lettere a, e, i, o, u, A, E, I, O, U (es.: print "perch@'e";);

[2] Un elenco completo dei caratteri stampabili è disponibile alla tabella 2B dell'Inform Designer Manual di Graham Nelson che potete consultare all'indirizzo http://www.inform-fiction.org/manual/html/tables.html#tbl2a. Jif, inoltre, dà la possibilità di utilizzare i caratteri normali (quelli cioè codificati nel codice ASCII) anziché quelli codificati nel codice ZSCII (ad esempio è al posto di @`e); questo è possibile tramite il mapping dei caratteri, una delle tante caratteristiche che lo differenziano da tutti gli altri programmi del genere.
[3] Inform supporta anche i caratteri Unicode (come ad esempio il simbolo dell'euro "€") che non fanno parte della tabella citata nella nota 2. Ulteriori informazioni potete trovarle al paragrafo 4.10.

- @` che pone l'accento grave sulle lettere a, e, i, o, u, A, E, I, O, U (es.: print "@`e";);
- @@92 che stampa a video il carattere \ (in inglese backslash);
- @@64 per stampare a video il carattere @;
- @LL per stampare a video il segno £;
- la parentesi graffa aperta { che si ottiene tenendo premuto il tasto ALT sinistro + i tasti 1, 2, 3 del tastierino numerico premuti in rapida successione;
- la parentesi graffa chiusa } che si ottiene tenendo premuto il tasto ALT sinistro + i tasti 1, 2, 5 del tastierino numerico premuti in rapida successione.

§2.5 Un po' di matematica non guasta mai

In Inform, ci sono quattro operatori aritmetici fondamentali:

- il segno di moltiplicazione (*) ⟶ a = 10 * 5;
- il segno di divisione (/) ⟶ a = 10 / 5;
- il segno di addizione (+) ⟶ a = 10 + 5;
- il segno di sottrazione (–) ⟶ a = 10 – 5;

Quando un'operazione aritmetica richiede l'utilizzo di più operatori aritmetici, ci troviamo di fronte a un'espressione ⟶ a = (((10*5) + (10-3)) * 6) + (-9);
In questo caso bisogna fare molta attenzione all'ordine con cui gli operatori lavorano. Infatti, se scriviamo 4 + 2 * 8 e 4 * 2 + 8, avremo come risultato 20 nel primo caso e 16 nel secondo (vengono cioè eseguite, in ordine di priorità, prima le moltiplicazioni e poi le addizioni).
Possiamo quindi affermare che:

1. GLI OPERATORI ARITMETICI LAVORANO IN ORDINE DI PRIORITÀ: IN UN'ESPRESSIONE ARITMETICA VIENE ESEGUITA PER PRIMA LA MOLTIPLICAZIONE E, A SEGUIRE, LA DIVISIONE, L'ADDIZIONE E LA SOTTRAZIONE. NE CONSEGUE CHE:
2. IN UN'ESPRESSIONE ARITMETICA LE SINGOLE OPERAZIONI DEVONO ESSERE SEMPRE RACCHIUSE DA DELLE PARENTESI TONDE;
3. A OGNI PARENTESI TONDA APERTA, NE DEVE SEMPRE CORRISPONDERE UNA TONDA CHIUSA.

Se vogliamo quindi sommare 4 + 2 e moltiplicare poi il risultato ottenuto per 8, avremo: (4 + 2) * 8

In Inform è anche possibile usare l'operatore % per ottenere il resto di una divisione. Quindi:

- 15/5 darà come risultato 3, mentre 15%5 darà come risultato 0;
- 17/5 darà come risultato 3, mentre 17%5 darà come risultato 2;
- non è invece possibile dividere per 0 (come d'altra parte la stessa matematica ci insegna) così come non è possibile usare i numeri con la virgola (ad esempio 6.34).

È anche possibile operare sui valori contenuti all'interno delle variabili. Ecco dei possibili esempi:

```
a = b + 20;      b = (c * 60) + (b / 30);      d = (b + 7) * (b - 8);
```

Spesso poi, il contenuto di una variabile può essere incrementato o decrementato di un certo valore, come avviene nei seguenti esempi:

```
a = a + 2;          b = b + 5;          b = b – a          b = b – (a + 3);
```

Quindi, supponendo che a abbia il valore 12 e che b abbia il valore 6, avremo:

a = a + 2	b = b – a	b = b – (a + 3)
a = 12 + 2	b = 6 – 12	b = 6 – (12 + 3)
a = 14	b = –6	b = –9

Se vogliamo poi incrementare o decrementare il valore contenuto in una variabile di 1, esistono delle forme abbreviate[4]:

- a++ ⟶ viene PRIMA assegnato il nuovo valore alla variabile e DOPO viene incrementato di 1;
- ++a ⟶ viene PRIMA incrementato di 1 il nuovo valore e DOPO viene assegnato alla variabile;
- a-- ⟶ viene PRIMA assegnato il nuovo valore alla variabile e DOPO viene decrementato di 1;
- --a ⟶ viene PRIMA decrementato di 1 il nuovo valore e DOPO viene assegnato alla variabile;

§2.6 Passaggio dei parametri e ritorno dei valori

Ora che sappiamo cosa sono le variabili, possiamo esaminare il seguente esempio:

```
! Esempio sul passaggio dei parametri  - versione 1

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip x n;
  x = 5;
  ClearScreen(); ! pulisce lo schermo
  n = Quadrato(x); print "Il quadrato di ", x , " @`e ", n , ".^";
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Quadrato number;
  return number*number;
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "ItalianG";
```

Il primo cambiamento possiamo notarlo subito all'inizio della funzione Funz_princip: gli argomenti questa volta sono due (x e n). Nulla di strano, perché UNA QUALSIASI FUNZIONE IN INFORM PUÒ AVERE UN MASSIMO DI 15 ARGOMENTI, SEPARATI TRA DI LORO DA ALMENO UNO SPAZIO.
Quando poi viene chiamata la funzione Quadrato viene anche passato il valore tra parentesi (detto parametro) contenuto all'interno della variabile x. Ecco in sostanza quello che accade: alla riga n = Quadrato(x); il programma salta alla funzione Quadrato e il valore contenuto nella variabile x (in questo caso 5) viene assegnato alla variabile number (definita nella funzione Quadrato). Viene poi eseguita l'elevazione al quadrato e il valore ottenuto (25) viene restituito, mediante l'istruzione return, e assegnato alla variabile n della funzione Funz_princip.

[4] a++ e ++a (così come a-- e --a) hanno esattamente lo stesso effetto sulla variabile a e la differenza sta solo nel valore delle espressioni.

Se qualcuno di voi ha le idee confuse, si concentri su quello che sto per dire adesso: UNA VARIABILE PUÒ ESSERE LOCALE (VIENE CIOÈ VISTA SOLO ALL'INTERNO DELLA FUNZIONE IN CUI È DEFINITA) O GLOBALE (VIENE CIOÈ VISTA IN QUALSIASI PUNTO DEL PROGRAMMA).
Proviamo a riscrivere la funzione Quadrato in questo modo:

```
[ Quadrato number;
   return x*x;
];
```

e compiliamo nuovamente il programma: il file eseguibile non viene creato, perché sono presenti ben due errori, nell'ordine:

1. "No such constant as x" che, tradotto in italiano, significa "Non c'è nessuna variabile denominata x". Questo accade perché la variabile locale x è stata definita nella funzione Funz_princip e appartiene ad essa soltanto. Ecco quindi il motivo per cui ho definito una nuova variabile number all'interno della funzione Quadrato; se non lo avessi fatto, il valore 5 sarebbe andato irrimediabilmente perso nel momento in cui il programma sarebbe passato dalla funzione Funz_princip alla funzione Quadrato.
   Occorre anche notare che se definisco questa funzione nel seguente modo:

   ```
   [ Quadrato x;
      return x*x;
   ];
   ```

   il programma funziona perfettamente, ed è normale che sia così. Infatti, la variabile x definita all'interno della funzione Quadrato è diversa dalla variabile x definita all'interno della funzione Funz_princip.
2. "Local variable number declared but not used" che, tradotto in italiano, significa "La variabile locale number è stata dichiarata ma non usata". Ecco dunque un'altra regola sacrosanta di questo linguaggio: TUTTE LE VARIABILI DA NOI DICHIARATE, DEVONO ESSERE USATE. Nella funzione Quadrato viene definita la number, ma al suo posto viene usata la x: ecco il motivo dell'errore.

Qualcuno di voi potrebbe però chiedersi: perché, al posto di usare più variabili locali non ne uso una sola globale? Perché di funzioni in un'avventura testuale ce ne sono moltissime e usare solo delle variabili globali diventa veramente un azzardo: si genera troppa confusione e aumenta il rischio che Inform scambi un valore per un altro. Comunque, a scopo puramente didattico, ecco come usare una variabile globale con l'esempio di prima:

```
! Esempio sul passaggio dei parametri  - versione 2

Include "Parser";
Include "VerbLib";
Include "Replace";

Global x;

[ Funz_princip n;
  x = 5;
  ClearScreen(); ! pulisce lo schermo
  n = Quadrato(); print "Il quadrato di ", x , " @`e ", n , ".^";
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];
```

```
[ Quadrato;
  return x*x;
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "ItalianG";
```

Come potete vedere, la variabile globale x , essendo stata dichiarata al di fuori di una qualsiasi funzione[5], può essere vista, riconosciuta e modificata in qualsiasi punto del programma.

§2.7 L'input dei dati numerici: l'istruzione Getnumber

Riesaminando per un attimo l'esempio del paragrafo precedente:

```
[ Funz_princip x n;
  x = 5;
  ClearScreen(); ! pulisce lo schermo
  n = Quadrato(x); print "Il quadrato di ", x , " @`e ", n , ".^";
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Quadrato number;
  return number*number;
];
```

notiamo che, se vogliamo elevare al quadrato un numero diverso da 5, dobbiamo modificare il valore di x (ad esempio x = 10;) e poi ricompilare ed eseguire il tutto. Decisamente scomodo, non vi pare?
Inform stesso (che pur essendo un linguaggio di programmazione strepitoso è ben lungi dall'essere un prodotto perfetto), non ci viene minimamente incontro in un caso come questo, perché non esiste di default un'istruzione che chieda all'utente di inserire un numero dalla tastiera.
Fortunatamente però, è possibile includere in un programma delle nuove funzioni che possono risolvere qualsiasi tipo di problema (rimanendo ovviamente relegati nell'ambito delle avventure testuali). Vediamo ora come possiamo scrivere un esempio che possa fare al caso nostro:

```
! Esempio sul passaggio dei parametri  - versione 3

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip x n;
  ClearScreen(); ! pulisce lo schermo
  print "^Inserire il numero da elevare al quadrato: ";
  x = GetNumber(2);
  n = Quadrato(x); print "Il quadrato di ", x ," @`e ", n ,".^";
  print "^Premi un tasto per uscire^";
```

[5] O di un qualsiasi oggetto, come vedremo meglio nel capitolo 3.

```
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Quadrato number;
  return number*number;
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

L'istruzione include "Io"; dice al compilatore che una parte del codice da compilare è contenuto nel file Io.h, contenente a sua volta l'istruzione Getnumber da me scritta (che acquisisce in input il numero digitato dall'utente). Il file deve trovarsi nella directory "C:\inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando JIF) e potete reperirlo nella directory inform\Capitolo2\libraries.
GetNumber ha bisogno di sapere da quante cifre è composto il numero che l'utente deve digitare (1, 2, 3, 4 o 5), e il valore che ritorna (il numero stesso) deve essere contenuto in una variabile (altrimenti viene perso, ricordate?). Non sono inoltre ammessi i numeri negativi e con la virgola.
Questa istruzione è quindi in grado di elaborare dei numeri che vanno da un minimo di 0 a un massimo di 32767, un range più che sufficiente nell'ambito delle avventure testuali.

§2.8 If (condition) then... else...

Nella stesura di un programma, le scelte sono quasi sempre d'obbligo, proprio come avviene in questo esempio:

```
! Esempio sull'utilizzo dell'istruzione IF THEN – versione 1

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Domanda();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Domanda x;
  print "^Ogni quanti anni compare la cometa di Halley? ";
  x = GetNumber(2);
  if (x == 76) print "La risposta @`e esatta!!!^";
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];
```

```
Include "Io";
Include "ItalianG";
```

Tradotta in italiano, l'istruzione if (condition) then suona più o meno così: se (condizione) allora. In questo caso, se il numero introdotto dall'utente è proprio 76, allora il programma stampa a video "La risposta è esatta!!!", altrimenti prosegue senza visualizzare nulla.
Fate molta attenzione a non confondere l'operatore = con l'operatore ==; il primo, come già sapete da quanto visto finora, assegna un certo valore a una variabile, mentre il secondo verifica che un certo valore (in questo esempio 76) sia **UGUALE** a un altro (in questo caso il valore contenuto nella variabile x, ovvero il numero digitato dall'utente mediante l'istruzione Getnumber).
Gli altri operatori sono:

(a ~= b) ⟶ a è diverso da b
(a > b) ⟶ a è maggiore di b
(a < b) ⟶ a è minore di b
(a >= b) ⟶ a è maggiore o uguale a b
(a <= b) ⟶ a è minore o uguale a b

È anche possibile usarli insieme, ma in questo caso bisogna utilizzare anche gli OPERATORI LOGICI, nell'ordine:

- && detto anche "AND" ⟶ es: if ((a > 5) && (b < 10)) ⟶ restituisce true (la condizione è cioè soddisfatta) se il valore contenuto nella variabile a è maggiore di 5 **E** se il valore contenuto nella variabile b è minore di 10;
- || detto anche "OR" ⟶ es: if ((a > 5) || (b < 10)) ⟶ restituisce true se il valore contenuto nella variabile a è maggiore di 5 **OPPURE** se il valore contenuto nella variabile b è minore di 10. Questo operatore non deve essere confuso con l'operatore (non logico) or che viene usato per testare più valori in una volta sola (ad es.: if (a == 2 or 3 or 4) print...);
- ~~ detto anche "NOT" ⟶ es: if (~~ (a > 5)) ⟶ restituisce true se il valore contenuto nella variabile a è minore di 5 (la condizione viene cioè **NEGATA** e di conseguenza invertita).

Così com'è, il nostro esempio stampa un messaggio nel caso in cui l'utente risponda correttamente alla domanda. E se invece sbaglia?
Non è un'idea così malvagia, quella di modificare il nostro programma in modo tale che stampi un messaggio anche nel caso di una risposta errata da parte dell'utente. Ecco una possibile soluzione:

```
! Esempio sull'utilizzo dell'istruzione IF THEN - versione 2

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Domanda();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Domanda x;
  print "^Ogni quanti anni compare la cometa di Halley? ";
  x = GetNumber(2);
  if (x == 76) print "La risposta @`e esatta!!!^";
```

```
  else print "Spiacente, la risposta non @`e esatta...^";
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

di nuovo c'è solo l'istruzione else, che in italiano significa altrimenti e viene usata solo insieme all'istruzione if-then. Ecco quello che accade: se il numero digitato dall'utente è uguale a 76 viene stampato il messaggio "La risposta è esatta!!!", mentre se è diverso da 76 viene stampato il messaggio "Spiacente, la risposta non è esatta...".

§2.9 While e do...until

In Inform, così come in qualsiasi altro linguaggio di programmazione, oltre alle scelte esistono anche i cicli (o loop). Il primo di questi è while (condition) e il suo funzionamento è illustrato nel seguente esempio:

```
! Esempio sull'utilizzo del ciclo WHILE

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Domanda();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Domanda x;
  x = 0; ! per inizializzare la variabile x al valore zero
  while (x ~= 76)
  {
     print "^Ogni quanti anni compare la cometa di Halley? ";
     x = GetNumber(2);
     if (x == 76) print "La risposta @`e esatta!!!^";
     else print "Spiacente, la risposta non @`e esatta...^";
  }
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

Nella funzione Domanda, viene definito il ciclo while (condizione), la cui funzione è quella di eseguire una serie di istruzioni, racchiusa da dalle parentesi graffe, fino a quando il valore contenuto nella variabile x è diverso da 76. In questo modo, il ciclo termina solo se x è uguale a 76 (se cioè la risposta esatta), altrimenti va avanti all'infinito (richiede tutte le volte la stessa domanda).

Il ciclo do...until (esegui... fino a quando vale la condizione) è simile al ciclo while ma, a differenza di quest'ultimo, esegue le istruzioni in esso contenute almeno una volta:

```
! Esempio sull'utilizzo del ciclo DO...UNTIL

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Domanda();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Domanda x;
  do
  {
     print "^Ogni quanti anni compare la cometa di Halley? ";
     x = GetNumber(2);
     if (x == 76) print "La risposta @`e esatta!!!^";
     else print "Spiacente, la risposta non @`e esatta...^";
  }
  until (x == 76);
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

Notate però, che nell'istruzione until la condizione è esattamente opposta a quella del ciclo while; infatti, mentre in quest'ultimo il ciclo va avanti fino a quando il valore contenuto nella variabile x è diverso da 76, nel ciclo do-until le istruzioni racchiuse nelle parentesi graffe vengono eseguite fino a quando il contenuto della variabile x è uguale 76. Il risultato è lo stesso, ma la strada intrapresa è esattamente quella opposta.

§2.10 For, break e continue

Un altro ciclo da esaminare è il for. Ecco un esempio che ne illustra il funzionamento:

```
! Esempio sull'utilizzo del ciclo FOR – versione 1

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Domanda();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Domanda x count;
  for (count = 0:  count<5: count++)
  {
     print "^", count , "^";
     print "Ogni quanti anni compare la cometa di Halley? ";
     x = GetNumber(2);
     if (x == 76)
     {
        print "La risposta @`e esatta!!!^";
        count = 5;
     }
     else print "Spiacente, la risposta non @`e esatta...^";
  }
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

Questo ciclo è suddiviso in tre parti; la prima (count = 0) assegna il valore 0 alla variabile count; la seconda (count < 5) fa in modo che il ciclo vada avanti fino a quando la variabile count contiene un valore minore di 5; la terza (count++) incrementa di 1 il valore di count fino a quando non arriva a essere uguale a 5. A quel punto il programma termina.
Quando l'utente risponde correttamente alla domanda prima che i cinque tentativi a disposizione siano terminati, alla variabile count viene assegnato il valore 5, che fa così terminare il ciclo e di conseguenza anche il programma. Senza questo piccolo stratagemma, se l'utente indovina ad esempio la risposta al secondo tentativo, il ciclo va avanti ancora tre volte (e viene quindi richiesta la domanda per altre tre volte).
Anche il comando if ha adesso delle istruzioni racchiuse da delle parentesi graffe; questo accade perché le istruzioni che gli appartengono sono più di una, al contrario di quello che avviene per l'istruzione else.

Il comando break si usa per fermare un ciclo, qualunque sia la sua condizione:

```
! Esempio sull'utilizzo del ciclo FOR – versione 2

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Domanda();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Domanda x count;
  for (count=0:  count<5: count++)
  {
     print "^", count , "^";
     print "Ogni quanti anni compare la cometa di Halley? ";
     x = GetNumber(2);
     if (x == 76)
     {
        print "La risposta @`e esatta!!!^";
        break;
     }
     else print "Spiacente, la risposta non @`e esatta...^";
  }
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

il risultato è esattamente identico a quello di prima, solo questa volta abbiamo usato l'istruzione break al posto di count=5. Un altro tipico utilizzo di questa istruzione si ha quando abbiamo a che fare con un ciclo infinito:

```
! Esempio sull'utilizzo del ciclo FOR – versione 3

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Domanda();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Domanda x;
  for ( : : )
```

```
 {
     print "Ogni quanti anni compare la cometa di Halley? ";
     x = GetNumber(2);
     if (x == 76)
     {
        print "La risposta @`e esatta!!!^";
        break;
     }
     else print "Spiacente, la risposta non @`e esatta...^";
  }
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

In questo esempio, il computer continua a ripetere la domanda fino a quando l'utente non dà la risposta giusta. Il "merito" di tutto questo è dato dal ciclo for definito senza condizioni tra le sue parentesi (for (: :)).
Ecco ora un altro esempio che utilizza invece l'istruzione continue:

```
! Esempio che utilizza l'istruzione continue

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip k;
  ClearScreen(); ! pulisce lo schermo
  for (k=1: k<=10: k++)
  {
     if (k == 5)
     {
        k = 8;
        continue;
     }
     print k, " ";
  }
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "ItalianG";
```

quando la variabile k è uguale a 5, le viene assegnato il valore 8; il ciclo continua, saltando però i numeri 5, 6 e 7 per k == 5. Il risultato in output (quello che appare sul video) è: 1 2 3 4 9 10.

§2.11 L'utilizzo delle costanti

Possiamo raffinare ulteriormente il nostro esempio sull'istruzione for, utilizzando una costante per identificare il numero dei tentativi massimi possibili:

```
! Esempio che utilizza una costante

Include "Parser";
Include "VerbLib";
Include "Replace";

Constant TENTATIVI 5;

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Domanda();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Domanda x count;
  for (count=0: count<TENTATIVI: count++)
  {
     print "^Ogni quanti anni compare la cometa di Halley? ";
     x = GetNumber(2);
     if (x == 76)
     {
        print "La risposta è esatta!!!^";
        break;
     }
     else print "Spiacente, la risposta non @`e esatta...^";
  }
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

in questo modo, se si vuole aumentare o diminuire il numero dei tentativi, basta andare a modificare il valore della costante, ad esempio:

```
Constant TENTATIVI 7;
```

Forse, alcuni di voi non riescono ancora a comprendere il vantaggio di utilizzare una costante in un listato così semplice. Immaginate allora, cosa accade in un listato più complesso se devo andare a cambiare i valori di tutte le condizioni di controllo dei cicli. Una cosa davvero noiosa, soprattutto quando ci sono più cicli che terminano dopo lo stesso numero di tentativi. DAL MOMENTO CHE UNA COSTANTE È DICHIARATA AL DI FUORI DI UNA QUALSIASI FUNZIONE O ALTRA STRUTTURA, È GLOBALE, E QUINDI VISIBILE IN TUTTO IL PROGRAMMA: con un solo cambiamento (quello del valore della costante stessa), metto a posto tutte le funzioni e tutti i cicli che si rifanno a quel valore. Insomma, è proprio il caso di dire che con un piccione prendo più di una fava…

Attenzione però a non confondere una costante con una variabile: UNA COSTANTE È UN VALORE FISSO, CHE NON MUTA DURANTE L'ESECUZIONE DEL PROGRAMMA, MENTRE UNA VARIABILE SÌ.

§2.12 Switch... case

Questo ciclo permette di operare delle scelte multiple. Ecco un esempio che ne illustra il funzionamento:

```
! Esempio sull'utilizzo del ciclo SWITCH CASE

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Domanda();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Domanda x;
  do
  {
     print "^Scrivi ogni quanti anni compare la cometa di Halley:";
     print "^^50  60  70  76^^";
     print "? "; x = Getnumber(2);
     switch(x)
     {
       50, 60: print "Assolutamente no!!!^";
       70: print "No, ma ci sei quasi...^";
       76: print "Bravo, la risposta @`e esatta...^";
       default: print "Sii serio. Il valore da te scritto non @`e tra
                       quelli elencati.^";
     }
   }
   until (x == 76);
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

Abbiamo la nostra solita domanda sulla cometa di Halley, ma questa volta c'è una novità: ora il computer propone quattro possibili risposte, e l'utente deve indovinare quella giusta. A differenza degli esempi precedenti però, ora viene stampato un messaggio per ogni risposta data, compreso un valore che non rientra nei quattro suggeriti.
Sottolineo ancora una volta l'importanza della formattazione del codice perché, rispettando un certo ordine di spazi e di tabulazioni, è possibile riuscire a capire quali sono le istruzioni che appartengono a un certo ciclo piuttosto che a un altro. Come potete vedere, dopo la riga switch(x)

segue un elenco dei valori con, per ognuno, la stampa del relativo messaggio. La condizione default: è quella che permette al computer di capire se l'utente ha digitato un valore suggerito o no; essa dice al compilatore: PER TUTTO QUELLO CHE È DIVERSO DAI VALORI ELENCATI, SCRIVI "Sii serio. Il valore da te scritto non è tra quelli elencati.". Capito dove sta il trucco?

Notate che lo stesso risultato si può ottenere anche con l'istruzione if-then:

```
[ Domanda x;
  do
  {
     print "^Scrivi ogni quanti anni compare la cometa di Halley:";
     print "^^50  60  70  76^^";
     print "? "; x = Getnumber(2);
     if (x == 50 or 60) print "Assolutamente no!!!^";
     else
     {
        if ( x ==  70) print "No, ma ci sei quasi...^";
        else
        {
            if (x == 76) print "Bravo, la risposta @`e esatta...^";
            else
            {
               print "Sii serio. Il valore da te scritto non @`e tra
                      quelli elencati.^";
            }
        }
     }
  }
  until (x == 76);
];
```

tutto funziona alla perfezione (provare per credere), ma è proprio il caso di dire che con l'istruzione switch-case le cose si semplificano di molto. Nell'ambito della programmazione, più diventerete esperti e più vi renderete conto che esisteranno diverse strade per arrivare alla soluzione di un problema; alcune saranno migliori, altre no. Sarete voi a deciderlo.

§2.13 Salti ed etichette

In Inform, esiste anche l'istruzione jump in grado di controllare l'esecuzione del programma, come accade nel seguente esempio:

```
! Esempio sull'utilizzo dell'istruzione JUMP

Include "Parser";
Include "VerbLib";
Include "Replace";

Constant TENTATIVI 5;

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  Domanda();
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
```

```
];

[ Domanda x count;
  count = 0;

  .Inizio;
  print "^Ogni quanti anni compare la cometa di Halley? ";
  x = GetNumber(2);
  if (x == 76)
  {
     print "La risposta @`e esatta!!!^";
     count = TENTATIVI;
  }
  else print "Spiacente, la risposta non @`e esatta...^";
  count++; if (count < TENTATIVI) jump Inizio;
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

L'istruzione .Inizio; è un'etichetta (in inglese label) alla quale l'istruzione jump fa sempre riferimento. Alla fine della funzione Domanda, l'istruzione if (count < TENTATIVI) jump Inizio; dice al compilatore: SE IL VALORE CONTENUTO NELLA VARIABILE COUNT È MINORE DEL VALORE DELLA COSTANTE TENTATIVI, **VAI** ALL'ETICHETTA INIZIO, ALTRIMENTI PROSEGUI.
In questo modo abbiamo simulato un ciclo do-until, ovviamente a scopo puramente didattico e dimostrativo; non ha alcun senso, infatti, simulare qualcosa che esiste già. Questo stratagemma veniva utilizzato moltissimo con alcuni vecchi Basic degli anni '80, che di fatto non avevano i cicli while e do…until e dovevano in qualche modo "arrangiarsi".
È bene comunque sapere che Inform mette a disposizione anche questa possibilità, perché, in alcune funzioni particolarmente complesse, potrebbe rivelarsi una valida scappatoia.

§2.14 Caratteri e stringhe

Finora abbiamo visto come gestire i dati numerici (10, 13, 56 e così via), ma esistono anche i dati alfabetici (le sole lettere, come ad esempio a, b, c, casa, papà, ecc.) e i dati alfanumerici (le lettere e i numeri, come ad esempio alpha202, omega3, ecc.).
In Inform una variabile, oltre ai valori numerici, può contenere anche dei caratteri e perfino delle stringhe (sequenze di numeri e lettere, come ad es. "Ogni mattina mi sveglio alle 6:00"):

```
! Esempio sull'utilizzo di variabili con dati non numerici - versione 1

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip carattere parola stringa;
  ClearScreen(); ! pulisce lo schermo
  carattere = 'a'; parola = "terrazzo"; stringa = "I miei 9 gatti";
  print (char) carattere, "^" , (string) parola, "^" , (string)
```

```
        stringa, "^";
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "ItalianG";
```

come potete vedere, il computer visualizza “a”, “terrazzo”, “I miei 9 gatti”, esattamente quello che ci si aspetta. Questa volta entrano però in gioco le funzioni di stampa (char) e (string); la prima, dice all’istruzione print di stampare a video il contenuto della variabile carattere, la seconda il contenuto delle variabili parola e stringa. Altra cosa da notare, almeno per quelli un po’ più esperti nella programmazione, è che IN INFORM (PURTROPPO) LE VARIABILI NON SI DIFFERENZIANO PER TIPO. Osserviamo ora quest’altro esempio:

```
! Esempio sull'utilizzo di variabili con dati non numerici – versione 2

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip carattere parola stringa car;
  ClearScreen(); ! pulisce lo schermo
  carattere = 'a'; parola = "terrazzo"; stringa = "La mia casa";
  print (char) carattere, "^" , (string) parola, "^" , (string)
        stringa, "^";
  print "^Premi il tasto q per uscire^";
  do
  {
     car = Presskey();
  }
  until (car == 'q');
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

l’istruzione Presskey da me scritta legge il carattere del tasto premuto dall’utente e ne restituisce il valore alla variabile car. Se il tasto premuto corrisponde alla lettera q, il programma termina, altrimenti richiede la pressione di un altro tasto. Potete testare tutte le lettere che volete, ma se si volessero usare dei tasti “particolari” come ad esempio la barra spaziatrice o ESC?
In questo caso bisogna ricorrere all’utilizzo dei codici ZSCII (attenzione, non ASCII), perché a ogni simbolo rappresentato sulla tastiera, corrisponde un determinato codice numerico indicante il carattere, e varia a seconda del sistema operativo e dei programmi che si stanno usando. Sul mio computer (un PC equipaggiato con Windows XP), ecco a quali codici corrispondono i seguenti tasti:

- 8 per il tasto DELETE
- 13 per il tasto INVIO
- 27 per il tasto ESC
- 32 per la BARRA SPAZIATRICE

volendo quindi terminare il programma premendo la barra spaziatrice, occorre modificare il ciclo do-until nel seguente modo:

```
print "^Premi spazio per uscire^";
  do
  {
    car = Presskey();
  }
  until (car == 32);
];
```

se usate Linux o il Mac-OS, per sapere a quali codici corrispondono i vostri tasti, potete inserire una print car; sotto la riga car = Presskey();. Provate a premere un tasto che sia diverso dalla barra spaziatrice, e vedete quale numero appare sul video: quello è il codice effettivo da usare nel test.
Sempre per quanto riguarda la gestione dei caratteri e delle stringhe, Inform mette a disposizione alcune istruzioni molto utili, rappresentate nell'esempio che segue:

```
! Esempio sull'utilizzo di variabili con dati non numerici - versione 3

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip car1 car2 stringa lungh;
  ClearScreen(); ! pulisce lo schermo

  car1 = 'a'; car2 = UpperCase(car1); print (char) car2, "^";
  car1 = 'A'; car2 = LowerCase(car1); print (char) car2, "^";

  stringa = "ciao Andrea!";
  print (string) stringa, "^";
  lungh = Length(stringa); print "Lunghezza: ", lungh, "^";
  print (Cap) stringa, "^";

  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "ItalianG";
```

Il risultato che si ottiene è il seguente:

```
A
a
ciao Andrea!
```

```
Lunghezza: 12
Ciao Andrea!
```

L'istruzione UpperCase converte il carattere passatole come parametro da minuscolo a maiuscolo, mentre l'istruzione LowerCase fa esattamente l'opposto (converte cioè un carattere da maiuscolo a minuscolo). L'istruzione Length restituisce la lunghezza di una stringa passatale come parametro (restituisce, cioè, il numero totale di caratteri che compongono quest'ultima) mentre la funzione di stampa (Cap) stampa a video la stringa con la prima lettera maiuscola. Concludiamo ora il paragrafo con quest'ultimo esempio:

```
! Esempio sull'utilizzo di variabili con dati non numerici - versione 4

Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip stringa1 stringa2 car;
  ClearScreen(); ! pulisce lo schermo
  stringa1 = "Ciao Andrea!"; stringa2 = "Ciao Andrea!";
  print (string) stringa1 , "  ", (string) stringa2, "^";
  if (stringa1 == stringa2) print "Le due stringhe sono uguali.^";
  print "^Premi il tasto q per uscire^";

  do
  {
     car = Presskey();
  }
  until (car == 'q');
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

le stringhe sono uguali, ma la condizione dell'istruzione if viene ignorata. A questo punto, le sole variabili non bastano più, e bisogna passare ai vettori (o in inglese array).

§2.15 I vettori

Se siete riusciti ad arrivare vivi fino a qui, devo farvi davvero i miei complimenti. Adesso però, prima di affrontare l'argomento di questo paragrafo, vi consiglio vivamente di bere un po' d'acqua, rilassarvi completamente e magari uscire per un paio d'ore con la vostra moglie o fidanzata (e se siete dei single con qualche amica: non si sa mai, potrebbe nascere una nuova relazione amorosa).
Vi siete rilassati? Ok. UN VETTORE PUÒ ESSERE CONSIDERATO COME UN INSIEME DI VARIABILI INDICIZZATE e in Inform può essere definito in questo modo:

```
Array nome_array->lunghezza_max_array;
```

Scrivendo ad esempio Array numeri->20; abbiamo a disposizione 20 celle (che equivalgono di fatto a delle variabili) alle quali è possibile accedere tramite un indice:

```
! Esempio sull'utilizzo dei vettori numerici - versione 1
```

```
Include "Parser";
Include "VerbLib";
Include "Replace";

Constant LUNGH_MAX 10;

Array numeri->LUNGH_MAX;

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  LoadArray(numeri);
  PrintNumericArray(numeri, LUNGH_MAX);
  print "^^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ LoadArray the_array i;
  for (i=1: i<=LUNGH_MAX: i++) the_array->(i - 1) = i;
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

All'inizio, il vettore numeri si presenta così:

0	1	2	3	4	5	6	7	8	9
?	?	?	?	?	?	?	?	?	?

abbiamo 10 celle che devono ancora essere inizializzate (non sappiamo quindi cosa contengono, da qui il ?). I numeri che vanno da 0 a 9 sono gli indici delle celle (uno per ogni cella). Dopo la chiamata alla funzione LoadArray ecco come si ripresenta il nostro vettore:

i	0	1	2	3	4	5	6	7	8	9
the_array->i	1	2	3	4	5	6	7	8	9	10

Nel ciclo for di questa funzione, alla variabile i viene assegnato l'indice e l'istruzione:

```
the_array->(i - 1) = i;
```

fa riferimento alla cella corrispondente dell'indice (i – 1), e le assegna il valore della variabile i. Ricordatevi sempre che IN INFORM, L'INDICE DELLA PRIMA CELLA DI UN VETTORE NON È 1, MA 0. Ecco perché ho messo (i – 1) al posto di i: se non lo avessi fatto, il contenuto della decima cella sarebbe stato 9 anziché 10.
Volendo semplificare un po' le cose, possiamo fare in modo che il vettore numeri sia inizializzato al momento della sua dichiarazione:

```
! Esempio sull'utilizzo dei vettori numerici - versione 2

Include "Parser";
```

```
Include "VerbLib";
Include "Replace";

Constant LUNGH_MAX 10;

Array numeri-> 1 2 3 4 5 6 7 8 9 10;

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  PrintNumericArray(numeri, LUNGH_MAX);
  print "^^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

In questo modo, la funzione LoadArray non serve più, mentre la costante LUNGH_MAX continua a fungere da parametro all'istruzione PrintNumericArray (che, come dovreste ormai aver capito, stampa a video il contenuto di un vettore numerico).

* *

Ecco adesso, un esempio che usa un vettore con le stringhe:

```
! Esempio sull'utilizzo dei vettori con le stringhe - versione 1

Include "Parser";
Include "VerbLib";
Include "Replace";

Array nome buffer "Massimo è andato a casa";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  PrintStringArray(nome);
  print "^^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

abbiamo ora un vettore nome di tipo stringa che dopo l'istruzione:

```
Array nome buffer "Massimo è andato a casa";
```

si presenta così:

0	1	2	3	4	5	6	7	8	9	10	...
0	23	M	a	s	s	i	m	o		è	...

La cella numero 2 (nome->1) contiene il numero 23 che corrisponde alla lunghezza della stringa in esso contenuta. La stringa vera e propria comincia quindi dalla cella successiva (nome->2).
Nell'esempio appena visto, abbiamo assegnato una stringa a un vettore al momento della dichiarazione di quest'ultimo. Tuttavia, in un programma vero e proprio (e quindi in un'avventura testuale), difficilmente si utilizza un vettore una sola volta. Come si può, cioè, far sì che sia possibile assegnare al vettore nome più stringhe? Osserviamo l'esempio che segue:

```
! Esempio sull'utilizzo dei vettori con le stringhe - versione 2

Include "Parser";
Include "VerbLib";
Include "Replace";

Constant LUNGH_MAX 33;

Array nome->LUNGH_MAX;

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo

  PrintToBuffer(nome, LUNGH_MAX, "Massimo è andato a casa");
  PrintStringArray(nome); print "^";
  PrintToBuffer(nome, LUNGH_MAX, "Massimo sta dormendo");
  PrintStringArray(nome);

  print "^^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

Come potete facilmente vedere, è l'istruzione PrintToBuffer che effettua l'assegnamento da noi voluto. PrintToBuffer ha però bisogno di tre parametri obbligatori: il nome dell'array (nome), la sua lunghezza (la costante LUNGH_MAX o, se preferite, 33) e la stringa vera e propria racchiusa tra le virgolette. RICORDATEVI SEMPRE CHE NON POTETE MAI USARE UNA STRINGA CHE SUPERI LA LUNGHEZZA MASSIMA DEL VETTORE DESTINATO A CONTENERLA; DOVETE INOLTRE AGGIUNGERE 3 NUMERI ALLA LUNGHEZZA MASSIMA DELLA VOSTRA STRINGA FINO A UN MASSIMO DI 160 CARATTERI. In poche parole, in questo esempio non potete inserire una stringa lunga più di 30 caratteri, e la lunghezza del vettore che la deve contenere deve essere 33 su un massimo di 160. Piuttosto complicato, non è vero?

* *

Vediamo ora, come fare in modo che sia l'utente a inserire una stringa:

```
! Esempio sull'utilizzo dei vettori con le stringhe – versione 3

Include "Parser";
Include "VerbLib";
Include "Replace";

Constant LUNGH_MAX 18;

Array nome_player->LUNGH_MAX;
Array nome_computer->LUNGH_MAX;

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo

  print "^Come ti chiami? ";
  ReadArray(nome_player, LUNGH_MAX);
  PrintToBuffer(nome_computer, LUNGH_MAX, "massimo");
  print "Ciao ", (PrintStringArray) nome_player, ", ";
  if((CmpStr(nome_player, nome_computer))==1) print "abbiamo lo stesso
     nome.^^";
  else print "io sono ", (PrintStringArray) nome_computer, ".^^";

  print "Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

L'istruzione ReadArray da me scritta, definita nel file Io.h (che contiene a sua volta anche le istruzioni CmpStr e PrintStringArray), è quella che permette l'inserimento in input di una stringa da parte dell'utente[6]. La riga ReadArray(nome_player, LUNGH_MAX); chiama quindi questa istruzione passandole, come parametri, il nome dell'array che deve contenere la stringa dell'utente e la sua lunghezza massima.
L'istruzione CmpStr confronta due stringhe e restituisce il valore 1 se sono uguali, 0 se invece non lo sono. La riga CmpStr(nome_player, nome_computer); chiama quindi questa istruzione passandole, come parametri, i nomi degli array contenenti le due stringhe da confrontare. Occorre però notare ancora una cosa: perché al vettore nome_computer viene passata la stringa "massimo" anziché "Massimo"? Dovete sapere che l'istruzione ReadArray (o meglio, l'istruzione KeyboardPrimitive usata da ReadArray), trasforma automaticamente tutte le lettere maiuscole in minuscole. Se l'utente digita quindi "Mario", il vettore nome_player conterrà "mario" anziché "Mario". Di per sé il problema non sarebbe tanto grave, se non fosse per il fatto che due righe più sotto il nome dell'utente viene stampato a video insieme a quello del computer.
Ecco allora come risolvere il problema:

[6] In Inform esiste anche un modo più semplice per chiedere in input una stringa da parte dell'utente, come spiegato alla fine del paragrafo 2.16.

```
! Esempio sull'utilizzo dei vettori con le stringhe - versione 4

Include "Parser";
Include "VerbLib";
Include "Replace";

Constant LUNGH_MAX 18;

Array nome_player->LUNGH_MAX;
Array nome_computer->LUNGH_MAX;

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo

  print "^Come ti chiami? ";
  ReadArray(nome_player, LUNGH_MAX);
  nome_player->2 = UpperCase(nome_player->2);
  PrintToBuffer(nome_computer, LUNGH_MAX, "Massimo");
  print "Ciao ", (PrintStringArray) nome_player, ", ";
  if((CmpStr(nome_player, nome_computer))==1) print "abbiamo lo stesso
     nome.^^";
  else print "io sono ", (PrintStringArray) nome_computer, ".^^";

  print "Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

basta usare l'istruzione UpperCase passandole come parametro la lettera iniziale della stringa digitata dall'utente (contenuta a sua volta nella terza cella del vettore nome_player ⟶ nome_player->2).

* *

Concludiamo il paragrafo con quest'ultimo programma:

```
! Esempio sull'utilizzo dei vettori con le stringhe - versione 5

Include "Parser";
Include "VerbLib";
Include "Replace";

Constant LUNGH_MAX 10;

Array nome->LUNGH_MAX;

[ Funz_princip car;
  do
  {
     ClearScreen(); ! pulisce lo schermo
```

```
     PrintToBuffer(nome, LUNGH_MAX, "Massimo");
     ChangeStringArray(nome);
     print "Ciao, ", (PrintStringArray) nome, ".^";
     print "^Vuoi continuare (s/n)? ";
     car = ReadChar();
  }
  until (car == 'n');
];

[ ChangeStringArray the_array i;
  for (i=2: i<=(the_array->1)+2: i++)
  {
     if (the_array->i == 'a' or 's') the_array->i = random('e', 'i',
         'u');
  }
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

la funzione ChangeStringArray cambia le lettere “a” e “s” della parola “Massimo”, scegliendo a caso fra le lettere “e”, “i”, “u” per mezzo dell’istruzione random. Quest’ultima, è anche in grado di generare dei numeri casuali come nel seguente esempio:

```
[Funz_princip a count;
  for (count=1: count<=10: count++)
  {
     a = random(6);
     print a, " ";
  }
];
```

a può valere un numero a caso compreso tra 1 e 6; un’efficace simulazione del lancio di un dado.
L’istruzione ReadChar (definita anch’essa nel file Io.h) legge un carattere inserito dall’utente proprio come nell’istruzione Presskey ma, a differenza di quest’ultima, deve essere seguito dalla pressione del tasto INVIO.

§2.16 Le stringhe numeriche

In Inform, una stringa può anche essere costituita da soli numeri (ad esempio “10456798”), che sono però visti da Inform come dei caratteri. Se una stringa contiene quindi 5, Inform interpreta questo dato come il carattere 5 (e non il numero 5 come invece avviene se assegniamo questo valore a una variabile).
In molte avventure testuali viene spesso richiesta l’introduzione di un codice da parte del giocatore per poter proseguire nel gioco. Dopo quanto detto, ecco quindi come fare in Inform una cosa del genere:

```
! Esempio sull'utilizzo dei vettori con le stringhe - versione 6

Include "Parser";
Include "VerbLib";
Include "Replace";

Constant LUNGH_MAX 15;

Array codice_gioco->LUNGH_MAX;
Array codice_player->LUNGH_MAX;

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  print "^Inserisci il codice: "; ReadArray(codice_player, LUNGH_MAX);
  PrintToBuffer(codice_gioco, LUNGH_MAX, "3123456789");
  print "Il codice da te inserito ";
  if((CmpStr(codice_player, codice_gioco)) == 1) print "@`e corretto e la
     porta si apre.^^";
  else print  "@`e errato. Quello esatto @`e: ", (PrintStringArray)
              codice_gioco, "^^";
  print "Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

Ricordatevi anche che due o più stringhe numeriche non possono essere sommate fra di loro: capite ora la differenza che intercorre tra un numero contenuto in una variabile e un numero contenuto in una stringa?

Dal momento che l’inserimento di una stringa più lunga del vettore che la deve contenere può generare errori piuttosto gravi, Nelson ha pensato bene di definire, di default, due vettori di libreria chiamati buffer e parse (usati, tra l’altro, da Inform stesso per i comandi inseriti dal giocatore). Se vengono utilizzati per l’inserimento in input di una stringa, non ci si deve più preoccupare della larghezza massima del vettore che la deve contenere (a meno che l’utente non vada a inserire, come già detto nel paragrafo precedente, una stringa più lunga di 160 caratteri – cosa alquanto improbabile). Ecco allora come si potrebbe effettuare una prima semplificazione dell’esempio visto poco fa:

```
Include "Parser";
Include "VerbLib";
```

```
Include "Replace";

Array codice_gioco buffer "3123456789";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  print "^Inserisci il codice: "; KeyboardPrimitive(buffer, parse);
  print "Il codice da te inserito ";
  if((CmpStr(buffer, codice_gioco)) == 1) print "@`e corretto e la
     porta si apre.^^";
  .
  .
  .
```

Stando così le cose, non è più necessario usare l'istruzione ReadArray (che rimane comunque sempre utile nel caso in cui si debbano utilizzare dei vettori diversi da buffer e parse). Volendo poi semplificare ulteriormente l'esempio:

```
Include "Parser";
Include "VerbLib";
Include "Replace";

[ Funz_princip;
  ClearScreen(); ! pulisce lo schermo
  print "^Inserisci il codice: "; KeyboardPrimitive(buffer, parse);
  print "Il codice da te inserito ";
  if (parse-->1 == '3123456789') print "@`e corretto e la porta si
     apre.^^";
  else print  "@`e errato. Quello esatto @`e: 3123456789^";
  print "Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "ItalianG";
```

possiamo anche fare a meno di utilizzare l'istruzione CmpStr (che rimane però sempre utile nel caso in cui si debbano confrontare i contenuti di due o più vettori tra di loro).

§2.17 Le direttive

Come in tutti i linguaggi di programmazione che si rispettino, anche in Inform sono presenti le direttive. Esse sono sostanzialmente delle istruzioni, inizianti tutte con il carattere #, che vanno a influire sulla compilazione. Se apriamo il file Io.h e osserviamo il contenuto dell'istruzione PrintStringArray:

```
[ PrintStringArray the_array i;
  #IfDef TARGET_ZCODE;
  for (i=2: i<(the_array->1)+2: i++) print (char) the_array->i;
  #IfNot;
  for (i=4: i<=(the_array->3)+3: i++) print (char) the_array->i;
  #EndIf;
];
```

possiamo già da subito individuarne alcune:

- la riga #IfDef TARGET_ZCODE; dice a Inform di compilare il codice che segue (la riga for (i=2...)) solo se ci si trova "nell'ambiente" Z-CODE (se si sta, cioè, utilizzando il compilatore di Inform). Se così non è (l'istruzione #IfNot) compila il restante codice (la riga for (i=4...)) nell'ambiente Glulx[7] (testabile, volendo, come #IfDef TARGET_GLULX;);
- l'istruzione #EndIf è sempre associata a ogni istruzione #IfDef perché chiude il ciclo if aperto da quest'ultima;

Le altre direttive che Inform mette a disposizione sono #IfnDef <nome> (che compila il codice che segue se nome non è stato definito), #Iftrue <condizione> (che compila il codice che segue solo se la condizione a cui fa riferimento è vera) e #Iffalse <condizione> (come la precedente solo che la condizione a cui fa riferimento deve essere falsa).

▲ La funzione ChangeStringArray, vista alla fine del paragrafo 2.15, funziona così com'è solo sotto Inform. Ecco allora, sulla base di quanto abbiamo appena visto, come fare in modo che funzioni anche sotto Glulx:

```
[ ChangeStringArray the_array i;
  #IfDef TARGET_ZCODE;
  for (i=2: i<(the_array->1)+2: i++)
  {
     if (the_array->i == 'a' or 's') the_array->i = random('e', 'i',
         'u');
  }
  #IfNot;
  for (i=4: i<=(the_array->3)+3: i++)
  {
     if (the_array->i == 'a' or 's') the_array->i = random('e', 'i',
         'u');
  }
  #EndIf;
];
```

Naturalmente l'istruzione #IfNot non è obbligatoria. Se volessimo per assurdo fare in modo che questa funzione venisse compilata solo in ambiente Inform, basterebbe attuare la seguente modifica:

```
[ ChangeStringArray the_array i;
  #IfDef TARGET_ZCODE;
  for (i=2: i<(the_array->1)+2: i++)
  {
     if (the_array->i == 'a' or 's') the_array->i = random('e', 'i',
         'u');
  }
  #EndIf;
];
```

e la funzione così definita verrebbe del tutto ignorata in ambiente Glulx.

[7] Glulx (pienamente supportato da Infit e da Jif) è un linguaggio di programmazione, scritto dal grande Andrew Plotkin, che risulta essere perfettamente compatibile con Inform (tanto da poter essere considerato a tutti gli effetti un'estensione di quest'ultimo) e offre la possibilità di creare delle avventure testuali grafiche e sonore eseguibili con il programma WinGlulxe. Ulteriori informazioni potete trovarle al capitolo 5.

§2.18 Il debugging

Il debugging è quel processo che permette al programmatore di individuare degli eventuali errori (chiamati più comunemente bug) durante la fase di esecuzione di un programma. Ecco un tipico esempio che chiarisce meglio quanto appena detto:

```
! Esempio sull'uso del debugging

Include "Parser";
Include "VerbLib";
Include "Replace";

Constant DEBUG_TEST;

[ Funz_princip x n;
  ClearScreen(); ! pulisce lo schermo
  print "^Inserire il numero da elevare al quadrato: ";
  x = GetNumber(2);

  #IfDef DEBUG_TEST;
     print "***x ora vale ", x, "***^";
     KeyCharPrimitive();
  #Endif;

  n = Quadrato(x); print "Il quadrato di ", x ," @`e ", n ,".^";
  print "^Premi un tasto per uscire^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

[ Quadrato number;
  return number*number;
];

[ Initialise;
     Funz_princip();
     quit; ! fine del programma
];

Include "Io";
Include "ItalianG";
```

Se la costante DEBUG_TEST è definita, allora Inform stampa a video il messaggio corrispondente:

```
Inserire il numero da elevare al quadrato: 6
***x ora vale 6***
```

In questo modo, siamo veramente sicuri che l'istruzione GetNumber abbia effettivamente ritornato alla variabile x il valore da noi voluto. Dopo la pressione di un tasto qualsiasi da parte dell'utente, il programma continua normalmente:

```
Il quadrato di 6 è 36.

Premi un tasto per uscire
```

Se il codice compreso fra le istruzioni #IfDef ed #Endif viene compilato ed eseguito solo quando è definita la costante DEBUG_TEST, a questo punto dovrebbe essere ormai facile capire come fare in modo che tutto questo non avvenga. Basta semplicemente commentare con un punto esclamativo la riga nella quale la costante viene definita:

```
! Constant DEBUG_TEST;
```

ottenendo così il seguente risultato:

```
Inserire il numero da elevare al quadrato: 6
Il quadrato di 6 è 36.

Premi un tasto per uscire
```

che la dice lunga sulle potenzialità di questo straordinario linguaggio di programmazione.

CAPITOLO 3

RUINS, L'AVVENTURA COMINCIA

Siamo negli anni '30; siete un intrepido (o forse folle?) archeologo che è andato nella foresta amazzonica alla ricerca di reperti archeologici da riportare alla civiltà, ma qualcosa è andato storto. Sono passati ormai diversi giorni da quando avete messo piede in quest'enorme "tappeto verde", e siete senz'acqua, sporchi e sudati. I rovi spinosi che siete stati costretti ad attraversare vi hanno lacerato la pelle e Londra vi sembra ormai tanto lontana…
Ma, proprio quando state per perdere ogni speranza, scorgete in lontananza un immenso altopiano. Vi mettete a correre, quasi come se il vostro sesto senso avesse individuato qualcosa, e così è: delle rovine, un cumulo di materiale da costruzione che una volta avrebbero potuto costituire una piramide sepolcrale… cosa volete di più dalla vita? Finalmente avete scoperto qualcosa che ha a che fare con l'archeologia, e chissà quanti tesori sepolti ci sono là sotto!
Vi guardate intorno: cupi ulivi crescono ovunque e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata. Tirate fuori dalla tasca il vostro "MANUALE DELL'ARCHEOLOGO ALLE PRIME ARMI" ma qualcosa non va: il titolo di questo libro è "COME SCRIVERE (E GIOCARE) DELLE AVVENTURE TESTUALI IN INFORM E GLULX".
Da dove salta fuori questa roba? Beh, la piramide non scappa e cominciate così a sfogliare le prime pagine…

§3.1 Il Grande Altopiano (dalla padella alla brace)[1]

In Inform, un'avventura testuale inizia con la funzione Initialise, proprio come accade in questo primo programma[2]:

```
Constant Story "RUINS";
Constant Headline
        "^Un esempio di lavoro interattivo.^
         Copyright (c) 1999 di Graham Nelson.^
         Traduzione e adattamenti di Vincenzo Scarpa e Raffaello Valesio
         (c) 2002-2003 su permesso dell'autore.^^";

Include "Parser";
Include "VerbLib";
Include "replace";

! ---------------------------------------------------------------------- !
!       Il Grande Altopiano.
! ---------------------------------------------------------------------- !
Object  Forest "Il Grande Altopiano"
  with  description
           "I  tuoi  appunti  menzionano  qualcosa  circa  questa  bassa
            scarpata  di  calcare,  ma  la  foresta  pluviale  ne  ha
            rivendicato il possesso. Cupi ulivi crescono ovunque, e una
            pioggia  appena  smessa  riempie  l'aria  con  una  calda  nebbia
            bagnata.^
            La ~Struttura 10~ @`e un cumulo di materiali da costruzione,
            che  una  volta  avrebbero  potuto  costituire  una  piramide
            sepolcrale,  ma  della  quale  ora  nulla  rimane,  eccetto  alcuni
            gradini  scolpiti  nella  nuda  roccia  che  portano  gi@`u,
            nell'oscurit@`a.",
```

[1] A tutti quelli di voi che sono passati dal capitolo 1 direttamente a questo (a quelli di voi, cioè, che hanno saltato il capitolo 2) consiglio vivamente di leggere con molta attenzione il paragrafo 2.1 (anche se naturalmente nessuno vi obbliga).
[2] Tutti i listati di questo capitolo si trovano nella directory Capitolo3 del file esercizi_manuale.zip..

```
        u_to
            "Gli alberi sono coperti di spine e ti ridurresti le mani a
            brandelli.",
        cant_go
            "La foresta pluviale @`e fitta, e non l'hai attraversata per
            giorni e giorni per rinunciare alla tua scoperta proprio ora.
            Hai bisogno di raccogliere un po' di manufatti da riportare
            alla civilt@`a, prima di abbandonare la spedizione.",
  has   light;


[ Initialise;
        location = Forest;
        "^^^Dopo giorni di inutili ricerche, passati senz'acqua
        attraversando i rovi della foresta, alla fine la tua pazienza
        @`e stata ricompensata: hai fatto una scoperta!^";
];

! ---------------------------------------------------------------------- !
Include "ItalianG";
```

OGNI AVVENTURA TESTUALE HA UNA LOCAZIONE DI PARTENZA (che in questo caso è l'oggetto Forest) ED È COSTITUITA DA UN CERTO NUMERO DI OGGETTI. Nel nostro caso, l'oggetto Forest ha a sua volta un nome interno (Forest) utilizzato all'interno del listato, un nome esterno (il Grande Altopiano) che viene stampato a video durante il gioco[3], la proprietà description (la descrizione dell'oggetto ⟶ non sempre presente), delle direzioni (u_to che indica sopra e cant_go che indica tutte le altre ⟶ non sempre presenti) e l'attributo light (anche questo non sempre presente).
Provando a compilare il programma ecco cosa appare:

```
Dopo giorni di inutili ricerche, passati senz'acqua attraversando i rovi della foresta, alla fine la tua pazienza è
stata ricompensata: hai fatto una scoperta!

RUINS
Un esempio di lavoro interattivo.
Copyright (c) 1999 di Graham Nelson.
Traduzione e adattamenti di Vincenzo Scarpa e Raffaello Valesio (c) 2002-2003 su permesso dell'autore.

Versione 1 -- Numero di serie 041109
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha
rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una
calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una
piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che
portano giù, nell'oscurità.

>
```

Come potete vedere, la dimostrazione effettiva che il gioco parte proprio dalla funzione Initialise, è il fatto che inizia dal Grande Altopiano e che il primo testo stampato a video è proprio quello contenuto all'interno della funzione stessa. Seguono poi le istruzioni per la stampa della scritta RUINS (definita all'inizio come Constant Story ⟶ deve essere sempre presente), del contenuto

[3] Più precisamente, tutte le volte che il giocatore si trova nella locazione a cui il nome esterno fa riferimento.

della costante Headline (anche questa, sempre presente), e i relativi file di include (sempre presenti) che sono nell'ordine:

- Parser(.h) - contiene le routine per il parser;
- Verblib(.h) - la libreria dei verbi;
- Replace(.h) - stampa a video alcune note relative a Infit;
- ItalianG(.h) - la grammatica dei verbi adattata alla nostra lingua.

I testi successivi che appaiono sono il nome esterno dell'oggetto Forest, la sua descrizione, e il segno di maggiore (>) che si chiama prompt dei comandi.
A questo punto tocca a noi. Proviamo allora a inserire un paio di comandi per vedere cosa succede:

```
>su
Gli alberi sono coperti di spine e ti ridurresti le mani a brandelli.

>est
La foresta pluviale è fitta, e non l'hai attraversata per giorni e giorni per rinunciare alla tua scoperta proprio ora. Hai bisogno di raccogliere un po' di manufatti da riportare alla civiltà, prima di abbandonare la spedizione.

>salta
Salti sul posto, senza risultato.

>aspetta
Il tempo passa.

>prega
Sembra che le tue preghiere non siano state esaudite.

>canta
Sei suonatissimo.

>dormi
Non sei per niente assonnato

>
```

In questo caso, ogni volta che si verifica l'evento u_to (quando cioè vogliamo dirigerci verso l'alto), Inform risponde che gli alberi sono coperti di spine e che quindi non possiamo farlo. Se invece proviamo ad andare in qualsiasi altra direzione (quando cioè si verifica l'evento cant_go), Inform ci dice che non possiamo abbandonare la spedizione.
Fino a qui non sembra esserci nulla di strano, perché questi sono eventi che abbiamo definito. Ma, come potete vedere, Inform "reagisce" anche a dei comandi che non sono stati descritti da nessuna parte all'interno del nostro programma. Come mai?
Perché Inform, a differenza della maggior parte dei linguaggi di programmazione, è dedicato solo ed esclusivamente alle avventure testuali e offre un ambiente integrato e pronto per essere tranquillamente utilizzato e personalizzato. Questo non significa però che l'avventura testuale è già pronta; tuttavia, tutte le routine per la gestione dell'inventario, del salvataggio e del caricamento della posizione, del parser dei comandi e molto altro ancora (che nei linguaggi di programmazione non specifici per le avventure testuali devono essere programmati da zero con delle difficoltà non indifferenti) sono state già incluse nei famosi file di Include descritti prima. Un bel vantaggio, non è vero?

L'ultima cosa che rimane da vedere è l'attributo light, che dice a Inform che Forest è un oggetto "illuminato". Per farvi capire meglio quello che sto dicendo, proviamo a cambiare questo attributo

con ~light (per chi non avesse ancora letto il capitolo 2, la tilde è il simbolo di negazione) ed ecco quello che appare:

```
Dopo giorni di inutili ricerche, passati senz'acqua attraversando i rovi della foresta, alla fine la tua pazienza è stata ricompensata: hai fatto una scoperta!

RUINS
Un esempio di lavoro interattivo.
Copyright (c) 1999 di Graham Nelson.
Traduzione e adattamenti di Vincenzo Scarpa e Raffaello Valesio (c) 2002-2003 su permesso dell'autore.

Versione 1 -- Numero di serie 041109
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

Oscurità
È completamente buio, e non riesci a vedere niente.

>
```

Come potete vedere, la descrizione della foresta è scomparsa, perché siete… al buio, proprio come se vi avessero chiuso in un ascensore con le luci guaste (e chi di voi soffre di claustrofobia adesso starà sicuramente facendo dei salti di gioia). Questo evento, che in Inform si chiama the_dark, ci sarà molto utile più avanti per simulare il funzionamento della lampada al sodio: ma forse è meglio procedere con ordine, senza mettere troppa carne al fuoco.

Inform? Oggetti? Funzioni? Vi chiedete a cosa possano servire tutte queste nozioni, e vi chiedete anche chi possa essere così pazzo da scrivere un manuale del genere.
Urla di scimmie, pipistrelli, pappagalli, macao sono i soli suoni che in questo momento vi fanno compagnia. Vi guardate intorno e vedete che una cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà. Abbandonate il manuale di Inform al suo destino e andate a esaminare la cassa che a sua volta contiene un ingombrante, robusto e tenace modello di macchina fotografica a lastre con la struttura di legno, e un giornale stropicciato di un mese fa. Li prendete entrambi e li mettete nel vostro inventario, insieme agli oggetti che già possedete: la mappa di Quintana Roo, la lampada al sodio e il dizionario Maya di Waldeck.

§3.2 La cassa d'imballaggio (riuscirete a riempirla tutta?)

Abbiamo detto che un'avventura testuale è costituita da un insieme di oggetti. Ora, quello che occorre capire, è come (e soprattutto quando) si devono collegare tra loro; ecco perché nel nostro programma sono state inserite le definizioni di alcuni nuovi oggetti:

```
Constant Story "RUINS";
Constant Headline
        "^Un esempio di lavoro interattivo.^
         Copyright (c) 1999 di Graham Nelson.^
         Traduzione e adattamenti di Vincenzo Scarpa e Raffaello Valesio
         (c) 2002-2003 su permesso dell'autore.^^";

Constant MAX_CARRIED = 7;

Include "Parser";
Include "VerbLib";
Include "replace";

! ---------------------------------------------------------------------- !
!       Il Grande Altopiano
! ---------------------------------------------------------------------- !
Object  Forest "Il Grande Altopiano"
  with  description ...
        u_to ...
        cant_go ...
        before [;
            Listen: "Urla di scimmie, pipistrelli, pappagalli,
                     macao.";
            ],
  has   light;

Object  -> packing_case "cassa d'imballaggio"
  with  name 'imballaggio' 'cassetta' 'cassa',
        initial
            "La tua cassa d'imballaggio giace qui, pronta per contenere
             eventuali scoperte di alto valore culturale da riportare
             alla civilt@`a.",
        before [;
            Take, Remove, PushDir:
                "La cassa @`e troppo pesante per preoccuparsi di
                 spostarla, almeno finch@'e la spedizione @`e ancora
                 incompleta.";
            ],
```

```
  has   female static container open openable;

Object  -> -> camera "macchina fotografica a lastre"
  with  name 'lastre' 'macchina' 'fotografica',
        description
            "@`E un ingombrante, robusto e tenace modello di macchina
             fotografica a lastre con la struttura di legno: come tutti
             gli archeologi, le sei particolarmente affezionato.",
  has   female;

Object  -> -> newspaper "giornale di un mese fa"
  with  name 'times' 'giornale' 'mese',
        description
            "Il ~Times~ del 26 febbraio 1938, che si @`e subito bagnato e
             stropicciato dopo essere stato esposto per un mese a questo
             clima, pi@`u o meno come ti senti tu ora. Forse c'@`e la
             nebbia a Londra. Forse ci sono le bombe.";

! ---------------------------------------------------------------------- !
!    Questi oggetti vengono spostati nelle locazioni a mano a mano che
!    il gioco procede.
! ---------------------------------------------------------------------- !
Object  sodium_lamp "lampada al sodio" ...

Object  dictionary "dizionario maya di Waldeck" ...

Object  map "mappa di Quintana Roo"
  with  article "la",
        name 'mappa' 'schizzo' 'abbozzo' 'quintana' 'roo',
        description
            "Questa mappa evidenzia meglio il ruscello che ti ha portato
             qui, al di l@`a del confine meridionale del Messico, nella
             pi@`u profonda foresta pluviale, interrotta solo da questo
             altopiano.",
  has   female;

! ---------------------------------------------------------------------- !
!       Utility routines.
! ---------------------------------------------------------------------- !
[ Initialise;
        location = Forest;
        move map to player;
        move sodium_lamp to player;
        move dictionary to player;
        "^^^Dopo giorni di inutili ricerche, passati senz'acqua
         attraversando i rovi della foresta, alla fine la tua pazienza
         @`e stata ricompensata: hai fatto una scoperta!^";
        ];

! ---------------------------------------------------------------------- !
Include "ItalianG";
```

Partiamo dall'oggetto packing_case (la nostra cassa d'imballaggio). La prima cosa che si può notare, è la presenza, alla prima riga, del simbolo `->`: in questo modo, diciamo ad Inform che questo oggetto è visibile nell'oggetto Forest (che cioè la cassa d'imballaggio è visibile dal giocatore

quando si trova nel Grande Altopiano). Esiste comunque un modo più semplice per mettere in relazione i due oggetti. Scrivendo infatti:

```
Object packing_case "cassa d'imballaggio" Forest
```

otteniamo lo stesso e identico risultato, ma per questa avventura si è deciso di adottare il simbolo `->` anziché il nome dell'oggetto di appartenenza.
OGNI OGGETTO INOLTRE, PUÒ ESSERE CHIAMATO CON DEI NOMI DIVERSI DURANTE IL GIOCO. È quello che accade nella riga with name… e se scrivo ad esempio: prendi la cassetta o prendi la cassa, per Inform il riferimento all'oggetto packing_case è identico per entrambi i comandi (esegue cioè la stessa azione in entrambi i casi)[4].
Il testo posto dopo la proprietà initial è quello che appare tutte le volte che il giocatore si trova nel Grande Altopiano e descrive la cassa in questione. Se si cerca invece di prenderla (Take), rimuoverla (Remove), o spingerla in una direzione (PushDir), la risposta è sempre che è troppo pesante da spostare…
Per quanto riguarda gli attributi, qui ne abbiamo in abbondanza, nell'ordine:

- female: indica che l'oggetto è femminile (se invece è maschile – in inglese male - non si mette nulla, perché è il valore di default);
- static: l'oggetto è fisso al suo posto e non può essere preso o spostato dal giocatore;
- container: l'oggetto in questione è un contenitore che, come dice la parola stessa, può contenere al suo interno altri oggetti che possono essere a loro volta presi, esaminati, usati e riposti;
- open: "Elementare Watson" come direbbe Sherlock Holmes; la cassa è aperta, ma il giocatore non può entrarvi dentro. Se questo attributo non è presente la cassa è chiusa di default;
- openable: la cassa può essere aperta o chiusa dal giocatore o da un altro personaggio del gioco (che in gergo "informiano" è un NPC[5] o, in italiano, PNG[6]).

Gli oggetti camera (la macchina fotografica) e newspaper (il giornale) presentano solo una novità: la presenza del doppio simbolo `-> ->` per dire a Inform che sono visibili all'interno della cassa d'imballaggio. Occorre notare che, in questo caso, l'ordine di definizione degli oggetti è molto importante: in poche parole, l'oggetto packing_case deve essere messo subito dopo l'oggetto Forest, mentre gli oggetti newspaper e camera devono essere messi subito dopo l'oggetto packing_case.

La mappa si trova invece nell'inventario del giocatore, e questo è possibile grazie all'istruzione move che nella funzione Initialise la sposta insieme al dizionario e alla lampada al sodio che verranno descritti in dettaglio più avanti.
INFORM PREVEDE ANCHE LA POSSIBILITÀ DI ASSOCIARE UNO SPECIFICO ARTICOLO A OGNI OGGETTO. Di default, se è maschile gli viene attribuito l'articolo indeterminativo un, se è femminile una. Nel caso della mappa però, essendo proprio la mappa di Quintana Roo (nel senso che ne esiste una sola), è meglio attribuirle l'articolo la. Ora, se durante il gioco proviamo a fare una lista dell'inventario:

```
>i
Stai portando:
  il dizionario maya di Waldeck
  una lampada al sodio
  la mappa di Quintana Roo
```

Inform si "accorge" del nuovo articolo e lo visualizza insieme al nome dell'oggetto cui appartiene.

[4] Vi consiglio di abbondare sempre con i nomi alternativi, perché in questo modo non si limita il campo di azione del giocatore (si evita che l'oggetto possa essere chiamato in un modo soltanto, rendendo così le azioni meno frustranti).
[5] Acronimo inglese di Non Player Characters.
[6] Acronimo di Personaggi Non Giocatori.

Le ultime due cose da notare in questo paragrafo sono:

- l'aggiunta del verbo Listen (in italiano Ascolta) all'oggetto Forest. In questo modo, se durante il gioco il giocatore si trova nel Grande Altopiano e dà il comando ASCOLTA:

  ```
  >ascolta
  Urla di scimmie, pipistrelli, pappagalli, macao.
  ```

  viene visualizzato il messaggio da noi voluto. Notate che di default questo comando dà il seguente messaggio:

  ```
  >ascolta
  Non si sente niente di particolare.
  ```

  e questa è la dimostrazione pratica di come Inform offra un ambiente integrato ma anche altamente personalizzabile. Attenzione però: il messaggio da noi voluto viene visualizzato SOLO nell'oggetto Forest, mentre per tutte le altre locazioni il messaggio sarà sempre quello di default (a meno che non si prevederanno nuovi messaggi in altre locazioni).
- l'inserimento della costante MAX_CARRIED, il cui valore indica il numero massimo di oggetti trasportabili dal giocatore (100 di default). Quindi, quando il nostro inventario è pieno (se possediamo cioè il numero massimo di oggetti → nel nostro caso 7), non è più possibile prenderne altri, a meno che non se ne posi qualcuno (per es. posa il giornale).

Vorreste esplorare la piramide, ma l'entrata è bloccata dalle macerie e non avete la più pallida idea di come toglierle. Stanchi e affamati, vi sedete per terra e notate che poco distante c'è un fungo macchiato con il gambo sottile. Vorreste provare a mangiarlo, ma non siete del tutto sicuri che non sia velenoso. Se poi state male, chi vi viene a soccorrere? Il telefonino non è stato ancora inventato, e sapete bene che ci vorranno ancora moltissimi anni prima che qualcuno lo costruirà.
Al diavolo, quando si ha fame si ha fame. Raccogliete delicatamente il fungo e ne assaggiate un pezzetto. Il sapore è orribile, ma almeno non siete ancora morti, è già qualcosa. Improvvisamente, un macao passa sopra la vostra testa… il battito delle sue ali è assordante… delle pietre crollano l'una sull'altra…
Le macerie sono sparite e l'entrata si è liberata! Ora potete vedere dei gradini rotti e logori che conducono verso una sala poco illuminata: potreste essere le prime persone a mettervi piede dopo cinquecento anni…

§3.3 Il fungo (velenoso o non velenoso?)

La definizione dell'oggetto mushroom (il nostro fungo, per l'appunto), presenta alcune novità rispetto a quanto visto finora:

```
Object  -> mushroom "fungo macchiato"
  with  name 'macchiato' 'macchiettato' 'fungo' 'velenoso',
        initial
            "Un fungo macchiato cresce dalla terra fradicia, su un gambo
             lungo.",
        description
            "Il fungo @`e ricoperto da delle macchie e non sei del tutto
             sicuro che non sia velenoso.",
        after [;
            Take:
                if (self.mushroom_picked)
                    "Hai raccolto il fungo mezzo marcio.";
                self.mushroom_picked = true;
                "Hai raccolto abilmente il fungo, senza staccarlo dal suo
                 gambo sottile.";
            Drop:
                "Il fungo cade sul terreno e comincia a decomporsi.";
            Eat:
                steps.rubble_filled = false;
                "Lo sgranocchi ad un angolo, incapace di capire l'origine
                 di un gusto cos@`i acre, distratto dal volo di un macao
                 sopra la tua testa che sembra quasi un'esplosione nel
                 sole. Il battito delle sue ali @`e quasi assordante, e
                 delle pietre crollano una sull'altra.";
            ],
        mushroom_picked false,
  has   edible;
```

Allora… qui, il messaggio della proprietà description, appare solo quando si dà il comando esamina:

>esamina il fungo
Il fungo è ricoperto da delle macchie e non sei del tutto sicuro che non sia velenoso.

mentre il messaggio della proprietà initial, come già sappiamo, appare insieme alla descrizione del Grande Altopiano. Un'altra cosa molto importante in Inform, è la gestione delle azioni (o dei verbi). Nel nostro caso, sono stati previsti dei controlli per i verbi prendi (Take), posa (Drop) e mangia (Eat). Alcune azioni sono possibili solo se l'oggetto che le contiene ha dei determinati attributi: il giocatore può mangiare il fungo perché l'oggetto mushroom contiene l'attributo edible.

La variabile locale mushroom_picked viene dichiarata con il valore false (falso)[7] e viene usata da Inform per vedere se il giocatore ha raccolto il fungo per la prima volta oppure no. Non appena questi lo raccoglie, Inform si chiede se il valore di mushroom_picked è uguale a true (vero)[8]. Se no, glielo assegna (il fungo è stato raccolto per la prima volta) e stampa a video il messaggio che segue. Se il giocatore lo posa poi a terra e lo raccoglie una seconda volta, la variabile vale true e viene quindi stampato l'altro messaggio. Ecco in pratica cosa accade:

>prendi il fungo
Hai raccolto abilmente il fungo, senza staccarlo dal suo gambo sottile.

>posa il fungo
Il fungo cade sul terreno e comincia a decomporsi.

>prendi il fungo
Hai raccolto il fungo mezzo marcio.

Il nome della variabile è preceduto da self. per dire a Inform che mushroom_picked appartiene solo e soltanto all'oggetto mushroom. La stessa cosa vale per la variabile locale rubble_filled che però appartiene all'oggetto steps e serve per verificare se l'ingresso della piramide è bloccato oppure no. Questo oggetto dichiara il valore della variabile come true:

```
Object  -> steps "gradini scolpiti nella roccia"
  with  article "dei",
        name 'gradino' 'gradini' 'roccia' 'scale' 'scolpiti' 'piramide'
             'funebre' 'struttura' 'dieci' '10',
        description [;
            if (self.rubble_filled)
                "Le macerie ostruiscono il passaggio dopo appena pochi
                 passi.";
            print "Dei gradini rotti e logori conducono verso una sala
                   poco illuminata. Potresti essere ";
            if (Square_Chamber hasnt visited)
                print "la prima persona a mettervi piede ";
            else
                print "stata la prima persona ad avervi messo piede ";
            "dopo cinquecento anni. Sul primo gradino @`e inscritto il
             simbolo Q1.";
            ],
        door_to [;
            if (self.rubble_filled)
                "Le macerie ostruiscono il passaggio dopo appena pochi
                 passi.";
            return Square_Chamber;
            ],
        door_dir d_to,
        rubble_filled true,
  has   scenery pluralname door open;
```

[7] mushroom_picked false è l'equivalente di mushroom_picked = false.
[8] if (self.mushroom_picked) è l'equivalente di if (self.mushroom_picked == true).

facendo così in modo che all'inizio l'entrata resti bloccata dalle macerie. Quando invece il giocatore mangia il fungo, la variabile vale false e le macerie spariscono:

>prendi il fungo
Hai raccolto abilmente il fungo, senza staccarlo dal suo gambo sottile.

>mangia il fungo
Lo sgranocchi ad un angolo, incapace di capire l'origine di un gusto così acre, distratto dal volo di un macao sopra la tua testa che sembra quasi un'esplosione nel sole. Il battito delle sue ali è quasi assordante, e delle pietre crollano una sull'altra.

>esamina la struttura
Dei gradini rotti e logori conducono verso una sala poco illuminata. Potresti essere la prima persona a mettervi piede dopo cinquecento anni. Sul primo gradino è inscritto il simbolo Q1.

Tutte le volte poi che il giocatore visita una locazione, Inform le attribuisce il valore visited. Se proviamo a scendere i gradini e subito dopo a risalirli, accade una cosa interessante:

>giù

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

>su

Il Grande Altopiano

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>esamina la struttura
Dei gradini rotti e logori conducono verso una sala poco illuminata. Potresti essere stata la prima persona ad avervi messo piede dopo cinquecento anni. Sul primo gradino è inscritto il simbolo Q1.

L'oggetto Square_Chamber (la Sala Quadrata) è (almeno per ora) piuttosto semplice:

```
Object  Square_Chamber "La Sala Quadrata"
  with  name 'sala' 'quadrata' 'architrave' 'architravi' 'est' 'sud'
             'entrata',
        description
            "Sei in una sala di pietra oscura e profonda, larga circa
             dieci metri. Un raggio di sole, proveniente dalla cima della
             scalinata, la illumina diffusamente, ma le ombre del livello
             pi@`u basso rivelano dei passaggi verso est e sud, che
             conducono verso la pi@`u profonda oscurit@`a del Tempio.",
        u_to Forest,
  has   female light;
```

non appena il giocatore vi mette piede, la sala è visited (visitata). Quando ritorna poi in superficie e riesamina la struttura, i giochi sono fatti: essendo la Sala Quadrata stata visitata, viene stampato il secondo messaggio ("stata la prima persona ad avervi messo piede") anziché il primo ("la prima persona a mettervi piede")[9].

[9] Ricordatevi che has = ha e hasnt = non ha.

🔴 Un'altra cosa da notare è che quando si ritorna in una stanza visitata, il testo della sua descrizione scompare: perché?
Perché, di default, Inform è in modalità normale, e dà descrizioni lunghe per i luoghi mai visitati prima e brevi se già visitati. Esistono altre due modalità che sono la breve (dà descrizioni brevi per tutti i luoghi, anche se mai visitati) e la completa (dà descrizioni lunghe per tutti i luoghi, anche se già visitati). A livello di codice, queste tre modalità possono essere impostate usando la variabile di libreria lookmode[10] nella funzione Initialise, come segue:

```
[ Initialise;
        location = Forest;
        move map to player;
        move sodium_lamp to player;
        move dictionary to player;
        lookmode = 1;
        "^^^Dopo giorni di inutili ricerche, passati senz'acqua
            attraversando i rovi della foresta, alla fine la tua pazienza
            @`e stata ricompensata: hai fatto una scoperta!^";
        ];
```

Se lookmode vale 1, come nel nostro esempio, è impostata la modalità normale; se vale 2, è impostata la modalità completa e se invece vale 3 è impostata la modalità breve. A ogni modo, non dovete preoccuparvi se la descrizione della stanza visitata scompare: basta dare il comando guarda (o g in forma abbreviata) e tutto torna come prima (tranne però che nella modalità breve, dove le descrizioni non compaiono per tutta la durata del gioco).
Inform mette a disposizione delle istruzioni specifiche anche per collegare tra di loro gli oggetti in termini di direzioni. Come tutti ben sappiamo, in un'avventura testuale il giocatore può muoversi di locazione in locazione attraverso l'uso dei punti cardinali:

ISTRUZIONE	COMANDO
n_to	"nord" o "n" o "vai a nord/n"
ne_to	"nordest" o "ne" o "vai a nordest/ne"
nw_to	"nordovest" o "no" o "vai a nordovest/no"
s_to	"sud" o "s" o "vai a sud/s"
se_to	"sudest" o "se" o "vai a sudest/se"
sw_to	"sudovest" o "so" o "vai a sudovest/so"
e_to	"est" o "e" o "vai a est/e"
w_to	"ovest" o "o" "vai a ovest/o"
u_to	"su" o "sopra" o "vai su/sopra"
d_to	"giù" o "sotto" o "vai giù/sotto"
in_to	"entra" o "vai dentro"
out_to	"esci" o "vai fuori"
cant_go	tutte le direzioni che non possono essere prese

ma è bene tenere presente che è anche possibile spostarsi da una locazione all'altra senza che esse siano necessariamente collegate tra di loro (avete presente il teletrasporto?). Questo è possibile grazie all'istruzione PlayerTo, che verrà spiegata nel paragrafo 3.13.

Infine, focalizzando nuovamente la nostra attenzione sull'oggetto steps, diamo un'occhiata a tre nuovi attributi:

- scenery: in questo caso, viene usato per impedire a Inform di visualizzare il messaggio "Puoi anche vedere dei gradini scolpiti nella roccia qui." quando il giocatore si trova nel Grande Altopiano. L'oggetto in questione, inoltre, non può essere preso, spinto, tirato o girato;

[10] Ricordatevi che tutte le variabili e le funzioni di libreria non hanno bisogno di essere dichiarate per poter essere utilizzate.

- pluralname: indica che il nome dell'oggetto a cui si riferisce è plurale;
- door: questo attributo viene usato quando un oggetto deve fare da tramite tra due diverse locazioni. I gradini infatti, sono collegati da una parte al Grande Altopiano, e dall'altra alla Sala Quadrata: in questo modo, se il giocatore si trova nel Grande Altopiano può percorrerli verso il basso per esplorare i meandri più oscuri della piramide e poter così continuare la sua avventura. Occorre però fare molta attenzione a due proprietà fondamentali: door_dir che indica la direzione verso la quale si incontra la porta (in questo caso giù), e door_to che indica la stanza che si trova dietro la porta (in questo caso la Sala Quadrata). Ovviamente, la porta è aperta per la presenza dell'attributo open.

Siete in una sala di pietra oscura e profonda, pervasa dalla foschia e larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che non sapete dove conducono. Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto. Provate a dirigerti verso sud, e vi ritrovate al buio. Vi sentite quasi soffocare, e come se non bastasse, da qualche parte dei piccoli artigli si stanno muovendo velocemente.
Il vostro respiro diventa sempre più affannoso, e del sudore freddo vi cola dalla fronte: la mamma vi ha sempre detto di evitare certi posti, ma non le avete mai dato retta, e se ora vi ritrovate in questa pessima situazione la colpa è soltanto vostra.
Sentite un ticchettio ai vostri piedi e date un calcio contro qualcosa di cheratinoso che vola via. Il suono si è trasformato in un minaccioso stridio e avete la spiacevole sensazione che questi maledetti insetti vogliano divorarvi: non potete far altro che tornare indietro, salvandovi appena in tempo dai loro pericolosissimi artigli. Questa volta ve la siete vista davvero brutta…

§3.4 Chi ha paura del buio (gli insetti no di certo)?

Nel paragrafo 3.2 abbiamo visto come mettere in relazione due oggetti fra di loro. Ma se voglio mettere in relazione un oggetto con più oggetti (se voglio cioè fare in modo che un oggetto sia visibile in due o più di locazioni) come posso fare?
Diamo un'occhiata alla definizione dell'oggetto low_mist:

```
Object  low_mist "nebbia bassa"
  with  article "della",
        name 'bassa' 'foschia' 'nebbia',
        description "La nebbia ha un aroma che ricorda la tortilla.",
        before [;
            Examine, Search:
                ;
            Smell:
                <<Examine self>>;
            default:
                "La nebbia non @`e abbastanza consistente.";
            ],
        react_before [;
            Smell: if (noun == nothing) <<Smell self>>;
            ],
        found_in Square_Chamber Forest,
  has   female scenery;
```

Alla penultima riga, l'istruzione found_in fa in modo che la nebbia sia presente nella Sala Quadrata e nel Grande Altopiano; ne consegue quindi, che tutti gli eventi descritti al suo interno si verificano in entrambe le locazioni.

🔴 Altra cosa da notare, è che LA MAGGIOR PARTE DELLE AZIONI IN INFORM, SONO INCLUSE IN DUE PROPRIETÀ: after E before. Nell'appendice A è presente un elenco di tutte le azioni principali presenti in Inform (o meglio, di tutte le azioni principali definite in Infit), che sono a loro volta suddivise in tre gruppi:

- GRUPPO 1 rappresentato dai cosiddetti "comandi di sistema" (come ad esempio ESCI, VERIFICA, SALVA, ecc.) che non vengono associati a nessuna proprietà:

  ```
  if (yesorno()) <RESTORE>;
  ```

  in questo esempio, se il giocatore digita S, viene caricata la situazione del gioco salvata precedentemente;
- GRUPPO 2 a cui appartengono tutte le azioni controllate dalla proprietà after;

- GRUPPO 3 a cui appartengono, invece, tutte le azioni controllate dalla proprietà before.

Cerchiamo ora di capire meglio quanto detto. Nell'oggetto Forest, per esempio, abbiamo definito il verbo ASCOLTA nel seguente modo:

```
before [;
    Listen: "Urla di scimmie, pipistrelli, pappagalli, macao.";
    ],
```

e tutto sembra quadrare con quanto fin qui detto. Infatti, andando a vedere la tabella dell'appendice A, vediamo che questo verbo corrisponde in inglese all'azione Listen e che appartiene al gruppo 3.
Questo discorso però, è valido fino a un certo punto: nell'oggetto sodium_lamp, che sarà descritto in dettaglio nel prossimo paragrafo, il verbo ACCENDI, appartenente al gruppo 2, è presente nella proprietà before. Perché?
Il motivo è da ricercare nel funzionamento di queste due proprietà. Francesco Cordella[11] cerca di farcelo capire proponendo questo semplice esempio:

```
Object -> Cassa "cassa"
    with name 'cassa',
    initial "Per terra c'e' un'enorme cassa."
    after [;
        Open: "Pensavi che fosse pi@`u difficile aprirla ma ci riesci in
               un batter d'occhio.";
        ],
has static female container openable;
```

Abbiamo quindi una cassa chiusa ma apribile. Se il giocatore digita:

>apri la cassa

Inform, a questo punto, "apre" la cassa, ovvero dà a quest'ultima il valore open. Poi, prima di stampare il messaggio standard ("Hai aperto la cassa"), consulta after e si chiede se c'è qualcos'altro che il programmatore gli ordina di fare prima di stamparlo. Se after non c'è, Inform prosegue e stampa il messaggio standard. Ma se c'è, come nel nostro caso, stampa quello che c'è da stampare:

>apri la cassa
Pensavi che fosse più difficile aprirla ma ci riesci in un batter d'occhio.

>esamina la cassa
La cassa è vuota.

Se invece sostituiamo la proprietà after con before, ecco cosa accade:

>apri la cassa
Pensavi che fosse più difficile aprirla ma ci riesci in un batter d'occhio.

>esamina la cassa
Non puoi vedere dentro, perché la cassa è chiusa.

apparentemente tutto sembra andare per il verso giusto, ma non è così: esaminando la cassa infatti, scopriamo che è ancora chiusa. Usando before Inform non apre la cassa, dobbiamo dirglielo noi:

```
before [;
    Open: give self open; ! give (dai) self (alla cassa) open
          "Pensavi che fosse pi@`u difficile aprirla ma ci riesci in
           un batter d'occhio.";
    ],
```

[11] Autore di "Flamel" e "Kazan", Francesco Cordella è anche il curatore del sito http://www.avventuretestuali.com/.

Se tutto questo vi ha confuso un po' le idee, non preoccupatevi più di tanto e seguite questa semplice regola: QUANDO DEFINITE UN OGGETTO E STABILITE UNA CERTA AZIONE, PROVATE A USARE PER PRIMA LA PROPRIETÀ RELATIVA AL GRUPPO DI APPARTENENZA DI QUELLA AZIONE: SE FA QUELLO CHE VOLETE, BENE, ALTRIMENTI PROVATE CON L'ALTRA PROPRIETÀ. Tenete inoltre presente che, all'interno della definizione di un oggetto, una stessa azione può essere usata in entrambe le proprietà (può cioè essere presente allo stesso tempo sia in after che in before a seconda di quello che deve fare).
Se volete, potete provare come esercizio a invertire le proprietà delle azioni di tutti gli oggetti visti finora. Ne vedrete delle belle…

Riportiamo ora la nostra attenzione sull'oggetto low_mist. In Inform, oltre alla after e alla before, esiste un'altra proprietà, chiamata react_before, sulla quale occorre fare alcune considerazioni:

1. agisce solo sugli oggetti in_scope (quelli cioè visibili dal giocatore);
2. il suo uso più comune è quello di evitare le azioni senza oggetto. Nel nostro caso, infatti, se noun = nothing (se scrivo cioè ANNUSA al posto di ANNUSA L'ARIA, dove aria è il noun), Inform esegue il codice previsto per l'azione smell definita nella proprietà before (ovvero di eseguire il comando ESAMINA LA NEBBIA che, come già sappiamo, stampa a video il messaggio appartenente alla proprietà description dell'oggetto stesso ⟶ self);
3. alcuni programmatori, come ad esempio Tommaso Caldarola[12], la usano sull'azione Examine degli oggetti di tipo switchable (di oggetti cioè, che si possono attivare o disattivare, come vedremo meglio nel prossimo paragrafo), evitando così i fastidiosissimi messaggi di libreria del tipo "in questo momento è attivato/disattivato".

Per qualsiasi altra azione eseguita sulla nebbia (ad esempio PRENDI LA NEBBIA) viene visualizzato il messaggio di default della proprietà before ("La nebbia non è abbastanza consistente").

* *

Ora possiamo finalmente occuparci del buio, il tema principale di questo paragrafo. L'oggetto Corridor:

```
Object  Corridor "Corridoio in pendenza"
  with  description
            "Un corridoio basso e squadrato va da nord verso sud,
             inclinandosi verso la fine.",
        n_to Square_Chamber;
```

non ha l'attributo light. Come abbiamo già visto nel paragrafo 3.1, Inform attiva in questo caso la modalità the_dark. La prima cosa da notare è la possibilità personalizzare il messaggio di default, cosa che avviene nella funzione Initialise:

```
[ Initialise;
        .
        .
        .
        thedark.description =
            "L'oscurit@`a intorno a te @`e opprimente e ti senti quasi
             soffocare.";
        .
        .
        .
        ];
```

[12] Tommaso Caldarola è l'autore della bellissima Pecos Town, nonché uno dei programmatori più esperti di questo linguaggio di programmazione. Ulteriori informazioni potete trovarle all'indirizzo http://www.caldarola.net/.

the_dark, a sua volta, viene visto da Inform come un oggetto vero e proprio, al quale si possono relazionare altri oggetti come nel caso del tiny_claws:

```
Object  tiny_claws "suono di piccoli artigli" thedark
  with  name 'piccoli' 'artigli' 'suono' 'creature' 'mostri' 'insetti',
        initial
            "Da qualche parte, dei piccoli artigli si stanno muovendo
             velocemente.",
        before [;
            Listen:
                "Come sembrano intelligenti per essere solo degli
                 insetti.";
            Touch, Taste:
                "Non dovresti volerlo. Davvero.";
            Smell:
                "Puoi solo sentire la tua paura.";
            Attack:
                "Le creature schivano facilmente i tuoi colpi.";
            default:
                "Le creature ti sfuggono squittendo.";
            ],
        each_turn [; StartDaemon(self); ],
        daemon [;
            if (location ~= thedark) {
                self.turns_active = 0;
                StopDaemon(self);
                rtrue;
                }
            switch (++(self.turns_active)) {
                1:  "^Senti che le creature si stanno avvicinando: il tuo
                     respiro diventa sempre pi@`u affannoso.";
                2:  "^Del sudore freddo ti cola dalla fronte: le creature
                     sono quasi qui!";
                3:  "^Senti un ticchettio ai tuoi piedi e dai un calcio
                     contro qualcosa di cheratinoso che vola via. Il loro
                     suono si @`e trasformato in un minaccioso
                     stridio.";
                4:  deadflag = true;
                    "^Senti improvvisamente un dolore al polpaccio,
                     simile a quello di una puntura ipodermica. Quasi
                     subito i tuoi polmoni si paralizzano, le spalle e le
                     ginocchia si bloccano, la lingua si gonfia...";
                }
            ],
        turns_active 0;
```

La novità principale della definizione di questo nuovo oggetto è l'utilizzo di un Daemon.

IN INFORM, OGNI MOSSA CHE VIENE FATTA DAL GIOCATORE EQUIVALE A UN TURNO. Cercando di semplificare il più possibile, possiamo affermare che un Daemon è un particolare oggetto che ci permette di correlare le azioni con lo scorrere del tempo (con l'avanzare cioè, delle numero dei turni). Proviamo allora a vedere in azione l'oggetto tiny_claws:

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>s

Oscurità
L'oscurità intorno a te è opprimente e ti senti quasi soffocare.

Da qualche parte, dei piccoli artigli si stanno muovendo velocemente.

>aspetta
Il tempo passa.

Senti che le creature si stanno avvicinando: il tuo respiro diventa sempre più affannoso.

>esamina le creature
Le creature ti sfuggono squittendo.

Del sudore freddo ti cola dalla fronte: le creature sono quasi qui!

>uccidi le creature
Le creature schivano facilmente i tuoi colpi.

Senti un ticchettio ai tuoi piedi e dai un calcio contro qualcosa di cheratinoso che vola via. Il loro suono si è trasformato in un minaccioso stridio.

>prendi le creature
Le creature ti sfuggono squittendo.

Senti improvvisamente un dolore al polpaccio, simile a quello di una puntura ipodermica. Quasi subito i tuoi polmoni si paralizzano, le spalle e le ginocchia si bloccano, la lingua si gonfia...

*** Sei morto ***

In questa partita hai totalizzato 0 punti su 0 possibili, in 10 turni.

Vuoi RICOMINCIARE, CARICARE una partita salvata o USCIRE ?
>

L'istruzione each_turn esegue il suo contenuto a ogni mossa del giocatore, condizione ideale per il Daemon. L'istruzione StartDaemon(self) lo attiva e lo fa partire; subito dopo, segue la definizione vera e propria (il corpo del Daemon), che deve trovarsi all'interno della funzione daemon. Qui, la variabile locale turns_active viene incrementata di 1 ogni volta che il giocatore fa una mossa e, attraverso un ciclo switch_case, ne viene testato il contenuto:

- alla prima mossa, turns_active vale 1 e viene stampato il messaggio corrispondente;
- alla seconda mossa, turns_active vale 2 e viene stampato il messaggio corrispondente;
- alla terza mossa, turns_active vale 3 e viene stampato il messaggio corrispondente;
- alla quarta mossa turns_active vale 4 (viene quindi stampato il messaggio corrispondente), la variabile di libreria deadflag vale true (o meglio, vale 1) e il giocatore viene così "ucciso" facendo terminare il daemon insieme al gioco.

Notate che, se la locazione non è the_dark, il daemon termina (StopDaemon(self);), turns_active vale zero, e viene infine restituito il valore true che fa fermare Inform (nel senso che Inform non esegue più le istruzioni della funzione daemon). In questo modo, diamo la possibilità al povero archeologo di non essere sbranato dagli insetti famelici:

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>s

Oscurità
L'oscurità intorno a te è opprimente e ti senti quasi soffocare.

Da qualche parte, dei piccoli artigli si stanno muovendo velocemente.

>aspetta
Il tempo passa.

Senti che le creature si stanno avvicinando: il tuo respiro diventa sempre più affannoso.

>n

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>

andando verso nord, si ritorna in una locazione che ha l'attributo light e il daemon, come abbiamo già detto, termina.

E ora come potete procedere? Di affrontare nuovamente gli insetti non se ne parla… avete bisogno di luce. Vi vengono in mente un paio di trovate geniali, come cospargere di benzina l'intera piramide per poterle poi dare fuoco, oppure di abbattere un paio di alberi (a mani nude) per avere della legna da ardere. Ma all'improvviso vi ricordate di avere nel vostro inventario una robusta lampada al sodio da archeologo che "forse" fa proprio al caso vostro. La posate per terra, la accendete e la spingete verso sud. Dopo avere sudato sette camicie e nove fazzoletti, vi ritrovate nello stesso punto di poco fa, quando eravate al buio e a momenti finivate in pasto a quelle orribili creature. Ora, la brillante luce gialla emessa dalla lampada le fa scappare via, ma il passaggio è bloccato da una massiccia porta di pietra gialla.

§3.5 La porta di pietra (gialla è più carina)

🔴 In Inform bisogna distinguere due tipi di oggetti: QUELLI CHE FUNGONO DA SCENARIO, e QUELLI CHE FUNGONO DA CONTORNO.

Quelli che fungono da scenario (nel nostro caso Forest, Square_Chamber, Corridor) si riconoscono perché si richiamano tra di loro attraverso le istruzioni di direzione viste nel paragrafo 3.3. Questi oggetti non possono essere esaminati o manipolati dal giocatore e su di essi non si usa la proprietà initial.

All'inizio del gioco, la prima cosa che appare è la descrizione della foresta:

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

come è giusto che sia, la curiosità porta spesso a voler esaminare in dettaglio qualche oggetto citato, come ad esempio gli ulivi cupi:

>esamina gli ulivi cupi
Non vedi nulla del genere.

ma la risposta non sembra essere delle migliori. Ora, dal momento che l'oggetto Forest funge da scenario, se vogliamo che Inform "veda" gli ulivi cupi dobbiamo definire un nuovo oggetto:

```
Object  Ulivi "ulivi cupi" Forest
  with  article "degli",
        name 'ulivi' 'cupi' 'uliveti' 'alberi',
        description
            "Potresti farci dell'ottimo olio d'oliva.",
  has   pluralname scenery;
```

e a questo punto il problema è risolto[13]:

>esamina gli ulivi cupi
Potresti farci dell'ottimo olio d'oliva.

Gli oggetti che fungono da contorno, invece, possono appartenere a uno scenario o al giocatore stesso, e possono essere esaminati e manipolati da quest'ultimo (a meno che, ovviamente, non abbiano l'attributo static o scenery). Praticamente, quasi tutti quelli che abbiamo visto finora.

[13] Questo problema può essere risolto in maniera più efficace tramite l'utilizzo della libreria scenic.h, che dà la possibilità di attribuire a un oggetto dei nomi "scenici" che estendono la descrizione di default. Ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 4.9.

Passiamo ora a esaminare l'oggetto sodium_lamp (la lampada al sodio):

```
Object  sodium_lamp "lampada al sodio"
  with  name 'sodio' 'lampada' 'pesante',
        describe [;
            if (self has on)
                "^La lampada al sodio, posata a terra, si sta lentamente
                  consumando.";
            "^La lampada al sodio @`e posata saldamente a terra.";
            ],
        before [;
            Examine:
                print "@`E una pesante e robusta lampada da archeologo,";
                if (self hasnt on) "attualmente spenta.";
                if (self.battery_power < 10) "che emette una fioca luce
                    gialla.";
                "che risplende di una brillante luce gialla.";
            Burn:
                <<SwitchOn self>>;
            SwitchOn:
                if (self.battery_power <= 0)
                    "Sfortunatamente, le batterie sembrano essere
                     scariche.";
                if (parent(self) hasnt supporter && self notin location)
                    "La lampada deve essere ben posizionata prima di
                     essere accesa.";
            Take, Remove:
                if (self has on)
                    "La lampadina @`e troppo fragile e la maniglia
                      metallica @`e troppo calda per poter alzare la
                      lampada mentre @`e ancora accesa.";
            PushDir:
                AllowPushDir();
                rtrue;
            ],
        after [;
            SwitchOn:
                give self light;
            SwitchOff:
                give self ~light;
            ],
        daemon [;
            if (self hasnt on) return;
            if (--self.battery_power == 0) give self ~light ~on;
            if (self in location) {
                switch (self.battery_power) {
                    10: "^La lampada al sodio diventa sempre pi@`u
                         debole!";
                    5:  "^La lampada al sodio non far@`a luce ancora per
                         molto.";
                    0:  "^La lampada al sodio si affievolisce e
                         improvvisamente si spegne.";
                    }
                }
            ],
        battery_power 100,
  has   female switchable;
```

L'attributo switchable indica che l'oggetto in questione può essere acceso (on) o spento (off), mentre la variabile locale battery_power funge da "batteria", e viene decrementata di 1 a ogni mossa del giocatore con la lampada accesa (operazione che, come già sapete, si effettua tramite un daemon). Quando battery_power vale zero (dopo cioè che il giocatore ha effettuato 100 mosse con la lampada accesa), la batteria si scarica e la lampada non può più essere usata.

Per quanto riguarda la proprietà describe, il suo messaggio viene stampato sempre (mentre il messaggio della proprietà initial non viene più stampato una volta che l'oggetto è stato rimosso). Se ad esempio il giocatore si trova nel Grande Altopiano e posa la lampada accesa a terra, l'oggetto sodium_lamp viene rimosso dall'inventario e inserito nella locazione Forest. Se vogliamo quindi che Inform stampi a video il messaggio "La lampada al sodio, posata a terra, si sta lentamente consumando.", dobbiamo usare questa proprietà che ha, tra l'altro, la priorità su initial.

Per quanto riguarda invece l'azione SwitchOn (usata per stabilire quello che accade quando l'oggetto a cui si riferisce è acceso), dobbiamo dare un'occhiata a una particolare condizione if che le appartiene: (parent(self) hasnt supporter && self notin location)
Questo è un modo un po' complicato per far sì che Inform si chieda se la lampada, quando deve essere accesa, si trovi nell'inventario del giocatore oppure no. L'istruzione parent(self) ci permette di sapere, in qualsiasi momento, chi è il "possessore" della lampada al sodio: se si trova nell'inventario il parent (che in inglese significa "genitore" e non "parente") è il giocatore, mentre se si trova in una locazione il parent è la locazione stessa. Quindi, se ci troviamo ad esempio nel Grande Altopiano e posiamo la lampada a terra, il parent di quest'ultima sarà l'oggetto Forest.
L'attributo supporter si usa per quegli oggetti su cui è possibile posare qualcosa (come ad esempio un tavolo o il pavimento); quindi, ritornando alla nostra condizione if, se l'oggetto che possiede la lampada non ha l'attributo supporter **e** se la lampada non si trova in quella locazione (e si trova invece nell'inventario del giocatore che quindi non l'ha posata) non è possibile accenderla. Tutto questo per dire che la lampada, per poter essere accesa, deve essere prima posata a terra.
Sempre per quanto riguarda l'azione SwitchOn, un'altra cosa molto interessante da esaminare è l'azione PushDir che permette al giocatore di spostare l'oggetto a cui si riferisce con un comando del tipo SPINGI [OGGETTO] A [DIREZIONE]. Le regole principali da seguire sono tre:

1. descriverla sempre sotto la proprietà before;
2. richiamare sempre l'istruzione AllowPushDir();
3. restituire sempre il valore true con l'istruzione rtrue;

La porta di pietra, invece, è un tipo di door leggermente più complessa di quella vista nel paragrafo 3.3 (i gradini scolpiti nella roccia, ricordate?):

```
Object  -> StoneDoor "porta di pietra"
  with  name 'porta' 'massiccia' 'grande' 'pietra' 'gialla',
        description
            "@`E solo una grossa porta di pietra.",
        when_closed
            "Il passaggio @`e bloccato da una massiccia porta di pietra
             gialla.",
        when_open
            "La grande porta di pietra gialla @`e aperta.",
        door_to [; if(self in Corridor)return Shrine; return Corridor; ],
        door_dir [; if (self in Shrine) return n_to; return s_to; ],
        with_key stone_key,
        found_in Corridor Shrine,
  has   female static door openable lockable locked;
```

La prima cosa da notare è un'ulteriore finezza all'interno della proprietà description, perché è previsto un messaggio di descrizione sia quando la porta è chiusa (when_closed) che aperta (when_open). Altra caratteristica che la riguarda, è il fatto che è presente sia nel Corridoio che nella stanza del Tempio. Ecco quindi cosa accade nella proprietà door_to: se la porta (self) è nel Corridor (se il giocatore, cioè, si trova nel corridoio) ritorna Shrine (la porta dà sulla stanza del tempio); altrimenti (se il giocatore, cioè, si trova invece nella stanza del tempio) ritorna Corridor (la porta dà sul corridoio in pendenza). Per quanto riguarda la proprietà door_dir, se il giocatore si trova nella stanza del tempio la porta di pietra si trova a nord, viceversa se il giocatore si trova nel corridoio la porta di pietra si trova a sud[14].
Occorre poi, spendere ancora due parole su due nuovi attributi:

- lockable che permette a un oggetto di essere chiuso o meno a chiave (in questo caso con una chiave di pietra, come specificato nell'istruzione with_key stone_key);
- locked che specifica che l'oggetto a cui appartiene è chiuso a chiave.

Ma la chiave di pietra dov'è? È quello che vedremo nel prossimo paragrafo.

[14] È possibile attuare una gestione semplificata di una door grazie alla librerie doors.h e easydoors.h. Ulteriori informazioni sul loro utilizzo potete trovarle al paragrafo 4.9.

Avete sempre odiato le porte chiuse, soprattutto quelle di pietra. Avete tentato di tutto, dal pronunciare arcane formule magiche (Apriti Sesamo) a prenderla a calci e pugni, ma a parte il dolore lancinante alle mani e ai piedi, non avete risolto un bel nulla: la porta continua a non aprirsi.
Ritornate nella Sala Quadrata, sconsolati e sempre più decisi a rinunciare alla spedizione, quando all'improvviso vi ricordate che verso est c'è un altro passaggio che non avete ancora percorso. Vi dirigete verso quella direzione e vi ritrovate in quella che a prima vista sembra essere la tana di un verme gigante: un groviglio di cunicoli disordinati come una ragnatela si dirige verso le fessure tra le pietre e, appiccicato alla fessura di una parete, c'è un bozzolo bianco e scintillante grande come un pallone da spiaggia. Lo raccogliete (non senza un certo ribrezzo) e ritornate nella Sala Quadrata: qui, il bozzolo vi scivola dalle mani e cade nel bagliore solare, dove ribolle oscenamente, si dilata e poi scoppia. Centinaia di piccoli insetti corrono in tutte le direzioni nell'oscurità; gli spruzzi di melma e una curiosa chiave di pietra gialla sono tutto ciò che rimane sul pavimento.
Ma che fortuna! Raccogliete la chiave, correte verso la porta di pietra e… incredibile, ora si apre e potete finalmente proseguire: chi è Indiana Jones al vostro confronto?

§3.6 La chiave di pietra (così non si arrugginisce)

La Tana del Verme è un oggetto che si trova a est della Sala Quadrata:

```
Object  Wormcast "Tana Del Verme"
  with  description
            "Un groviglio di cunicoli disordinati come una ragnatela si
             dirige verso le fessure tra le pietre. I soli abbastanza
              larghi da poterci strisciare dentro sono quelli che si
              dirigono verso l'alto, a nordest e a sud.",
        w_to Square_Chamber,
        s_to [;
            print "La tana diventa scivolosa intorno a te, come se il
                   calore del tuo corpo stesse sciogliendo le resine
                   indurite, e chiudi gli occhi con forza mentre
                   precipiti nelle tenebre...^";
            if (eggsac in player) return Square_Chamber;
            return random(Square_Chamber, Corridor, Forest);
            ],
        ne_to [; return self.s_to(); ],
        u_to [; return self.s_to(); ],
        after [;
            Drop:
                move noun to Square_Chamber;
                print_ret (The)noun," scivola attraverso uno dei cunicoli
                          e ", (itorthem) noun, " perdi rapidamente di
                          vista.";
            ],
  has   light;
```

La particolarità che lo contraddistingue da quanto visto finora, si può mettere in evidenza quando il giocatore si dirige verso sud; in questo caso, se questi ha preso il bozzolo (un oggetto che esamineremo fra poco) allora ritorna nella Sala Quadrata, altrimenti (se il bozzolo non è ancora stato raccolto) il giocatore ritorna, a caso, o nella Sala Quadrata, o nel Corridoio in pendenza o nel Grande Altopiano. Se invece viene posato a terra un oggetto qualsiasi, viene immediatamente spostato nella Sala Quadrata.

È importante osservare che la funzione di stampa (The) stampa a video l'articolo determinativo (Il, Lo, La, I, Gli, Le) ⟶ con l'iniziale maiuscola) opportunamente correlato con l'oggetto associato (in questo caso l'oggetto posato dal giocatore), mentre l'istruzione itorthem (che appartiene a Infit e non a Inform) stampa lo, la, li, le a seconda del genere (maschile o femminile) e del numero (singolare o plurale)[15]. La funzione di stampa (the) ha la stessa funzione della (The) con la differenza che la lettera iniziale dell'articolo determinativo è minuscola anziché maiuscola. La funzione di stampa (A), invece, stampa a video l'articolo indeterminativo (Un, Uno, Una, Un') ⟶ con l'iniziale maiuscola) dell'oggetto ad essa associato, mentre la funzione di stampa (a) ha la stessa funzione della (A) con la differenza che la lettera iniziale dell'articolo indeterminativo è minuscola anziché maiuscola.

L'istruzione print_ret stampa a video il relativo messaggio ritornando poi il valore true, e rappresenta l'equivalente del seguente codice:

```
print (The)noun," scivola attraverso uno dei cunicoli
                  e ", (itorthem) noun, " perdi rapidamente di vista.";
rtrue; ! o return true;
```

La sua funzione, in questo caso, è quella di impedire che venga stampato a video il messaggio di default relativo all'azione Drop, come accade se proviamo a usare la sola istruzione print:

```
Tana Del Verme
Un groviglio di cunicoli disordinati come una ragnatela si dirige verso le fessure tra le pietre. I soli abbastanza larghi da poterci strisciare dentro sono quelli che si dirigono verso l'alto, a nordest e a sud.

Un bozzolo bianco e scintillante, grande come un pallone da spiaggia, è appiccicato alla fessura di una parete.

>posa la macchina
La macchina fotografica a lastre scivola attraverso uno dei cunicoli e la perdi rapidamente di vista.Posata.

>
```

Non restituendo più il valore true, Inform stampa anche la parola Posata (il messaggio di default, appunto).

Passiamo ora a esaminare il bozzolo citato poco fa:

```
Object  -> eggsac "bozzolo bianco scintillante",
  with  name 'uovo' 'sacchetto' 'sacco' 'bozzolo',
        initial
            "Un bozzolo bianco e scintillante, grande come un pallone da
             spiaggia, @`e appiccicato alla fessura di una parete.",
        after [;
            Take: "Bleah!";
            ],
        react_before [;
            Go:
                if (location == Square_Chamber && noun == u_obj) {
                    deadflag = true;
                    "Non appena il bozzolo @`e illuminato dalla luce
                     naturale, ribolle oscenamente e si dilata. Prima che
                     tu possa gettarlo via, scoppia in centinaia di
                     piccoli insetti affamati...";
```

[15] Per ulteriori informazioni su come usare le funzioni di Infit, consultate la breve guida contenuta all'interno del file Infit25.zip.

```
                }
            ];
```

Come potete vedere, questo oggetto non è poi così complicato. L'unica cosa da notare, a livello di codice, è che se il giocatore raccoglie il bozzolo e dalla Sala Quadrata si dirige verso l'alto, il bozzolo scoppia e il giocatore viene divorato dai piccoli insetti affamati: un altro modo per farsi uccidere…
Per fare poi in modo che dal bozzolo "esca" la chiave di pietra gialla, occorre aggiungere un altro pezzo di codice all'oggetto Square_Chamber:

```
before [;
            Insert:
                if (noun == eggsac && second == sunlight) {
                    remove eggsac;
                    move stone_key to self;
                    "Lasci cadere il bozzolo nel bagliore solare. Ribolle
                     oscenamente, si dilata e poi scoppia. Centinaia di
                     piccoli insetti corrono in tutte le direzioni
                     nell'oscurit@`a; gli spruzzi di melma e una curiosa
                     chiave di pietra gialla sono tutto ci@`o che rimane
                     sul pavimento.";
                    }
            ],
```

In poche parole, quando il giocatore POSA IL BOZZOLO (noun) NEL RAGGIO (second), il bozzolo (l'oggetto eggsac) viene rimosso tramite l'istruzione remove e viene sostituito con la chiave di pietra (l'oggetto stone_key) mediante l'istruzione move. Nulla di complicato, come potete vedere.
Qualche difficoltà comincia invece ad esserci nel momento in cui il giocatore deve aprire la porta di pietra con la chiave. Il problema nasce dal fatto che in italiano, a differenza dell'inglese, il verbo APRI può essere usato a livello di comando sia quando la porta è chiusa a chiave (APRI LA PORTA CON LA CHIAVE) che quando invece non lo è (APRI LA PORTA). Inform invece, gestisce questi eventi con due verbi differenti (unlock e open) e se omettiamo il nuovo pezzo di codice presente all'inizio del programma:

```
Object LibraryMessages
    with before [;
        Unlock:
          if (lm_n == 4) {
              "Ora ", (the) noun, " non @`e pi@`u chius",
              (genderandnumber) noun, " a chiave.";
          }
    ];
```

ecco quello che accade:

```
Corridoio in pendenza
Un corridoio basso e squadrato va da nord verso sud, inclinandosi verso la fine.

Il passaggio è bloccato da una massiccia porta di pietra gialla.

>apri la porta con la chiave
Ora hai aperto la porta di pietra.

>g
```

Corridoio in pendenza
Un corridoio basso e squadrato va da nord verso sud, inclinandosi verso la fine.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Il passaggio è bloccato da una massiccia porta di pietra gialla.

>

Come potete vedere, quando il giocatore dà il comando APRI LA PORTA CON LA CHIAVE, viene stampato a video il messaggio "Ora hai aperto la porta di pietra." che però non dice la verità, perché la porta è ancora chiusa. Questo problema può essere risolto facilmente andando a personalizzare il messaggio di libreria relativo al verbo Unlock, sostituendo la frase "Ora hai aperto la porta di pietra." con "Ora la porta di pietra non è più chiusa a chiave.":

Corridoio in pendenza
Un corridoio basso e squadrato va da nord verso sud, inclinandosi verso la fine.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Il passaggio è bloccato da una massiccia porta di pietra gialla.

>apri la porta con la chiave
Ora la porta di pietra non è più chiusa a chiave.

>apri la porta
Ora hai aperto la porta di pietra.

>g

Corridoio in pendenza
Un corridoio basso e squadrato va da nord verso sud, inclinandosi verso la fine.

La lampada al sodio, posata a terra, si sta lentamente consumando.

La grande porta di pietra gialla è aperta.

>

Adesso le cose vanno decisamente meglio, non vi pare?
genderandnumber è un'altra istruzione appartenente a Infit, e riconosce il genere e il numero di un oggetto. Inoltre, viene testato il numero 4 perché il verbo Unlock ha diversi tipi di messaggi, e il quarto è proprio quello che ci interessa[16].

🔴 Ricordatevi anche che QUANDO VOLETE PERSONALIZZARE UN MESSAGGIO DI LIBRERIA, DOVETE INSERIRE L'OGGETTO LibraryMessages TRA LE RIGHE DI CODICE Include "parser" E Include "verblib", proprio come accade in questo esempio:

```
Include "Parser";

Object LibraryMessages
    with before [;
         Sing:
           "Smettila, accidenti a te...";
        Save:
          if (lm_n == 1) "La partita non è stata salvata.";
```

[16] Potete trovare un elenco completo dei messaggi della libreria all'appendice B.

```
        Miscellany:
          if (lm_n == 37) "Non ottieni nessuna risposta.";
    ];

Include "VerbLib";
Include "replace";
```

Oltre alle azioni, è possibile personalizzare anche i messaggi di errore (testati sotto la voce Miscellany): l'errore numero 38, per esempio, si verifica quando il giocatore vuole eseguire un'azione con un verbo che non esiste (che non è stato, cioè, dichiarato in Infit o all'interno dell'avventura stessa → es.: ALLAGA LA CASA).
Già che ci siamo, vediamo come si può personalizzare anche il prompt dei comandi. Per fare in modo che al posto del segno di maggiore appaia una frase del tipo "Cosa vuoi fare?", occorre creare un oggetto LibraryMessages nel seguente modo:

```
Object LibraryMessages
    with before [;
        Prompt: print "^Cosa vuoi fare?";
    ];
```

Infit, inoltre, mette a disposizione quindici domande: per usarle, basta inserire l'istruzione constant PROMPT; all'inizio del programma e Inform ne sceglierà una a caso ogni volta che il giocatore effettuerà una nuova mossa.
È perfino possibile stabilire il tipo di prompt da usare in base a una determinata mossa del giocatore, come accade in quest'altro esempio:

```
Object LibraryMessages
    with before [;
        Prompt: switch (turns) {
           1: print "^E adesso?^>";
           2 to 5: print "^Cosa vuoi fare?^>";
           default: print "^Sono ai tuoi ordini:^>";
        }
        rtrue;
    ];
```

turns è una variabile di libreria che contiene il numero delle mosse effettuate dal giocatore. Dopo la quinta mossa, il messaggio del prompt ("Sono ai tuoi ordini:") non cambia più fino alla fine del gioco.

Siete pronti a dirigervi verso sud, impazienti di vedere cosa nasconde l'imponente porta di pietra, ma c'è uno strano oggetto che attira la vostra attenzione: una statuetta minacciosa di uno spirito pigmeo, di aspetto quasi grottesco, che giace per terra alla vostra destra.
Se non fosse per quel serpente intorno al collo, potreste usarla come soprammobile, ma decidete comunque di prenderla e riporla nella cassa, se non altro per non tornare a mani vuote da quelli della fondazione. Posate quindi a terra tutti gli oggetti che possedete, ad eccezione dell'elefantiaca macchina fotografica a lastre che sistemate invece accanto alla lampada al sodio. Mettete poi in posa la statuetta… et voilà, la foto è fatta. E mentre ritornate verso la superficie, vi chiedete quando inventeranno la Polaroid.

§3.7 Il primo tesoro (speriamo che non sia un anello)

In Inform esiste la possibilità di creare delle azioni nuove. Quelle elencate nell'appendice A infatti, sono "solo" quelle standard ma se, come nel nostro caso, volessimo fotografare un oggetto?
Come potete vedere, questo verbo non c'è, ed ecco perché proprio verso la fine del listato troviamo la seguente riga di codice:

```
Verb 'fotografa' * noun                              -> Photograph;
```

che può essere così interpretata: il verbo (Verb) FOTOGRAFA ('fotografa') [OGGETTO] (* noun) equivale all'azione PHOTOGRAPH (-> Photograph;). Ricordatevi però, che OGNI NUOVA AZIONE HA BISOGNO DI UN'APPOSITA SUBROUTINE DEFINITA COME [NOME_AZIONE+SUB] (nel nostro specifico caso PhotographSub):

```
[ PhotographSub;
        if (camera notin player) "Non puoi farlo senza la tua macchina
            fotografica.";
        if (noun == player) "Meglio di no. Non ti sei fatto la barba da
                             quando hai lasciato il Messico.";
        if (children(player) > 1)
            "Fare fotografie @`e una cosa impegnativa, che necessita
             dell'uso di entrambe le mani. Devi posare qualcosa.";
        if (location == Forest) "In questa foresta inzuppata di pioggia?
                                 Meglio di no.";
        if (location == thedark) "@`E assolutamente troppo buio.";
        if (AfterRoutines()) return;
        print_ret "Prepari l'elefantiaca macchina fotografica a lastre,
                   sistemi la lampada al sodio e metti pazientemente in
                   posa ",(the) noun, ".";
        ];
```

Se il giocatore non possiede la macchina fotografica (camera notin player) Inform stampa a video il messaggio corrispondente ("Non puoi farlo senza la tua macchina fotografica."); se prova invece a fotografare se stesso (FOTOGRAFA ME STESSO) viene stampato a video il messaggio "Meglio di no. Non ti sei fatto la barba da quando hai lasciato il Messico.", un pizzico di umorismo che mette però in risalto un'altra importante caratteristica di Inform: il player.
Esso infatti, viene considerato come un oggetto vero e proprio, che di default ha la seguente descrizione:

```
>esamina me stesso
Hai sempre lo stesso bell'aspetto.
```

Se poi il giocatore possiede più di un oggetto (children(player) > 1), ecco che diventa impossibile fare delle fotografie. I children (in inglese 'bambini' o 'figli') del player sono proprio tutti gli oggetti che lui possiede. E se avete capito la descrizione della lampada al sodio, dovreste già sapere che gli oggetti del player hanno come parent il player stesso. Chiaro, no?

🔴 Dal momento però, che questo povero archeologo non si è fatto la barba da quando ha lasciato il Messico, non credo che questo messaggio corrisponda proprio alla verità. Come vedete, è molto importante personalizzare "l'ambiente", se non altro per rendere più veritiera l'avventura in sé.
Ecco quindi come possiamo modificare anche questo messaggio:

```
[ Initialise;
        location = Forest;
        .
        .
        .
        player.description = "Assomigli proprio a Matusalemme.";
        .
        .
        .
        "^^^Dopo giorni di inutili ricerche, passati senz'acqua
         attraversando i rovi della foresta, alla fine la tua pazienza
         @`e stata ricompensata: hai fatto una scoperta!^";
        ];
```

ottenendo così il seguente risultato:

>esamina me stesso
Assomigli proprio a Matusalemme.

Volendo, è perfino possibile aggiungere delle parti del corpo:

```
.
.
.
Object  tiny_claws "suono di piccoli artigli" thedark...

Object nose "naso"
  with name 'naso',
       description
          "A punta e pieno di lentiggini.";

[ Initialise;
        location = Forest;
        selfobj.add_to_scope = nose;
        move map to player;
        move sodium_lamp to player;
        move dictionary to player;
        StartDaemon(sodium_lamp);
        thedark.description =
            "L'oscurit@`a intorno a te @`e opprimente e ti senti
             quasi soffocare.";
        "^^^Dopo giorni di inutili ricerche, passati senz'acqua
         attraversando i rovi della foresta, alla fine la tua pazienza
         @`e stata ricompensata: hai fatto una scoperta!^";
        ];
```

In questo esempio abbiamo definito il naso. Di per sé, l'oggetto nose non ha nulla di particolare, mentre la riga selfobj.add_to_scope = nose; contenuta nella funzione Initialise è quella che dice a Inform di assegnare l'oggetto in questione al player (selfobj). Il risultato che si ottiene è il seguente:

>esamina il mio naso
A punta e pieno di lentiggini.

Nel caso in cui si vogliano aggiungere più parti del corpo, si deve allora ricorrere al seguente stratagemma:

```
.
.
.
Object  tiny_claws "suono di piccoli artigli" thedark...

Object nose "naso"
  with name 'naso',
       description
          "A punta e pieno di lentiggini.";

Object hair "capelli"
  with name 'capelli' 'capello',
       description
          "Li hai persi tutti quando eri piccolo.";

[ IncludeBodyParts; PlaceInScope(nose); PlaceInScope(hair); ];

[ Initialise;
        location = Forest;
        selfobj.add_to_scope = IncludeBodyParts;
        move map to player;
        move sodium_lamp to player;
        move dictionary to player;
        StartDaemon(sodium_lamp);
        thedark.description =
            "L'oscurit@`a intorno a te @`e opprimente e ti senti
             quasi soffocare.";
        "^^^Dopo giorni di inutili ricerche, passati senz'acqua
         attraversando i rovi della foresta, alla fine la tua pazienza
         @`e stata ricompensata: hai fatto una scoperta!^";
        ];
```

Quella che viene adesso associata al player è l'intera funzione IncludeBodyParts al cui interno vengono resi visibili, tramite l'istruzione PlaceInScope, il naso e i capelli. Ecco allora il risultato:

>esamina i miei capelli
Li hai persi tutti quando eri piccolo.

>esamina il mio naso
A punta e pieno di lentiggini.

Provare per credere.

La rimanente parte del codice dovrebbe essere ormai abbastanza chiara, tranne che per quella strana istruzione denominata AfterRoutines. Una cosa molto importante da tenere a mente quando si descrive una nuova azione, è a quale gruppo essa deve appartenere (in genere 2 o 3). Ora, come dice

Paolo Lucchesi[17], se l'azione non fa praticamente nulla (ed è quindi molto semplice), appartiene al gruppo 3 e dipende dalla proprietà before:

```
[ XyzzySub; "Non succede nulla."; ];

Verb "xyzzy"    *                                -> Xyzzy;
```

Se invece l'azione compie delle operazioni complesse, appartiene allora al gruppo 2 e dipende dalla proprietà after. Quindi, dopo quanto detto, è facile dedurre che l'azione Photograph fa parte del gruppo 2, e l'istruzione AfterRoutines si prende cura di tutto quello che avviene in questa azione quando fa parte della proprietà after di un oggetto qualsiasi: se ritorna true, tutto è andato bene, se invece ritorna false, qualcosa è andato storto:

```
Object -> statuette "statuetta pigmea"
  with  name 'serpente' 'maya' 'pigmea' 'spirito' 'preziosa' 'statuetta'
             'statua',
        initial
            "C'@`e una preziosa statuetta maya qui!",
        description
            "@`E una statuetta minacciosa di uno spirito pigmeo di
             aspetto quasi grottesco. Ha un serpente intorno al collo.",
        before [;
            Take, Remove:
                if (self in packing_case)
                    "@`E meglio aspettare che sia la fondazione Carneige
                     a disimballare dalla cassa un manufatto cos@`i
                     prezioso.";
                if (self.photographed_in_situ == false)
                    "Questi sono gli anni '30 e non i tempi andati.
                     Prendere un manufatto senza prima registrarlo
                     equivale a un saccheggio.";
            Photograph:
                if (self has moved) "Cosa? Vorresti contraffare una
                    registrazione archeologica?";
                if (self.photographed_in_situ) "Non di nuovo.";
            ],
        after [;
            Insert:
                if (second == packing_case) {
                    score = score + self.cultural_value;
                    if (score == MAX_SCORE) deadflag = 2;
                    "Depositat", (genderandnumber) noun, " al sicuro!";
                    }
            Photograph:
                self.photographed_in_situ = true;
            ],
        cultural_value 5,
        photographed_in_situ false,
  has   female;
```

[17] Autore di "La pietra della Luna", Paolo Lucchesi è anche il creatore di MAC (Mistery Adventure Creator), uno strumento di creazione per le avventure testuali rivolto ai meno esperti della programmazione. Ulteriori informazioni potete trovarle all'indirizzo http://www.paololucchesi.it/).

La variabile locale photographed_in_situ vale inizialmente false. Quando il giocatore riesce in qualche modo a fotografare la statuetta, il valore della variabile diventa true. Ora, se omettiamo la riga di codice relativa alla AfterRoutines() in PhotographSub, ecco cosa accade:

Corridoio in pendenza
Un corridoio basso e squadrato va da nord verso sud, inclinandosi verso la fine.

Il passaggio è bloccato da una massiccia porta di pietra gialla.

C'è una preziosa statuetta maya qui!

>prendi la statuetta
Questi sono gli anni '30 e non i tempi andati. Prendere un manufatto senza prima registrarlo equivale a un saccheggio.

>fotografa la statuetta
Fare fotografie è una cosa impegnativa, che necessita dell'uso di entrambe le mani. Devi posare qualcosa.

>posa tutto eccetto la macchina
chiave di pietra: Posata.
dizionario maya di Waldeck: Posato.
mappa di Quintana Roo: Posata.

>fotografa la statuetta
Prepari l'elefantiaca macchina fotografica a lastre, sistemi la lampada al sodio e metti pazientemente in posa la statuetta pigmea.

>prendi la statuetta
Questi sono gli anni '30 e non i tempi andati. Prendere un manufatto senza prima registrarlo equivale a un saccheggio.

Nonostante la statuetta sia stata fotografata, non si può ancora prendere. Ecco quindi spiegato il motivo della presenza della AfterRoutines: senza, viene ignorata l'azione Photograph della proprietà after dell'oggetto statuette, la variabile locale photographed_in_situ continua a valere false e di conseguenza la statuetta continua a non poter essere fotografata (e a non poter essere presa dal giocatore).

* *

Occupiamoci adesso del punteggio:

```
Constant MAX_SCORE = 30;
.
.
.
Object -> statuette "statuetta pigmea"
        .
        .
        .
        after [;
            Insert:
                if (second == packing_case) {
                    score = score + self.cultural_value;
                    if (score == MAX_SCORE) deadflag = 2;
                    "Depositat", (genderandnumber) noun, " al sicuro!";
                    }
              .
              .
```

```
              .
              ],
         cultural_value 5,
         .
         .
         .
  has    female;
.
.
.
[ PrintRank;
         print ", guadagnando il rango di ";
         if (score == 30) "Direttore della Fondazione Carneige.";
         if (score >= 20) "Archeologo.";
         if (score >= 10) "Rigattiere.";
         if (score >= 5) "Esploratore.";
         "Turista.";
         ];
```

La costante MAX_SCORE determina il punteggio massimo, mentre la variabile locale cultural_value determina il punteggio della statuetta. Nella proprietà after, l'azione Insert (POSA LA STATUETTA NELLA CASSA) incrementa il valore della variabile di libreria score di 5 (il punteggio della statuetta); se score vale 30, allora il nostro archeologo ha raccolto tutti i tesori e ha finito la sua avventura nel migliore dei modi (non è stato cioè ucciso ma ha invece vinto). PrintRank è una funzione di libreria che, a seconda del punteggio, stampa a video il relativo messaggio; questo avviene però, solo se il gioco termina o se si danno i comandi PUNTEGGIO e PUNTEGGIO PIENO.

🔴 Esiste poi, una modalità "completa" del punteggio, nel senso che è possibile stabilire, per un'azione qualsiasi, quanti punti assegnarle. Se, ad esempio, vogliamo dare 5 punti al giocatore quando mangia il fungo e 10 punti quando posa il bozzolo nel raggio di sole, ecco cosa dobbiamo fare:

1) Per prima cosa, bisogna definire un array di nome task_scores nel seguente modo:

```
Constant Story "RUINS";
.
.
.
Constant MAX_CARRIED = 7;
Constant TASKS_PROVIDED;
Constant NUMBER_TASKS = 2;
Constant MAX_SCORE = 30;

Array    task_scores -> 5 10;
```

   Il numero 5 è il punteggio che viene assegnato alla prima azione (quando il giocatore mangia il fungo) mentre il numero 10 viene assegnato alla seconda azione (quando il giocatore posa il bozzolo nel raggio). A questo array, si associano sempre le costanti NUMBER_TASKS (che contiene, a sua volta, il numero dei punteggi totali - in questo caso 5 e 10 → 2) e TASKS_PROVIDED. Ricordatevi inoltre, che l'array deve avere una lunghezza massima di 255 celle (da 0 a 254) e che i punteggi non possono essere negativi o maggiori di 255.

2) Occorre poi, modificare l'azione Eat dell'oggetto mushroom come segue:

```
Eat:
    Achieved(0);
    steps.rubble_filled = false;
    "Lo sgranocchi ad un angolo, incapace di capire l'origine di un
     gusto cos@`i acre, distratto dal volo di un macao sopra la tua
```

```
        testa che sembra quasi un'esplosione nel sole. Il battito delle
        sue ali @`e quasi assordante, e delle pietre crollano una
        sull'altra.";
```

L'istruzione Achieved(0); è proprio quella che assegna 5 punti alla prima azione, e li va a prelevare dalla posizione 0 dell'array task_scores. Una modifica abbastanza simile va fatta per l'azione Insert dell'oggetto Square_Chamber:

```
Insert:
    if (noun == eggsac && second == sunlight) {
        achieved(1);
        remove eggsac;
        move stone_key to self;
        "Lasci cadere il bozzolo nel bagliore solare. Ribolle
         oscenamente, si dilata e poi scoppia. Centinaia di piccoli
         insetti corrono in tutte le direzioni nell'oscurit@`a; gli
         spruzzi di melma e una curiosa chiave di pietra gialla sono
         tutto ci@`o che rimane sul pavimento.";
         }
```

L'unica differenza, sta nel fatto che ora l'istruzione Achieved va a prelevare il valore 10 anziché il valore 5.
Per verificare che le azioni di assegnamento dei punteggi abbiano avuto effettivamente luogo, basta testarle con l'istruzione task_done:

```
if (task_done->0 == 0)... ! se i 5 punti non sono stati assegnati...
if (task_done->0 == 1)... ! se i 5 punti sono stati assegnati...
if (task_done->1 == 0)... ! se i 10 punti non sono stati assegnati...
if (task_done->1 == 1)... ! se i 10 punti sono stati assegnati...
```

Vediamo allora in pratica quanto è stato appena detto:

>prendi il fungo
Hai raccolto abilmente il fungo, senza staccarlo dal suo gambo sottile.

>mangia il fungo
Lo sgranocchi ad un angolo, incapace di capire l'origine di un gusto così acre, distratto dal volo di un macao sopra la tua testa che sembra quasi un'esplosione nel sole. Il battito delle sue ali è quasi assordante, e delle pietre crollano una sull'altra.

[Il tuo punteggio è appena aumentato di cinque punti.]

>giù

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>e

Tana Del Verme
Un groviglio di cunicoli disordinati come una ragnatela si dirige verso le fessure tra le pietre. I soli abbastanza larghi da poterci strisciare dentro sono quelli che si dirigono verso l'alto, a nordest e a sud.

Un bozzolo bianco e scintillante, grande come un pallone da spiaggia, è appiccicato alla fessura di una parete.

>prendi il bozzolo
Bleah!

>o

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>posa il bozzolo nel raggio
Lasci cadere il bozzolo nel bagliore solare. Ribolle oscenamente, si dilata e poi scoppia. Centinaia di piccoli insetti corrono in tutte le direzioni nell'oscurità; gli spruzzi di melma e una curiosa chiave di pietra gialla sono tutto ciò che rimane sul pavimento.

[Il tuo punteggio è appena aumentato di dieci punti.]

>punteggio
Finora hai totalizzato 15 punti su 30 possibili, in 8 turni, guadagnando il rango di Rigattiere.

>

🔴 Bisogna anche considerare, come dice lo stesso Paolo Lucchesi, la modalità a punteggio pieno, dove è necessario inserire la funzione di libreria PrintTaskName:

```
[ Initialize...
.
.
.
[ TitlePage...
.
.
.
[ PrintTaskName achievement;
  switch(achievement)
  {   0: "hai mangiato il fungo";
      1: "hai trovato la chiave di pietra";
  }
];
```

In poche parole, se il giocatore ha mangiato il fungo e ha posato il bozzolo nel raggio di sole, ecco cosa stampa a video Inform con il comando PUNTEGGIO PIENO:

>punteggio pieno
Finora hai totalizzato 15 punti su 30 possibili, in 8 turni, guadagnando il rango di Rigattiere.

Il punteggio è così composto:

 5 hai mangiato il fungo
10 hai trovato la chiave di pietra

15 in totale (su 30 possibili)

>

A ogni modo, non siete obbligati a supportare questa funzione nella gestione del punteggio in modalità completa; se il giocatore, durante il gioco, dà questo comando senza che la funzione PrintTaskName

sia stata implementata, Inform stampa a video lo stesso messaggio previsto per il comando PUNTEGGIO.

🔴 Non sempre le azioni effettuate dal giocatore meritano di essere premiate. Ecco allora come procedere per far sì che il punteggio possa anche essere diminuito:

1) La prima cosa da fare è quella di definire un array di nome task_scores nel seguente modo:

```
Constant Story "RUINS";
.
.
.
Replace TaskScore;

Constant MAX_CARRIED = 7;
Constant TASKS_PROVIDED;
Constant NUMBER_TASKS = 2;
Constant MAX_SCORE = 30;

Array task_scores --> 10 (-5);
```

2) Dal momento che è presente l'istruzione replace, occorre poi ridefinire la funzione di libreria TaskScore:

```
.
.
.
[ PrintRank;...

[ TaskScore i; return task_scores-->i; ];
```

Ecco fatto. Ora potete sia punire che premiare il giocatore a seconda dell'azione da lui effettuata:

>prendi il fungo
Hai raccolto abilmente il fungo, senza staccarlo dal suo gambo sottile.

>mangia il fungo
Lo sgranocchi ad un angolo, incapace di capire l'origine di un gusto così acre, distratto dal volo di un macao sopra la tua testa che sembra quasi un'esplosione nel sole. Il battito delle sue ali è quasi assordante, e delle pietre crollano una sull'altra.

[Il tuo punteggio è appena aumentato di dieci punti.]

>giu

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>e

Tana Del Verme
Un groviglio di cunicoli disordinati come una ragnatela si dirige verso le fessure tra le pietre. I soli abbastanza larghi da poterci strisciare dentro sono quelli che si dirigono verso l'alto, a nordest e a sud.

Un bozzolo bianco e scintillante, grande come un pallone da spiaggia, è appiccicato alla fessura di una parete.

>prendi il bozzolo
Bleah!

>o

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>posa il bozzolo nel raggio
Lasci cadere il bozzolo nel bagliore solare. Ribolle oscenamente, si dilata e poi scoppia. Centinaia di piccoli insetti corrono in tutte le direzioni nell'oscurità; gli spruzzi di melma e una curiosa chiave di pietra gialla sono tutto ciò che rimane sul pavimento.

[Il tuo punteggio è appena diminuito di cinque punti.]

>

Supponendo per assurdo che il bozzolo debba in qualche modo rimanere intatto, se il giocatore lo lascia cadere nel bagliore solare il punteggio viene diminuito di 5 punti.

🔴 Definendo, infine, la costante NO_SCORE nel seguente modo:

```
Constant Story "RUINS";
Constant Headline
         "^Un esempio di lavoro interattivo.^
          Copyright (c) 1999 di Graham Nelson.^
         Traduzione e adattamenti di Vincenzo Scarpa e Raffaello
         Valesio (c) 2002-2003 su permesso dell'autore.^^";

Constant MAX_CARRIED = 7;
Constant NO_SCORE;
   .
   .
   .
```

è possibile disabilitare il punteggio, che non viene più segnalato sulla status line. Inoltre, perdono la loro efficacia le funzioni di libreria PrintRank; e PrintTaskName;

Dopo aver riposto la statuetta pigmea nella cassa d'imballaggio, ritornate nel corridoio e, spingendo la lampada verso sud, varcate la soglia di quella che, alla luce della lampada al sodio, si rivela essere la sala di un magnifico Tempio che mostra segni di scavi da preesistenti miniere di calcare. Verso il lato occidentale, due lunghi cornicioni si dirigono a sud, e delle pitture particolarmente vivide e impegnative mostrano un sovrano corazzato che calpesta un prigioniero. Il Tempio poi, è dominato da una grande lastra di pietra sulla cui superficie giace una maschera facciale in mosaico di giada. La osservate attentamente per un po', e ne deducete che è troppo bella per appartenere a una compagnia teatrale, e che quindi è autentica; soddisfatti del vostro immenso "acume archeologico", provate a indossarla (non senza averla prima fotografata) ma vi accorgete che non c'è neanche uno specchio per vedere come vi sta. Improvvisamente, attraverso gli occhi di ossidiana della maschera, qualcosa appare davanti ai vostri occhi increduli: un sacerdote mummificato, che attende che voi parliate.

§3.8 La maschera facciale (non di carnevale)

Alcuni oggetti definiti all'interno di un'avventura testuale hanno la caratteristica di poter essere indossati (e successivamente tolti) dal giocatore: ne costituiscono dei classici esempi un orologio, un paio di guanti, una maglietta e, ovviamente, una maschera, definita in Ruins come segue:

```
Treasure -> -> mask "maschera facciale in mosaico di giada"
  with  name 'giada' 'mosaico' 'facciale' 'maschera' 'faccia',
        initial
            "Una maschera facciale in mosaico di giada @`e appoggiata
             sull'altare.",
        description
            "Sarebbe stupendo se si potesse esporre nel museo.",
        after [;
            Wear:
                move priest to Shrine;
                if (location == Shrine)
                    "Guardando attraverso gli occhi di ossidiana della
                     maschera in mosaico, si rivela una presenza
                     spettrale: un sacerdote mummificato aspetta che tu
                     parli.";
            Disrobe:
                remove priest;
            ],
        cultural_value 10,
  has   female clothing;
```

Come potete vedere, un oggetto di questo tipo è contraddistinto dall'attributo clothing. Quando la maschera viene indossata (wear), l'oggetto priest (il sacerdote) viene spostato nella sala del Tempio; se poi il giocatore si trova nella sala, Inform stampa a video il relativo messaggio. Quando la maschera viene invece tolta (disrobe) il sacerdote scompare.
Ma la vera novità di questo oggetto, sta proprio nella sua definizione, perché al posto di Object troviamo ora Treasure: come mai?
È arrivato il momento di parlare delle classi. Esse si usano quando si vuole far sì che uno stesso codice venga utilizzato da più oggetti. Nel paragrafo 3.7 abbiamo definito l'oggetto statuette nel seguente modo:

```
Object -> statuette "statuetta pigmea"
  with  name 'serpente' 'maya' 'pigmea' 'spirito' 'preziosa' 'statuetta'
             'statua',
```

```
        initial
            "C'@`e una preziosa statuetta maya qui!",
        description
            "@`E una statuetta minacciosa di uno spirito pigmeo di
             aspetto quasi grottesco. Ha un serpente intorno al collo.",
        before [;
            Take, Remove:
                if (self in packing_case)
                    "@`E meglio aspettare che sia la fondazione Carneige
                     a disimballare dalla cassa un manufatto cos@`i
                     prezioso.";
                if (self.photographed_in_situ == false)
                    "Questi sono gli anni '30 e non i tempi andati.
                     Prendere un manufatto  senza prima registrarlo
                     equivale a un saccheggio.";
            Photograph:
                if (self has moved) "Cosa? Vorresti contraffare una
                                     registrazione archeologica?";
                if (self.photographed_in_situ) "Non di nuovo.";
            ],
        after [;
            Insert:
                if (second == packing_case) {
                    score = score + self.cultural_value;
                    if (score == MAX_SCORE) deadflag = 2;
                    "Depositat", (genderandnumber) noun, " al sicuro!";
                    }
            Photograph:
                self.photographed_in_situ = true;
            ],
        cultural_value 5,
        photographed_in_situ false,
  has   female;
```

Nella nostra avventura, la parte di codice relativa alle proprietà after e before è quella da mettere in comune tra più oggetti (i diversi tesori che il giocatore deve raccogliere e riporre nella cassa per poterla terminare). Ecco perché è stata definita la classe Treasure:

```
Class   Treasure
  with  before [;
            Take, Remove:
                if (self in packing_case)
                    "@`E meglio aspettare che sia la fondazione Carneige
                     a disimballare dalla cassa un manufatto cos@`i
                     prezioso.";
                if (self.photographed_in_situ == false)
                    "Questi sono gli anni '30 e non i tempi andati.
                     Prendere un manufatto senza prima registrarlo
                     equivale a un saccheggio.";
            Photograph:
                if (self has moved) "Cosa? Vorresti contraffare una
                                     registrazione archeologica?";
                if (self.photographed_in_situ) "Non di nuovo.";
            ],
        after [;
            Insert:
```

```
                if (second == packing_case) {
                    score = score + self.cultural_value;
                    if (score == MAX_SCORE) deadflag = 2;
                    "Depositat", (genderandnumber) noun, " al sicuro!";
                    }
            Photograph:
                self.photographed_in_situ = true;
            ],
        cultural_value 5,
        photographed_in_situ false;
```

In questo modo, possiamo ridefinire l'oggetto statuette come segue:

```
Treasure -> statuette "statuetta pigmea"
  with  name 'serpente' 'maya' 'pigmea' 'spirito' 'preziosa' 'statuetta'
             'statua',
        initial
            "C'@`e una preziosa statuetta maya qui!",
        description
            "@`E una statuetta minacciosa di uno spirito pigmeo di
             aspetto quasi grottesco. Ha un serpente intorno al collo.",
  has   female;
```

Dal momento che appartiene alla classe Treasure, ne condivide automaticamente tutto il codice in essa contenuto, e lo stesso vale per l'oggetto mask: un bel risparmio, non vi pare?

Altre novità arrivano dall'oggetto stone_table (il ripiano dell'altare), così definito:

```
Object  -> stone_table "ripiano dell'altare"
  with  name 'pietra' 'tavolo' 'lastra' 'altare' 'grande',
        initial
            "Una grande lastra di pietra che funge da tavolo o altare,
             domina il Tempio.",
  has   enterable supporter static;
```

L'attributo enterable si usa per quei contenitori in cui è possibile entrare e, ovviamente, uscire (se sono aperti). Qui però abbiamo anche l'attributo supporter, che fa in modo che il giocatore possa salire sopra l'altare senza però potervi entrare (azione che in questo caso non avrebbe alcun senso):

Il Tempio
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

Sopra il ripiano dell'altare vedi una maschera facciale in mosaico di giada.

>sali sull'altare
Non penso si possa raggiungere qualcosa da qui.

>entra nell'altare
Ti trovi sopra il ripiano dell'altare.

Una maschera facciale in mosaico di giada è appoggiata sull'altare.

>

🔴 Come potete vedere però, l'azione sali non funziona, perché essa corrisponde al verbo inglese climb che non viene a sua volta "attivato" dall'attributo enterable:

```
Object  -> stone_table "ripiano dell'altare"
  with  name 'pietra' 'tavolo' 'lastra' 'altare' 'grande',
        initial
            "Una grande lastra di pietra che funge da tavolo o altare,
             domina il Tempio.",
        before [;
            climb: <<Enter self>>;
            ],
  has   enterable supporter static;
```

Per permettere l'utilizzo di questo comando, dobbiamo ricorrere a una forzatura estrema: ogni volta che il giocatore digita SALI SULL'ALTARE Inform esegue il comando ENTRA NELL'ALTARE, ottenendo così il risultato da noi voluto:

Il Tempio
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

Sopra il ripiano dell'altare vedi una maschera facciale in mosaico di giada.

>sali sull'altare
Ti trovi sopra il ripiano dell'altare.

Una maschera facciale in mosaico di giada è appoggiata sull'altare.

>g

Il Tempio (sopra il ripiano dell'altare)
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

La lampada al sodio, posata a terra, si sta lentamente consumando.

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una maschera facciale in mosaico di giada è appoggiata sull'altare.

>

▲ Prima di concludere il paragrafo, proviamo ancora a vedere com'è possibile creare una porta fittizia:

```
Object  -> StoneDoor "porta di pietra"
  with  name 'porta' 'massiccia' 'grande' 'pietra' 'gialla',
        initial
            "Il passaggio @`e bloccato da una massiccia porta di pietra
             gialla.",
        description
            "@`E solo una grossa porta di pietra.",
        before [;
            Enter: self.vanish();
                   <<Go s_obj>>;
            ],
        react_before [;
            Go: if (noun == s_obj) {
                self.vanish();
                <<Go s_obj>>;
                }
            ],
        vanish [;
            remove self;
            print "Non appena la tocchi, la porta svanisce.^";
            ],
  has   female locked lockable openable;
```

Questo genere di "trucchi" è molto apprezzato in questo tipo di avventure. In poche parole, ecco come si presenta la nuova StoneDoor:

Corridoio in pendenza
Un corridoio basso e squadrato va da nord verso sud, inclinandosi verso la fine.

C'è una preziosa statuetta maya qui!

Il passaggio è bloccato da una massiccia porta di pietra gialla.

>apri la porta
Sembra essere chiusa a chiave.

>apri la porta con la chiave
Non sembra entrare nella serratura.

>spingi la lampada a sud
Non appena la tocchi, la porta svanisce.

Il Tempio
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

Sopra il ripiano dell'altare vedi una maschera facciale in mosaico di giada.

>spingi la lampada a nord

Corridoio in pendenza

C'è una preziosa statuetta maya qui!

>

Apparentemente, sembra essere una vera e propria porta, mentre in realtà è solo un'illusione ottica: basta passarci attraverso (spingendo la lampada a sud o andando verso sud) che questa scompare. Occorre infine dare un'occhiata alle direzioni sud dell'oggetto Corridor e nord dell'oggetto Shrine:

```
Object  Corridor "Corridoio in pendenza"
  with  description
            "Un corridoio basso e squadrato va da nord verso sud,
             inclinandosi verso la fine.",
        n_to Square_Chamber,
        s_to Shrine;

e

Object  Shrine "Il Tempio"
  with  description
            "Questo magnifico tempio mostra segni di scavi da
             preesistenti miniere di calcare, specialmente verso il lato
             occidentale, dove due lunghi cornicioni si dirigono verso
             sud.",
        n_to Corridor,
```

in precedenza, entrambe le direzioni puntavano all'oggetto StoneDoor, che era di fatto una vera e propria porta; ora, invece, StoneDoor è un oggetto qualsiasi che non può più fare da tramite per queste due stanze. Ecco quindi, il motivo per cui è necessario cambiarle.

Osservate, spaventati a morte, il sacerdote mummificato: il suo corpo è disidratato, ed è tenuto insieme solo grazie alla sua forza di volontà. Provate a dirigervi verso sudovest ma la crepa, verso cui si dirigono i cornicioni, è ostruita dai ghiaccioli; il ghiaccio non riesce comunque a nascondere completamente il simbolo di una mezzaluna e vi chiedete quale possa essere il suo significato. Vi scervellate come non mai, pensando a uno scarabocchio di qualche bambino Maya o all'opera di due innamorati in vena di romanticismo, ma è ancora una volta il vostro immenso "acume archeologico" che vi toglie le castagne dal fuoco: avete con voi un dizionario Maya, perché non consultarlo?
Ed ecco svelato il mistero: quel simbolo si pronuncia "xibalba", anche se il suo significato è sconosciuto. Bella scoperta, e adesso? Nonostante la paura, provate a chiedere aiuto al sacerdote ed egli, con grande stupore da parte vostra, indica con le dita ossute i ghiaccioli, che si sciolgono come neve al sole quando parla: "Xibalbá, l'Oltretomba." è quello che riesce a dirvi, prima di tossire e di cadere quasi a pezzi.

§3.9 Il ritorno della mummia (non è un film, è tutto vero)

Quando si gioca a un'avventura testuale, è logico aspettarsi di dovere prima o poi interagire con qualche personaggio. Nel nostro caso, il personaggio è il sacerdote mummificato:

```
Object  priest "sacerdote mummificato"
  with  name 'mummificato' 'sacerdote',
        initial
            "Dietro la lastra, un sacerdote mummificato attende in
             piedi.",
        description
            "Il suo corpo @`e disidratato, ed @`e tenuto insieme solo
             grazie alla sua forza di volont@`a. Sebbene la sua lingua
             sia il Maya locale, hai la curiosa impressione che
             comprender@`a le tue parole.",
        life [;
            Answer:
                "Il sacerdote tossisce e cade quasi a pezzi.";
            Ask: switch (second) {
                'dizionario', 'libro':
                    if (dictionary.correct == false)
                        "~Il simbolo dell'uccello... molto divertente.~";
                    "~Un dizionario? Davvero?~";
                'simbolo', 'simboli', 'maya', 'dialetto':
                    "~Nella nostra cultura, i sacerdoti sanno sempre
                     leggere e scrivere.~";
                'sovrano', 'tomba', 'tempio':
                    "~Questa @`e una faccenda privata.~";
                'pitture':
                    "Il sacerdote si acciglia: ~10 baktun, 4 katun,
                     che sono 1,468,800 giorni dall'inizio del tempo:
                     nel tuo calendario il 19 gennaio 909.~";
                'rovine':
                    "~Le rovine sono sempre state difese dai ladri.
                     Nell'Oltretomba, i saccheggiatori sono stati
                     torturati per l'eternit@`a.~ Una pausa. ~Cos@`i come
                     gli archeologi.~";
                'ragnatele', 'Tana', 'tana', 'Verme':
                    "~Nessun uomo pu@`o oltrepassare la Tana Del
                     Verme.~";
```

```
                'xibalba':
                    if (Shrine.sw_to == Junction)
                        "Il sacerdote scuote le dita ossute.";
                    Shrine.sw_to = Junction;
                    "Il sacerdote indica con le dita ossute i ghiaccioli,
                     che si sciolgono come neve al sole quando parla.
                     ~Xibalb@'a, l'Oltretomba.~";
                    }
                "~Devi trovare la risposta da solo.~";
            Tell:
                "Il sacerdote non @`e interessato alla tua squallida
                 vita.";
            Attack, Kiss:
                remove self;
                "Il sacerdote si disidrata riducendosi in polvere
                 finch@'e non rimane pi@`u nulla.";
            ThrowAt:
                move noun to location; <<Attack self>>;
            Show, Give:
                if (noun == dictionary && dictionary.correct == false) {
                    dictionary.correct = true;
                    "Il sacerdote legge un pezzo del libro, sogghignando
                     sinistramente. Incapace di nascondere il suo
                     divertimento, scarabocchia qualche correzione prima
                     di restituirtelo.";
                    }
                if (noun == newspaper)
                    "Guarda la data. ~12 baktun 16 katun 4 tun 1 uinal
                     12 kin~, dichiara, prima di dare un'occhiata alla
                     prima pagina. ~Ah, vedo che continua.~";
                "Il sacerdote non @`e interessato alle cose terrene.";
            ],
        orders [;
            Go:
                "~Non devo lasciare il Tempio.~";
            NotUnderstood:
                "~Parli per enigmi.~";
            default:
                "~Non sono ai tuoi ordini~";
            ],
  has   animate;
```

Sulla base di quanto è stato detto finora, è facile dedurre che gli oggetti di questo tipo hanno l'attributo animate e che la loro "vita" (il loro grado cioè di interazione con il giocatore) è descritto all'interno della proprietà life:

>parla con il sacerdote
Il sacerdote non è interessato alla tua squallida vita.

>mostra al sacerdote il giornale
Guarda la data. "12 baktun 16 katun 4 tun 1 uinal 12 kin", dichiara, prima di dare un'occhiata alla prima pagina. "Ah, vedo che continua."

>mostra al sacerdote la macchina
Il sacerdote non è interessato alle cose terrene.

>mostra al sacerdote il dizionario

Il sacerdote legge un pezzo del libro, sogghignando sinistramente. Incapace di nascondere il suo divertimento, scarabocchia qualche correzione prima di restituirtelo.

>chiedi al sacerdote delle pitture
Il sacerdote si acciglia: "10 baktun, 4 katun, che sono 1,468,800 giorni dall'inizio del tempo: nel tuo calendario il 19 gennaio 909."

>chiedi al sacerdote di xibalba
Il sacerdote indica con le dita ossute i ghiaccioli, che si sciolgono come neve al sole quando parla. "Xibalbá, l'Oltretomba."

>sacerdote, vai a nord
"Non devo lasciare il Tempio."

Questi sono solo alcuni esempi delle possibili risposte che il sacerdote può dare[18]. Dobbiamo però fare attenzione a due azioni particolari: la prima (la correzione del dizionario) la vedremo meglio nel prossimo paragrafo, mentre la seconda (lo scioglimento dei ghiaccioli) è fondamentale per il proseguimento del gioco.

Quando il giocatore entra per la prima volta nella sala del Tempio, ecco cosa accade se cerca di dirigersi verso sudovest:

>so
I cornicioni si vanno restringendo in una crepa che proseguirebbe se non fosse ostruita dai ghiaccioli. Il ghiaccio non riesce a nascondere completamente il simbolo di una mezzaluna.

Il passaggio è quindi bloccato dai ghiaccioli; tuttavia, è possibile scorgere il simbolo di una mezzaluna. Se il giocatore cerca di rifarsi ancora una volta all'aiuto dello strano sacerdote:

>chiedi al sacerdote della mezzaluna
"Devi trovare la risposta da solo."

deve arrangiarsi da solo. A questo punto, ecco che entra in "azione" il dizionario Maya:

>cerca mezzaluna nel dizionario
Mezzaluna: credo che si pronunci "xibalba", sebbene il suo significato sia sconosciuto.

e il mistero è risolto: quando si chiede di "xibalba" al sacerdote, se la direzione sudovest del Tempio (Shrine.sw_to) è bloccata, la nuova direzione diventa Junction (Shrine.sw_to = Junction;). Così facendo, è finalmente possibile accedere all'incrocio di Xibalbá:

>chiedi al sacerdote di xibalba
Il sacerdote indica con le dita ossute i ghiaccioli, che si sciolgono come neve al sole quando parla. "Xibalbá, l'Oltretomba."

>so

Xibalbá
Sei a cinquanta metri sotto la foresta pluviale e lo scroscio dell'acqua si sente ovunque: queste profonde ed erose miniere di calcare si estendono come radici. Verso nordest un percorso scivoloso, formato da un'ampia colonna di roccia ricoperta dal ghiaccio, ti riporta al Tempio, mentre un canyon prosegue verso l'alto a nord e verso il basso a sud. Alla luce diffusa della lampada al sodio proveniente da sopra, sembra essere bianco come i denti di uno squalo.

>

[18] La libreria wtalk_it.h, scritta da Paolo Lucchesi, permette un sistema di conversazione con dialoghi a menu (in status line o finestra). Ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 4.9.

Leggendo la descrizione di questa nuova locazione, viene quasi istintivo chiedersi come far sì che la lampada non possa in qualche modo "seguire" il nostro archeologo verso sudovest. È presto detto: basta aggiungere le seguenti righe di codice all'azione PushDir dell'oggetto sodium_lamp:

```
Object  sodium_lamp "lampada al sodio"
      .
      .
      .
before [;
      .
      .
      .
    PushDir:
        if (location == Shrine && second == sw_obj)
            "La cosa migliore che puoi fare @`e spingere la lampada
             proprio verso il margine del Tempio, dove il pavimento della
             caverna si ritira.";
        AllowPushDir();
        rtrue;
    ],
      .
      .
      .
  has   female switchable;
```

Se il giocatore si trova nella sala del Tempio e dà il comando SPINGI LA LAMPADA A SO, Inform stampa a video il relativo messaggio, altrimenti viene eseguito il comando e la lampada si sposta insieme al giocatore.

* *

Passiamo ora a esaminare il dizionario Maya (l'oggetto dictionary):

```
Object  dictionary "dizionario maya di Waldeck"
  with  name 'dizionario' 'locale' 'guida' 'libro' 'maya' 'waldeck',
        description
            "Compilata dall'inaffidabile litografia del leggendario
             narratore ed esploratore ~Conte~ Jean Frederic Maximilien
             Waldeck (1766??-1875), questa guida contiene quel poco che
             si conosce sui simboli usati nell'antico dialetto locale.",
        before [ w1 w2 glyph;
            Consult:
                wn = consult_from;
                w1 = NextWord();
                w2 = NextWord();
                if (consult_words ==1 && w1~='simbolo' or 'simboli')
                    glyph = w1;
                else if (consult_words ==2 && w1=='simbolo') glyph = w2;
                else if (consult_words ==2 && w2~='simbolo') glyph = w2;
                else "Prova ~cerca <nome del simbolo> sul dizionario~.";
                switch (glyph) {
                    'q1':        "(Questo @`e uno dei simboli che hai
                                  annotato!)^^
                                  Q1: ~luogo sacro~.";
```

```
                'mezzaluna': "Mezzaluna: credo che si pronunci
                              ~xibalba~, sebbene il suo significato
                              sia sconosciuto.";
                'freccia':    "Freccia: ~viaggio; divenire~.";
                'teschio':    "Teschio: ~morte, destino funesto; fato
                               (non nec. brutto)~.";
                'cerchio':   "Cerchio: ~il Sole; anche la vita,
                              l'arco della vita~.";
                'giaguaro':  "Giaguaro: ~sovrano~.";
                'scimmia':   "Scimmia: ~sacerdote?~.";
                'uccello':   if (self.correct) "Uccello: ~morto
                             stecchito~.";
                             "Uccello: ~ricco, benestante?~.";
                default:     "@`E un simbolo che non @`e ancora stato
                              annotato.";
            }
        ],
    correct false;
```

Questo oggetto, in effetti, è piuttosto complicato, ma proviamo comunque a dargli ugualmente un'occhiata: la variabile di libreria wn contiene, grazie all'istruzione consult_from, la posizione del simbolo all'interno della frase (o, come dice più "tecnicamente" Giancarlo Niccolai[19], il numero delle parole riconosciute come topic). Nella definizione in Infit del verbo cerca, la parola-chiave topic (che in Inform si riferisce a qualsiasi testo digitato dal giocatore) occupa la seconda posizione e, con un comando del tipo CERCA FRECCIA NEL DIZIONARIO, abbiamo che:

1. la variabile di libreria wn vale 2 (il testo topic freccia occupa la seconda posizione nel comando digitato dal giocatore;
2. Inform mette a disposizione del programmatore le variabili locali w1 e w2.

L'istruzione NextWord, poi, "preleva" dal comando la parola che occupa la posizione wn e l'assegna alla variabile locale w1:

```
>cerca freccia nel dizionario
Freccia: "viaggio; divenire".
```

mentre la variabile locale w2, in questo caso, è vuota.
L'istruzione consult_words, si occupa invece del numero delle parole digitate dal giocatore nel topic:

```
>cerca la freccia nel dizionario
Freccia: "viaggio; divenire".

>cerca simbolo freccia nel dizionario
Freccia: "viaggio; divenire".
```

Nel primo esempio, abbiamo nel topic le parole la (prelevata dall'istruzione NextWord e memorizzata in w1) e freccia (prelevata dall'istruzione NextWord e memorizzata in w2); consult_words vale allora 2 (nell'esempio precedente valeva invece 1, perché il giocatore aveva digitato solo "freccia"). Se w1 è uguale a "simbolo" (è il caso del secondo esempio), allora il simbolo da cercare sul dizionario è quello contenuto in w2, altrimenti (è il caso del primo esempio) è sempre quello contenuto in w2.

[19] Giancarlo Niccolai è l'autore della stratosferica "Warmage", e mi piace definirlo come "un marziano travestito da terrestre". È in grado di fare, con Inform, praticamente tutto (o quasi) quello che vi può passare per la mente. L'indirizzo della pagina web della sua avventura è: http://pitermos.niccolai.cc/diven.php.

🔴 Concludiamo questo paragrafo con le definizioni dei verbi. Abbiamo visto nel paragrafo 3.7, la definizione del verbo fotografa:

```
Verb 'fotografa' * noun                                 -> Photograph;
```

dove noun si riferisce a qualsiasi oggetto definito all'interno dell'avventura stessa, che deve però essere visibile dal giocatore (in poche parole, non posso fotografare una statuetta se questa non è presente nella locazione in cui mi trovo). Ma, come è logico che sia, esistono diversi altri modi (alcuni più semplici e altri più complessi) per definire un nuovo verbo (e di conseguenza una nuova azione). Vediamo allora alcuni esempi presi dal file ItalianG.h:

```
Verb 'salta'     *                                      -> Jump;
```

Beh, questo è elementare: basta digitare SALTA per far sì che l'azione Jump abbia luogo. Occorre notare che, a differenza dell'azione Photograph, in questo caso non è necessario creare una funzione JumpSub, perché l'azione Jump è una di quelle riconosciute da Inform come standard.
Vediamo invece la definizione del verbo scava:

```
Verb 'scava'     * noun                                 -> Dig
                 * noun 'con'/'col' held                -> Dig;
```

Come potete vedere, È POSSIBILE DEFINIRE, PER UN VERBO, UNA STESSA AZIONE IN MODALITÀ DIFFERENTI. La prima funziona esattamente come l'azione Photograph, mentre la seconda presenta una novità: held. Questa parola-chiave si riferisce agli oggetti posseduti dal giocatore nel suo inventario: in questo caso, è possibile prevedere un comando del tipo SCAVA LA TERRA CON LA PALA, a patto però che il giocatore abbia nel suo inventario la pala medesima.
Il verbo bacia (poteva mai mancare un verbo così?) ci riserva altre novità:

```
Verb 'bacia' 'abbraccia'
                 * creature                             -> Kiss;
```

Quindi, È POSSIBILE DEFINIRE PIÙ VERBI PER UNA STESSA AZIONE. La parola-chiave creature si riferisce a tutti quegli oggetti che hanno l'attributo animate: in Ruins, infatti, posso baciare il sacerdote, ma questi è talmente timido da ridursi poi in polvere.
Invece, la definizione del verbo apri:

```
Verb 'apri' 'scopri'
                 * noun                                 -> Open
                 * noun 'con'/'col'/'a' held            -> Unlock;
```

mostra chiaramente che È POSSIBILE DEFINIRE AZIONI DIVERSE PER UNO O PIÙ VERBI.
Un'altra caratteristica interessante può essere poi notata per il verbo chiedi:

```
Verb 'chiedi' 'domanda'
                 * noun 'a'/'ad'/'all^'/'allo'/'alla'/'al'/'agli'/'ai'/
                   'alle' creature              -> AskFor reverse;
```

In questo caso, reverse inverte di posto noun e creature, dando al giocatore la possibilità di digitare un comando del tipo CHIEDI AL SACERDOTE DELLE ROVINE (e non CHIEDI DELLE ROVINE AL SACERDOTE, come avverrebbe se si lasciasse solo AskFor).
Ecco poi, come si possono inserire delle abbreviazioni:

```
Verb 'esamina' 'x//' 'descrivi'
                  * noun                                  -> Examine;
```

dove digitare ESAMINA LA CASSA e X CASSA equivale alla stessa azione.
Ci sono poi alcuni verbi che, per poter essere utilizzati, richiedono la presenza di almeno due oggetti visibili dal giocatore:

```
Verb 'lega' 'fissa' 'congiungi' 'unisci' 'allaccia' 'annoda'
                * noun                                  -> Tie
                * noun 'a'/'ad'/'all^'/'allo'/'alla'/'al'/'agli'/'ai'/
                  'alle'  noun                          -> Tie
                * noun 'con'/'col' noun                 -> Tie;
```

come accade, ad esempio, se si digita un comando del tipo LEGA LA CORDA AL LAMPADARIO, dove abbiamo appunto due oggetti: la corda (il primo noun) e il lampadario (il secondo noun)[20]. Ricordatevi però, che per testare in Inform un'azione di questo tipo occorre rifarsi al seguente codice:

```
Tie:     if (noun == corda && second == lampadario)...
```

Un'altra esigenza, potrebbe essere legata al fatto che alcuni verbi possono essere utilizzati con più oggetti alla volta:

```
Verb 'prendi' 'trasporta' 'afferra' 'raccogli'
                * multi                                 -> Take
                * multiinside 'da'/'dal'/'dallo'/'dalla'/'dall^'/
                  'dagli'/'dalle'/'dai' noun            -> Remove;

Verb 'metti'
                * held                                  -> Wear
                * 'gi@`u'/'giu' multiheld               -> Drop
                * multiheld 'gi@`u'/'giu'               -> Drop
                * multiexcept 'dentro'/'in'/'nel'/'nello'/'nell^'/'nella'
                  /'negli'/'nelle'/'nei' noun           -> Insert
                * multiexcept 'su'/'sul'/'sullo'/'sull^'/'sulla'/'sui'/
                  'sugli'/'sulle'/'sopra' noun          -> PutOn;
```

In questi esempi, il giocatore ha la possibilità di digitare dei comandi del tipo PRENDI IL FUNGO E LA MAPPA (multi), oppure PRENDI IL GIORNALE E LA MACCHINA DALLA CASSA (multiinside → agisce su più oggetti solo se si trovano all'interno di un contenitore), o ancora METTI GIÙ LA MASCHERA, LA MAPPA E LA MACCHINA (multiheld → agisce su più oggetti solo se si trovano nell'inventario del giocatore) e METTI LA MASCHERA E LA MAPPA SULL'ALTARE (multiexcept → agisce su più oggetti ad eccezione dell'altare stesso). Ovviamente, si possono anche usare forme del tipo PRENDI TUTTO, PRENDI TUTTE LE MONETE, LASCIA TUTTO, ecc.
Se invece dobbiamo rifarci a una direzione ben precisa, diamo allora un'occhiata alla definizione del verbo vai:

```
Verb 'vai' 'cammina' 'corri' 'va'
                * 'a'/'ad'/'verso' noun=Adirection       -> Go
```

dove noun è uguale a una direzione qualsiasi tra quelle permesse da Inform (Adirection). Se invece dobbiamo toglierci un vestito:

```
Verb 'rimuovi' 'togli'
                 * worn                                 -> Disrobe
```

[20] Attenzione: non si fa riferimento alle parole CORDA e LAMPADARIO, ma a due oggetti denominati corda e lampadario (oppure a due oggetti con dei nomi qualsiasi, ma con sinonimi relativi rispettivamente a corda per il primo e lampadario per il secondo, presenti nelle with name... dei rispettivi oggetti di appartenenza).

ecco la parola-chiave da usare (worn).
Può anche verificarsi il caso in cui si abbia la necessità di ampliare, in qualche modo, l'utilizzo di un verbo già esistente (o meglio, già dichiarato nel file ItalianG.h). In questo caso si deve ricorrere all'istruzione Extend, che possiamo applicare, ad esempio, al verbo cerca, così definito di default:

```
Verb 'cerca' 'trova' 'ricerca' 'fruga'
!               * noun                              -> Search
                * 'dentro'/'in'/'nel'/'nello'/'nell^'/'nella'/
                  'negli'/'nelle'/'nei'/'fra'/'tra' noun
                                                    -> Search
                * topic 'dentro'/'in'/'nel'/'nello'/'nell^'/'nella'/
                  'negli'/'nelle'/'nei'/'fra'/'tra' noun
                                                    -> Consult reverse
                * 'dentro'/'in'/'nel'/'nello'/'nell^'/'nella'/
                  'negli'/'nelle'/'nei'/'fra'/'tra' noun topic
                                                    -> Consult;
```

Facendo riferimento al dizionario Maya, con questo verbo è possibile dare il comando CERCA [NOME_SIMBOLO] NEL DIZIONARIO, ma non CERCA [NOME_SIMBOLO] SUL DIZIONARIO. Ecco allora come ampliare la definizione di questo verbo:

```
Extend 'cerca' * topic 'su'/'sul'/'sui'/'sulle'/'sullo'/'sull^'/
                       'sulla'/'sugli' noun              -> Consult;
```

La parola-chiave topic, come abbiamo già detto, si riferisce a un qualsiasi testo digitato dall'utente. Volendo essere estremamente pignoli, possiamo anche usare l'istruzione Extend only, utile nel caso in cui si voglia estendere il significato di un solo verbo, appartenente però a un'azione che ne prevede più di uno:

```
Extend only 'cerca' * topic 'su'/'sul'/'sui'/'sulle'/'sullo'/'sull^'/
                        'sulla'/'sugli' noun              -> Consult;
```

Così facendo, la seguente regola grammaticale vale solo per il verbo cerca e non per i verbi trova, ricerca e fruga. Nonostante però la definizione estesa del verbo sia giusta, il comando sembra non voler ancora funzionare. Ecco infatti quello che succede:

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>cerca freccia sul dizionario
Dentro cosa vuoi cercare?

Questo accade perché, per come è definito questo verbo di default, dobbiamo ricorrere alla parola-chiave first[21]:

[21] Questo è ovviamente un caso a sé. Nella maggior parte dei casi, infatti, per estendere il significato di un verbo basta ricorrere alle sole espressioni Extend 'nome_verbo' * ... e Extend only 'nome_verbo' * ...

```
Extend 'cerca' first * topic 'su'/'sul'/'sui'/'sulle'/'sullo'/'sull^'/
                        'sulla'/'sugli' noun              -> Consult;
```

che si usa quando abbiamo la necessità di stabilire quale riga della grammatica di un verbo debba essere eseguita per prima. Se invece deve essere eseguita per ultima, si può ricorrere all'utilizzo della parola-chiave last:

```
Extend     "spingi"   last
                * noun 'verso'/'vicino'/'sotto' noun        -> PushDir;
```

Se poi si vuole sostituire un verbo già esistente, si può usare la parola-chiave replace:

```
Extend "leggi"        replace
                      *                                     -> Leggi
                      * noun                                -> Leggi;
```

con la quale si sostituisce, in questo esempio, il verbo leggi.
La parola-chiave meta è riservata invece solo ai comandi per il gioco (come ad esempio CARICA, SALVA, FINE, ecc.) e non viene usata quasi mai.
Infine è possibile, come dice Francesco Cordella, aggiungere un sinonimo a un verbo già esistente tramite l'operatore di assegnamento =, come accade nel seguente esempio:

```
Verb 'perquisisci' = 'cerca';
```

dove perquisisci è il sinonimo in questione.

Vi dirigete verso sudovest e arrivate, finalmente, all'incrocio di Xibalbá. Lo scroscio dell'acqua si sente ovunque e una piccola stele giace su una sporgenza ad altezza d'uomo. A nordest, un percorso scivoloso formato da un'ampia colonna di roccia ricoperta dal ghiaccio vi riporta al Tempio, mentre un canyon prosegue verso l'alto a nord e verso il basso a sud. Percorrete il canyon verso il basso, che però si arresta improvvisamente a un baratro oscuro e vertiginoso. Niente può essere visto o sentito da lì sotto, ma il vero problema è che non c'è nessuna cabina telefonica nelle vicinanze, e non potete così trasformarvi in Superman. Di saltarlo non se ne parla, perché è davvero troppo largo, e siete costretti a tornare indietro. Ripercorrete il canyon nella direzione opposta, ma la più alta e vasta estremità a nord sale soltanto verso una parete irregolare di roccia calcarea vulcanica. Qui, l'unica cosa che potete notare, è la presenza di un'enorme sfera di pietra pomice larga circa due metri e mezzo, su cui vi appoggiate con una mano, incerti sul da farsi.

§3.10 L'incrocio di Xibalbá (ma è davvero così pauroso?)

🔴 Prima di procedere con la nostra avventura, forse è bene specificare che, come in tutte le avventure testuali che si rispettino, anche con quelle scritte in Inform è possibile salvare lo stato del gioco (tramite il comando SALVA) per poi ripristinarlo in seguito (tramite il comando CARICA).
Ma ci sono anche altri comandi su cui è bene spendere due parole[22]:

- ANCORA: ripete il comando digitato in precedenza;
- FINE/BASTA/USCIRE: come si può facilmente intuire, digitando a scelta uno di questi comandi l'avventura termina; prima però, Inform chiede se si è veramente sicuri di volerlo, e bisogna allora rispondere con un SÌ (o S) o con un NO (o N);
- MODALITÀ BREVE/NORMALE/LUNGA: questo comando attiva, in qualsiasi momento, una delle tre modalità spiegate nel paragrafo 3.3. Di default, è attivata quella normale (descrizioni lunghe per i luoghi mai visitati prima e brevi se già visitati);
- NOTIFY ON/OFF: viene attivata (NOTIFY ON) o disattivata (NOTIFY OFF) la notifica del punteggio. In poche parole, questo comando fa in modo che Inform, nella modalità completa del punteggio, visualizzi o meno il messaggio standard [Il tuo punteggio è appena aumentato di |n| punti.][23];
- OGGETTI: vengono stampati a video i nomi di tutti gli oggetti posseduti nel corso del gioco;
- POSTI: vengono stampati a video i nomi di tutte le locazioni visitate fino al momento dell'immissione del comando stesso;
- PRONOMI: vengono stampate a video delle informazioni relative ai pronomi (attivati o disattivati) usati da Inform nella stanza in cui il giocatore si trova al momento in cui viene dato il comando;
- RECORDING ON/OFF: viene attivata (RECORDING ON) o disattivata (RECORDING OFF) la trascrizione dei comandi inseriti in un file Script (un file di testo con l'estensione .rec);
- REPLAY: apre il file con l'estensione .rec; prima di eseguire i comandi in esso contenuti, Inform chiede al giocatore se vuole o meno una pausa a fine pagina durante lo scorrimento del testo (Do you want MORE prompts? (y/n)) a cui deve seguire – se è attiva quest'ultima – la pressione di un tasto qualsiasi da parte dell'utente;
- RICOMINCIA/RICOMINCIARE: se per un qualsiasi motivo desiderate ricominciare dall'inizio la vostra avventura, ecco quello che fa al caso vostro;
- TRANSCRIPT ON/OFF: viene attivata (TRANSCRIPT ON) o disattivata (TRANSCRIPT OFF) la trascrizione del testo del gioco in un file Script (un file di testo con l'estensione .log apribile tranquillamente da qualsiasi editor di testo);
- VERIFICA/VERIFICARE: con questo comando è possibile verificare l'integrità o meno del file di gioco;

[22] La libreria command_it.h, da me scritta, mostra un menu di aiuto per i comandi. Ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 4.9.
[23] A livello di programmazione, è possibile controllare questo evento tramite la variabile di libreria notify_mode; se essa vale true → notify_mode = true; la notifica del punteggio è attiva, se vale false → notify_mode = false; la notifica del punteggio non è attiva (di default, notify_mode vale true).

- VERSIONE: vengono stampate a video delle informazioni relative alle versioni delle librerie di sistema, dell'interprete standard e di Infit;

Torniamo adesso a Ruins: se esaminiamo la stele, ecco il messaggio che viene stampato a video:

Xibalbá
Sei a cinquanta metri sotto la foresta pluviale e lo scroscio dell'acqua si sente ovunque: queste profonde ed erose miniere di calcare si estendono come radici. Verso nordest un percorso scivoloso, formato da un'ampia colonna di roccia ricoperta dal ghiaccio, ti riporta al Tempio, mentre un canyon prosegue verso l'alto a nord e verso il basso a sud. Alla luce diffusa della lampada al sodio proveniente da sopra, sembra essere bianco come i denti di uno squalo.

Una piccola stele, o forse una pietra di confine, giace su una sporgenza ad altezza d'uomo.

>esamina la stele
Le iscrizioni sembrano avvertirti che il confine di Xibalbá, il Luogo Della Paura, è vicino. Il simbolo di un uccello è predominante.

>

Ora siamo finalmente in grado di esaminare la prima azione del sacerdote, lasciata in sospeso nel paragrafo precedente (la correzione del dizionario Maya):

```
Show, Give:
        if (noun == dictionary && dictionary.correct == false) {
            dictionary.correct = true;
            "Il sacerdote legge un pezzo del libro, sogghignando
             sinistramente. Incapace di nascondere il suo divertimento,
             scarabocchia qualche correzione prima di restituirtelo.";
        }
```

Quando il giocatore dà il dizionario al sacerdote questi, con un sorriso ironico, assegna il valore true alla variabile locale correct dell'oggetto dictionary. In questo modo, quando gli si chiede del dizionario:

```
Ask: switch (second) {
         'dizionario', 'libro':
            if (dictionary.correct == false)
                "~Il simbolo dell'uccello... molto divertente.~";
            "~Un dizionario? Davvero?~";
```

il messaggio stampato a video non è più "Il simbolo dell'uccello... molto divertente." ma "Un dizionario? Davvero?". Ecco dimostrata, ancora una volta, l'estrema flessibilità di questo straordinario linguaggio di programmazione, che offre la possibilità di verificare "dall'esterno" il valore di una variabile locale appartenente a un dato oggetto (notate bene: locale, quindi visibile in teoria solo all'interno dell'oggetto di appartenenza).

Focalizziamo ora la nostra attenzione sulla descrizione di Xibalbá: a un certo punto, più precisamente verso la fine, essa dice: "Alla luce diffusa della lampada al sodio proveniente da sopra, sembra essere bianco come i denti di uno squalo.". Ma è davvero così?
Facciamo una prova: dirigiamoci verso nordest, spegniamo la lampada e ritorniamo al punto di partenza:

Il Tempio
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

La lampada al sodio, posata a terra, si sta lentamente consumando.

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

>spegni la lampada
Hai spento la lampada al sodio.

È completamente buio qui!

>g

Oscurità
L'oscurità intorno a te è opprimente e ti senti quasi soffocare.

Da qualche parte, dei piccoli artigli si stanno muovendo velocemente.

>so

Xibalbá
Sei a cinquanta metri sotto la foresta pluviale e lo scroscio dell'acqua si sente ovunque: queste profonde ed erose miniere di calcare si estendono come radici. Verso nordest un percorso scivoloso, formato da un'ampia colonna di roccia ricoperta dal ghiaccio, ti riporta al Tempio, mentre un canyon prosegue verso l'alto a nord e verso il basso a sud. Alla luce diffusa della lampada al sodio proveniente da sopra, sembra essere bianco come i denti di uno squalo.

Una piccola stele, o forse una pietra di confine, giace su una sporgenza ad altezza d'uomo.

>

Come potete vedere, la piccola stele continua a essere visibile insieme a tutto il resto, nonostante la lampada al sodio sia spenta. Senza contare che c'è un ulteriore problema: anche le locazioni adiacenti all'incrocio di Xibalbá (rispettivamente l'estremità superiore e l'estremità inferiore del canyon) sono visibili (mentre dovrebbero essere anch'esse al buio poiché situate al di sotto della sala del Tempio). Paolo Maroncelli, altro autore di avventure testuali, consiglia in questo caso di utilizzare l'istruzione objectloop nel seguente modo:

1. Per prima cosa bisogna definire una nuova classe denominata Room:

```
.
.
.
Include "VerbLib";
Include "replace";

Class Room
  has;
```

2. Occorre poi "marcare" le stanze interessate (Junction, Canyon_N e Canyon_S) con la classe medesima:

```
Room    Canyon_N "Estremit@`a superiore del canyon"
   .
   .
   .
   d_to Junction;
```

```
Room    Junction "Xibalb@'a"
   .
   .
   .
   d_to Canyon_S;

Room    Canyon_S "Estremit@`a inferiore del canyon"
   .
   .
   .
   d_to nothing;
```

3. E definire, infine, le due funzioni light_room e nolight_room che potete posizionare, ad esempio, subito sotto la funzione PrintRank:

```
[light_room x;
   objectloop(x ofclass Room) give x light;
];

[nolight_room x;
   objectloop(x ofclass Room) give x ~light;
];
```

da richiamare all'occorrenza tramite la proprietà after dell'oggetto sodium_lamp:

```
after [;
     SwitchOn:
         give self light;
         light_room();
     SwitchOff:
         give self ~light;
         nolight_room();
     ],
```

I giochi sono fatti. L'istruzione objectloop testa tutte locazioni appartenenti alla classe Room e, se la lampada al sodio è spenta, assegna loro l'attributo ~light, mentre se è accesa assegna loro l'attributo light.

Per quanto riguarda invece l'oggetto chasm (il baratro):

```
Object  -> chasm "baratro pauroso"
  with  name 'oscuro' 'baratro' 'buca' 'pauroso' 'profondo',
        before [;
            Enter:
                deadflag = 3;
                "Precipiti attraverso il vuoto silenzio dell'oscurit@`a,
                 sbattendo la testa contro una sporgenza della roccia.
                 Ferito e dolorante, non puoi fare altro che raccomandare
                 a Dio la tua anima...";
            JumpOver:
                "@`E davvero troppo largo.";
            ],
        after [;
            Receive:
                remove noun;
                print_ret (The) noun,
                          " cade silenziosamente nell'oscurit@`a del
```

```
                        baratro.";
            Search:
                "Il baratro @`e profondo e fangoso.";
            ],
        react_before [;
            Jump:
                <<Enter self>>;
            Go:
                if (noun == d_obj) <<Enter self>>;
            ],
  has   scenery open container;
```

possiamo notare il terzo e ultimo valore che può assumere la variabile di libreria deadflag. Abbiamo visto che se essa vale 1, il gioco termina con una morte "standard" (Inform stampa cioè, il messaggio "***Sei Morto***"), mentre se vale 2 il giocatore ha terminato l'avventura in maniera positiva (Inform stampa cioè, il messaggio "***Hai vinto***"). E se vale 3 come in questo caso? Beh, abbiamo l'onore di assistere a una morte personalizzata, nel senso che Inform stampa, alla fine della partita, il messaggio da noi voluto anziché quello standard. Per fare questo però, occorre definire la funzione di libreria DeathMessage, posizionata in genere subito sotto la funzione di libreria PrintRank:

```
[ DeathMessage;
        if (deadflag == 3) print "Sei stato inghiottito dalle tenebre";
        ];
```

e se proviamo, magari, a lanciarci nel vuoto:

```
Estremità inferiore del canyon
All'estremità sud, più bassa e stretta, il canyon si arresta ad un baratro oscuro e vertiginoso. Niente può essere visto o sentito da lì sotto.

>giù
Precipiti attraverso il vuoto silenzio dell'oscurità, sbattendo la testa contro una sporgenza della roccia. Ferito e dolorante, non puoi fare altro che raccomandare a Dio la tua anima...


   *** Sei stato inghiottito dalle tenebre ***


In questa partita hai totalizzato 0 punti su 30 possibili, in 27 turni, guadagnando il rango di Turista.

Vuoi RICOMINCIARE, CARICARE una partita salvata o USCIRE ?
>
```

🔴 Per rendere più "realistica" la situazione, possiamo rifarci all'attributo talkable che permette al giocatore di parlare anche a un oggetto privo dell'attributo animate:

```
Object  -> chasm "baratro pauroso"
  with  name 'oscuro' 'baratro' 'buca' 'pauroso' 'profondo',
        before [;
            Enter:
                deadflag = 3;
                "Precipiti attraverso il vuoto silenzio dell'oscurit@`a,
                 sbattendo la testa contro una sporgenza della roccia.
                 Ferito e dolorante, non puoi fare altro che raccomandare
                 a Dio la tua anima...";
            JumpOver:
```

```
                "@`E davvero troppo largo.";
            Tell: "Il rimbombo della tua voce @`e quasi assordante.
                   Questo baratro sembra essere davvero molto profondo.";
            ],
        after [;
            Receive:
                remove noun;
                print_ret (The) noun,
                          " cade silenziosamente nell'oscurit@`a del
                           baratro.";
            Search:
                "Il baratro @`e profondo e fangoso.";
            ],
        react_before [;
            Jump:
                <<Enter self>>;
            Go:
                if (noun == d_obj) <<Enter self>>;
            ],
  has   scenery open container talkable;
```

In questo modo, se viene dato il comando PARLA AL BARATRO:

>parla al baratro
Il rimbombo della tua voce è quasi assordante. Questo baratro sembra essere davvero molto profondo.

ecco che viene stampato a video il relativo messaggio di risposta. Volendo, possiamo estendere il significato del verbo parla:

```
Extend 'parla' * 'nel'/'nello'/'nella' creature            -> Tell;
```

e magari, già che ci siamo, definire anche dei sinonimi:

```
Verb 'urla' 'grida' = 'parla';
```

Così facendo, se ora digitiamo i comandi PARLA NEL BARATRO, URLA NEL BARATRO o GRIDA NEL BARATRO, otteniamo la risposta di prima.

⚠ Vediamo, infine, com'è possibile creare un ponte sospeso:

```
Object Bridge "ponte di legno"
  with description [;
          if (self has crossable) "@`E l'ideale per chi soffre di
                                   vertigini.";
          "Non serve pi@`u a nulla ormai.";
       ],
       name 'fragile' 'pericolante' 'legno' 'ponte' 'passaggio',
       when_open [;
          if (self has crossable) "Un ponte di legno pericolante
                                   attraversa il baratro.";
          "Puoi vedere alcuni resti del ponte crollato qui.";
       ],
       door_to [;
         if (self has crossable) {
          if (children(player) > 1) {
               give self ~crossable;
               print "Dopo appena pochi passi, il ponte comincia a
```

```
                        crollare, ma riesci comunque ad arrivare di
                        corsa dall'altra parte.^";
            }
            else print "Hai attraversato il ponte pericolante.^";
            if (location == Canyon_S) return South_side; return Canyon_S;
          }
          else print "Vuoi forse cadere nel baratro?^";
          rtrue;
        ],
        door_dir [;
           if (location == Canyon_S) return s_to; return n_to;
        ],
        found_in Canyon_S South_side,
has  static door open crossable;

Object South_side "Lato sud del baratro"
  with description
               "Ti trovi sul lato sud del baratro. Delle chiazze di
                muschio, sparse un po' dappertutto, emettono una strana
                luce verde che ti permette di vedere dove ti trovi.",
       n_to Bridge,
  has  light scenery;
```

L'oggetto Bridge, presente nelle locazioni Canyon_S e South_side, l'ho inserito, per comodità, subito sotto l'oggetto chasm. Quello che forse può stupire, è che venga dichiarato come door (come abbiamo visto per gli oggetti steps e Stonedoor), ma in fin dei conti non è forse un ponte un punto di collegamento tra due locazioni?
Comunque, la vera novità di questo oggetto è l'attributo crossable: Inform, infatti, dà anche la possibilità di creare dei nuovi attributi da applicare a qualsiasi oggetto, tramite l'istruzione Attribute:

```
.
.
.
Include "VerbLib";
Include "replace";

Attribute crossable;
```

All'inizio, il ponte è agibile (l'oggetto Bridge ha l'attributo crossable). Però, se il giocatore attraversa il ponte con più di un oggetto nel suo inventario, ecco quello che accade:

Estremità inferiore del canyon
All'estremità sud, più bassa e stretta, il canyon si arresta ad un baratro oscuro e vertiginoso. Niente può essere visto o sentito da lì sotto.

Un ponte di legno pericolante attraversa il baratro.

>i
Stai portando:
  una maschera facciale in mosaico di giada (indossata)
  una macchina fotografica a lastre

>s
Dopo appena pochi passi, il ponte comincia a crollare, ma riesci comunque ad arrivare di corsa dall'altra parte.

Lato sud del baratro

Ti trovi sul lato sud del baratro. Delle chiazze di muschio, sparse un po' dappertutto, emettono una strana luce verde che ti permette di vedere dove ti trovi.

Puoi vedere alcuni resti del ponte crollato qui.
>esamina il ponte
Non serve più a nulla ormai.

>n
Vuoi forse cadere nel baratro?

>

Come potete vedere, il povero archeologo è rimasto intrappolato nel muschio per il resto dei suoi miseri giorni. Se invece attraversiamo il ponte con un solo (o ancora meglio, nessuno) oggetto nell'inventario, è possibile tornare indietro:

Estremità inferiore del canyon
All'estremità sud, più bassa e stretta, il canyon si arresta ad un baratro oscuro e vertiginoso. Niente può essere visto o sentito da lì sotto.

Un ponte di legno pericolante attraversa il baratro.

>posa la macchina
Posata.

>attraversa il ponte
Hai attraversato il ponte pericolante.

Lato sud del baratro
Ti trovi sul lato sud del baratro. Delle chiazze di muschio, sparse un po' dappertutto, emettono una strana luce verde che ti permette di vedere dove ti trovi.

Un ponte di legno pericolante attraversa il baratro.

>esamina il ponte
È l'ideale per chi soffre di vertigini.

>n
Hai attraversato il ponte pericolante.

Estremità inferiore del canyon

Un ponte di legno pericolante attraversa il baratro.

Puoi anche vedere una macchina fotografica a lastre qui.

>g

Estremità inferiore del canyon
All'estremità sud, più bassa e stretta, il canyon si arresta ad un baratro oscuro e vertiginoso. Niente può essere visto o sentito da lì sotto.

Un ponte di legno pericolante attraversa il baratro.

Puoi anche vedere una macchina fotografica a lastre qui.

>

Il verbo attraversa corrisponde all'azione Enter, che viene automaticamente attivata da Inform quando si definisce un oggetto di tipo door. Non deve quindi stupire se è possibile utilizzare questo verbo pur

non avendolo dichiarato da nessuna parte. Per far sì però che il ponte "funzioni", dobbiamo però fare ancora alcune piccole modifiche agli oggetti Canyon_S e chasm:

```
Object  Canyon_S "Estremit@`a inferiore del canyon"
  with  description
            "All'estremit@`a sud, pi@`u bassa e stretta, il canyon si
             arresta ad un baratro oscuro e vertiginoso. Niente pu@`o
             essere visto o sentito da l@`i sotto.",
        n_to Junction,
        u_to Junction,
        s_to Bridge,
        d_to nothing,
  has   light;

Object  chasm "baratro pauroso"
  with  name 'oscuro' 'baratro' 'buca' 'pauroso' 'profondo',
        before [;
            Enter:
                deadflag = 3;
                "Precipiti attraverso il vuoto silenzio dell'oscurit@`a,
                 sbattendo la testa contro una sporgenza della roccia.
                 Ferito e dolorante, non puoi fare altro che raccomandare
                 a Dio la tua anima...";
            JumpOver:
                "@`E davvero troppo largo.";
            ],
        after [;
            Receive:
                remove noun;
                print_ret (The) noun,
                          " cade silenziosamente nell'oscurit@`a del
                           baratro.";
            Search:
                "Il baratro @`e profondo e fangoso.";
            ],
        react_before [;
            Jump:
                <<Enter self>>;
            Go:
                if (noun == d_obj) <<Enter self>>;
            ],
            found_in Canyon_S South_side,
  has   scenery open container;
```

Nell'oggetto Canyon_S la direzione sud punta ora all'oggetto Bridge, mentre chasm è ora visibile nelle locazioni Canyon_S e South_side (esattamente come per l'oggetto Bridge); questo per evitare che il baratro sia visibile da una parte sola e non da entrambe. Dal momento poi, che è ora presente l'istruzione found_in, non è più necessario per quest'ultimo oggetto il simbolo ->.

Non potete stare tutto il tempo a poltrire, perché la prima regola che vi hanno insegnato alla scuola serale di archeologia è stata quella di non perdersi mai d'animo. Il vostro fiuto "infallibile" vi dice che dovete dirigervi verso il Tempio, ma pochi attimi dopo aver tolto la mano dalla sfera di pietra pomice, essa comincia rotolare a sud, diretta verso il baratro oscuro e profondo. La rincorrete più veloce che potete, ma non potete fare nulla per impedire che la sfera rotoli giù nel canyon senza controllo per alcuni metri, prima di sussultare nelle fauci del baratro. Con un piccolo rimbalzo, però, vi colpisce di striscio sulla fronte. Perdete i sensi e cadete in avanti sanguinanti; quando riprendete conoscenza, l'estremità sud del canyon continua sopra la sfera di pietra pomice incastrata nel baratro e vi compiacete nuovamente con voi stessi per il vostro incredibile "acume" archeologico. Un osso inciso, più luminoso di quello che sembra, sporge da una polla di limo bagnato nella parete del canyon. Lo raccogliete chiedendovi se piacerà al vostro cane e, dal momento che non potete più proseguire, tornate indietro in direzione della Sala del Tempio.

§3.11 La sfera di pietra pomice (avete indossato il casco?)

Sulla base di quanto abbiamo visto finora, il nuovo codice di Ruins non presenta delle grosse novità, a parte la funzione each_turn dell'oggetto chasm:

```
Object  -> chasm "baratro pauroso"
  with  name 'oscuro' 'baratro' 'buca' 'pauroso' 'profondo',
        .
        .
        .
        each_turn [;
            if (huge_ball in parent(self)) {
                remove huge_ball;
                Canyon_S.s_to = On_Ball;
                Canyon_S.description =
                    "L'estremit@`a sud del canyon continua ora sopra la
                     sfera di pietra pomice incastrata nel baratro.";
                "^La sfera di pietra pomice rotola gi@`u nel canyon senza
                 controllo per alcuni metri, prima di sussultare nelle
                 fauci del baratro. Con un piccolo rimbalzo, ti colpisce
                 di striscio sulla fronte: cadi in avanti sanguinante
                 e... la sfera di pietra pomice diventa pi@`u piccola
                 oppure @`e la tua mano
                 che cresce, perch@'e ora ti sembra di tenerla, mentre
                 fissi l'Alligatore, figlio di sette Macao, le teste dei
                 suoi ultimi avversari infilzati sulle punte, una
                 congregazione che scalpita per il tuo sangue, e comunque
                 non c'@`e nulla che tu possa fare e... ma tutto questo
                 non ha senso e hai un mal di testa martellante.";
                }
            ],
  has   scenery open container;
```

Se la sfera di pietra pomice si trova nel baratro (if huge_ball in parent(self)), a ogni mossa del giocatore la direzione sud dell'oggetto Canyon_S (l'estremità inferiore del canyon) punta alla nuova locazione On_Ball (la sporgenza di pietra pomice). Per quanto riguarda invece le descrizioni, se la sfera di pietra pomice non è ancora rotolata nel baratro, Inform stampa a video il messaggio "La sfera di pietra pomice rotola giù nel canyon senza controllo..." altrimenti viene stampato a video il messaggio "L'estremità sud del canyon continua ora sopra la sfera di pietra pomice incastrata nel baratro.". Ecco la dimostrazione "pratica":

Estremità superiore del canyon
La più alta e vasta estremità a nord del canyon sale soltanto verso una parete irregolare di roccia calcarea vulcanica.

C'è un'enorme sfera di pietra pomice larga circa due metri e mezzo qui.

>spingi la sfera a sud
La sfera è difficile da fermare una volta che è stata mossa.

Xibalbá

Una piccola stele, o forse una pietra di confine, giace su una sporgenza ad altezza d'uomo.

>g

Xibalbá
Sei a cinquanta metri sotto la foresta pluviale e lo scroscio dell'acqua si sente ovunque: queste profonde ed erose miniere di calcare si estendono come radici. Verso nordest un percorso scivoloso, formato da un'ampia colonna di roccia ricoperta dal ghiaccio, ti riporta al Tempio, mentre un canyon prosegue verso l'alto a nord e verso il basso a sud. Alla luce diffusa della lampada al sodio proveniente da sopra, sembra essere bianco come i denti di uno squalo.

C'è un'enorme sfera di pietra pomice larga circa due metri e mezzo qui.

Una piccola stele, o forse una pietra di confine, giace su una sporgenza ad altezza d'uomo.

>spingi la sfera a sud
La sfera è difficile da fermare una volta che è stata mossa.

Estremità inferiore del canyon

La sfera di pietra pomice rotola giù nel canyon senza controllo per alcuni metri, prima di sussultare nelle fauci del baratro. Con un piccolo rimbalzo, ti colpisce di striscio sulla fronte: cadi in avanti sanguinante e... la sfera di pietra pomice diventa più piccola oppure è la tua mano che cresce, perché ora ti sembra di tenerla, mentre fissi l'Alligatore, figlio di sette Macao, le teste dei suoi ultimi avversari infilzati sulle punte, una congregazione che scalpita per il tuo sangue, e comunque non c'è nulla che tu possa fare e... ma tutto questo non ha senso e hai un mal di testa martellante.

>g

Estremità inferiore del canyon
L'estremità sud del canyon continua ora sopra la sfera di pietra pomice incastrata nel baratro.

>s

Sporgenza di pietra pomice
Una sporgenza improvvisa formata dalla sfera di pietra pomice è conficcata in un punto del baratro. Il canyon tuttavia termina qui.

Di tutte le offerte sacrificali gettate nel baratro, nulla forse sarà recuperato: fatta eccezione per un osso inciso, più luminoso di quello che sembra, che sporge da una polla di limo bagnato nella parete del canyon.

>

Qualcuno di voi può ancora confondersi con la nuova parte di codice relativa alla sfera di pietra pomice (l'oggetto huge_ball):

```
Object  -> huge_ball "enorme sfera di pietra pomice"
  with  name 'enorme' 'pomice' 'pietra' 'sfera',
        initial
            "C'@`e un'enorme sfera di pietra pomice larga circa due metri
             e mezzo qui.",
        description
            "@`E larga circa due metri e mezzo, sebbene sembri piuttosto
             leggera.",
        before [;
            PushDir:
                if (location == Junction && second == ne_obj)
                    "L'entrata del Tempio non @`e abbastanza larga.";
                AllowPushDir();
                rtrue;
            Pull, Push, Turn:
                "Non dovrebbe essere cos@`i difficile farla rotolare.";
            Take, Remove:
                "@`E troppo ingombrante.";
            ],
        after [;
            PushDir:
                if (second == n_obj) "Spingere la sfera in salita @`e
                                      troppo faticoso.";
                if (second == u_obj) <<PushDir self n_obj>>;
                if (second == s_obj) "La sfera @`e difficile da fermare
                                      una volta che @`e stata mossa.";
                if (second == d_obj) <<PushDir self s_obj>>;
            ],
  has   female static;
```

Per quanto riguarda l'azione PushDir appartenente alla proprietà before, se il giocatore si trova all'incrocio di Xibalbá e cerca di spingere la sfera (che deve ovviamente trovarsi anch'essa lì) verso nordest, Inform stampa a video un messaggio che ci avvisa che l'entrata del Tempio non è abbastanza larga. Ricordo ancora per l'ennesima volta, che l'istruzione second si riferisce alla direzione inclusa nel comando (ad esempio SPINGI LA SFERA A NORDEST, dove noun è la sfera e second è nordest).

🔴 Tra le innumerevoli possibilità offerte da Inform, c'è anche quella di stabilire se un oggetto debba essere visibile o meno al buio. Ecco infatti quello che succede quando il giocatore si trova nella Sala del Tempio e prova a spegnere la lampada al sodio:

Il Tempio
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

Sopra il ripiano dell'altare vedi una maschera facciale in mosaico di giada.

>spegni la lampada
Hai spento la lampada al sodio.

È completamente buio qui!

>esamina le pitture
Non vedi nulla del genere.

>accendi la lampada
Non vedi nulla del genere.

>i
Stai portando:
  una chiave di pietra
  un dizionario maya di Waldeck
  la mappa di Quintana Roo

>posa la mappa
Posata.

>g

Oscurità
L'oscurità intorno a te è opprimente e ti senti quasi soffocare.

>prendi la mappa
Non vedi nulla del genere.

>

Per non confondere troppo le idee, ho preferito disattivare momentaneamente il daemon degli insetti carnivori[24]. Come potete vedere, di default, se ci si trova al buio Inform non riconosce nessun oggetto all'infuori di quelli contenuti nell'inventario del giocatore. Quando però uno di essi viene posato a terra, diventa irriconoscibile come tutti gli altri. Ora, questa situazione mi sembra alquanto irreale ed esagerata; tenendo infatti conto che la lampada al sodio si trova accanto al nostro archeologo, non credo proprio che gli risulti così difficile riuscire ad accenderla anche se si trova al buio. La funzione di libreria InScope, posta subito dopo la funzione di libreria DeathMessage, è proprio quella che fa al caso nostro:

```
[ InScope temp;
   if (temp == player && location == thedark)
       PlaceInScope(sodium_lamp);
   return false;
];
```

Così facendo, rendiamo riconoscibile la lampada al sodio anche al buio:

Oscurità
L'oscurità intorno a te è opprimente e ti senti quasi soffocare.

>accendi la lampada
La lampada deve essere ben posizionata prima di essere accesa.

C'è un problema però: la lampada al sodio si può accendere solo se l'oggetto a cui appartiene ha l'attributo supporter. Occorre quindi fare una piccola modifica all'oggetto sodium_lamp:

```
if (location ~= thedark) {
   if (parent(self) hasnt supporter && self notin location)
      "La lampada deve essere ben posizionata prima di essere accesa.";
}
```

[24] Per farlo, cancellate o eventualmente commentate la definizione dell'oggetto tiny_claws nel file 3.11.inf.

Così facendo, se si cerca di accendere la lampada in una locazione che è diversa da thedark (quando cioè non si è al buio), viene verificato che la lampada sia effettivamente posata a terra, altrimenti si procede normalmente alla sua accensione:

Oscurità
L'oscurità intorno a te è opprimente e ti senti quasi soffocare.

>accendi la lampada
Hai acceso la lampada al sodio.

Corridoio in pendenza
Un corridoio basso e squadrato va da nord verso sud, inclinandosi verso la fine.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Il passaggio è bloccato da una massiccia porta di pietra gialla.

C'è una preziosa statuetta maya qui!

>

Possiamo anche fare in modo che la lampada possa essere riconosciuta al buio ed accesa solo in una locazione ben precisa (ad esempio il corridoio in pendenza):

```
[ InScope temp;
   if (temp == player && location == thedark && real_location ==
       Corridor)
       PlaceInScope(sodium_lamp);
   return false ;
];
```

o dappertutto tranne che nel corridoio:

```
[ InScope temp;
   if (temp == player && location == thedark && real_location ~=
       Corridor)
       PlaceInScope(sodium_lamp);
   return false ;
];
```

🔴 In una situazione reale, se siamo al buio e procediamo a tentoni all'interno di una grande piramide sepolcrale, corriamo il serio rischio di farci male se non addirittura di perdere la vita cadendo in una buca molto profonda. Che ci crediate o no, Inform ci viene incontro anche in questo caso, mettendoci a disposizione la funzione di libreria DarkToDark. Per prima cosa, occorre definire una variabile globale denominata dark:

```
.
.
.
Include "VerbLib";
Include "replace";

Global dark;
```

che vale false anche se non viene esplicitamente dichiarato. Segue poi la parte di codice relativa alla funzione vera e propria, che va posizionata subito dopo la funzione di libreria DeathMessage (o eventualmente InScope se presente):

```
[ DarkToDark;
  if (dark == false) {
      dark = true;
      "@`E pericoloso procedere con il buio; potresti cadere in una
       buca...";
  }
  if (random(3) == 1)
  {
      deadflag = 3;
      "Sei caduto in una buca e ti sei rotto tutte le ossa!";
  }
  rfalse; !o return false;
];
```

La variabile dark viene immediatamente testata: se vale false, le viene allora assegnato il valore true e viene stampato a video il relativo messaggio; altrimenti, si passa alla seconda if, dove viene generato un numero casuale da 1 a 3 a ogni mossa del giocatore. Se non "esce" il numero 1 il gioco può proseguire:

Il Tempio
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

La grande porta di pietra gialla è aperta.

Dietro la lastra, un sacerdote mummificato attende in piedi.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

Puoi anche vedere la mappa di Quintana Roo, un dizionario maya di Waldeck e una chiave di pietra qui.

>spegni lampada
Hai spento la lampada al sodio.

È completamente buio qui!

>n
È pericoloso procedere con il buio; potresti cadere in una buca...

Oscurità
L'oscurità intorno a te è opprimente e ti senti quasi soffocare.

>n

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>

altrimenti, ecco la fine che fa il nostro povero archeologo:

Oscurità

L'oscurità intorno a te è opprimente e ti senti quasi soffocare.

>s
Sei caduto in una buca e ti sei rotto tutte le ossa!

*** Sei morto ***

In questa partita hai totalizzato 0 punti su 30 possibili, in 28 turni, guadagnando il rango di Turista.

Vuoi RICOMINCIARE, CARICARE una partita salvata o USCIRE ?

Siete ritornati nella Sala del Tempio (dopo aver raccolto la stele ed esservi tolti la maschera), e notate che i cornicioni formano a sudest una curiosa anticamera. Spingete la lampada al sodio verso quella direzione e vi ritrovate davanti a una gabbia di ferro, alquanto sinistra, con la porta aperta. A un esame più attento potete notare che è abbastanza larga da poterci entrare dall'alto; provate allora a entrare, ma gli scheletri che popolano la gabbia ritornano in vita, bloccandovi con le loro mani ossute, schiacciandovi e prendendovi a pugni. Perdete conoscenza e quando vi risvegliate vi rendete conto che è accaduto qualcosa di impossibile e grottesco.
Siete fuori dalla gabbia, e vi chiedete come mai. Confusi e storditi, cercate di entrare di nuovo, ma vi rendete conto che non potete fare una cosa simile perché il vostro aspetto è simile a quello di un facocero, o meglio… SIETE DIVENTATI UN FACOCERO. Accidenti: e come diamine farete ora a fare la corte alla vostra vicina di casa?

§3.12 Una gabbia per un facocero (e l'archeologo che fine ha fatto?)

🔴 Abbiamo visto che in Inform è possibile creare un contenitore con la particolarità di poter essere aperto o chiuso (come ad esempio la cassa da imballaggio vista nel paragrafo 3.2). Esiste però anche il caso in cui il contenitore sia "trasparente" e che dia quindi la possibilità, al giocatore, di vedere gli oggetti in esso contenuti anche se è chiuso (come ad esempio la teca di vetro di un museo). L'attributo che permette di fare quanto detto è transparent:

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>chiudi la cassa
Ora hai chiuso la cassa d'imballaggio.

>esamina la cassa
Non puoi vedere dentro, perché la cassa d'imballaggio è chiusa.

>

Questo è quello che accade a un contenitore chiuso privo dell'attributo in questione: come è giusto che sia, se il giocatore cerca di esaminare il contenuto della cassa quando è chiusa, Inform stampa a video il relativo messaggio. Cosa accade invece se la cassa ha l'attributo transparent? È presto detto:

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>chiudi la cassa
Ora hai chiuso la cassa d'imballaggio.

>esamina la cassa
Dentro la cassa d'imballaggio vedi una macchina fotografica a lastre e un giornale di un mese fa.

>

Anche se è chiusa, il giocatore può vedere ugualmente il suo contenuto (immaginate, con un po' di fantasia, che sia fatta di vetro).

Rimanendo sempre in tema di contenitori, è arrivato il momento di esaminare la gabbia di ferro di Ruins. Ecco una prima versione semplificata:

```
Object  -> cage "gabbia di ferro"
  with  name 'ferro' 'gabbia' 'sbarre' 'struttura' 'simboli',
        description
            "Puoi leggere i seguenti simboli: Uccello Freccia Facocero.",
        inside_description
            "Sei circondato dalle sbarre della gabbia.",
        when_open
            "Una gabbia di ferro, di aspetto alquanto sinistro, ha la
             porta aperta. @`E abbastanza larga da poterci entrare
             dall'alto e ci sono alcuni simboli sulla struttura.",
        when_closed
            "La gabbia di ferro @`e chiusa.",
  has   female enterable transparent container openable open static;
```

la novità in questione riguarda la proprietà inside_description, il cui messaggio viene stampato a video solo se il giocatore si trova dentro la gabbia:

Anticamera
I cornicioni formano, a sudest del Tempio, una curiosa anticamera.

Una gabbia di ferro, di aspetto alquanto sinistro, ha la porta aperta. È abbastanza larga da poterci entrare dall'alto e ci sono alcuni simboli sulla struttura.

>entra nella gabbia
Ti trovi dentro la gabbia di ferro.

>g

Anticamera (dentro la gabbia di ferro)
I cornicioni formano, a sudest del Tempio, una curiosa anticamera.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Sei circondato dalle sbarre della gabbia.

>

ma dal momento che il suo aspetto è alquanto sinistro, è facile aspettarsi qualche "scherzetto". Ecco perché Nelson ha previsto la presenza degli scheletri, che si risvegliano e vi aggrediscono non appena entrate nella gabbia, come previsto dal seguente codice:

```
Object  -> cage "gabbia di ferro"
  with  name 'ferro' 'gabbia' 'sbarre' 'struttura' 'simboli',
        description
            "Puoi leggere i seguenti simboli: Uccello Freccia Facocero.",
        inside_description
            "Sei circondato dalle sbarre della gabbia.",
        when_open
            "Una gabbia di ferro, di aspetto alquanto sinistro, ha la
             porta aperta. @`E abbastanza larga da poterci entrare
             dall'alto e ci sono alcuni simboli sulla struttura.",
        when_closed
            "La gabbia di ferro @`e chiusa.",
        after [;
            Enter:
                print "Gli scheletri che popolano la gabbia ritornano in
                       vita, bloccandoti con le loro mani ossute,
                       schiacciandoti e prendendoti a pugni. Perdi
                       conoscenza e quando ti risvegli, ti rendi conto
                       che @`e accaduto qualcosa di impossibile e
                       grottesco...^";
                move warthog to Antechamber;
                remove skeletons;
                give self ~open;
                give warthog light;
                self.after = 0;
                ChangePlayer(warthog, 1);
                <<Look>>;
            ],
   has  female enterable transparent container openable open static;

Object  -> -> skeletons "scheletri"
  with  name 'scheletri' 'osso' 'teschio' 'ossa' 'teschi',
  has   pluralname static;
```

L'oggetto skeletons, come alcuni di voi avranno sicuramente notato, è privo della proprietà description. Questo è un trucco volutamente usato da Nelson per fare in modo che gli scheletri siano "invisibili"; essi cioè, sono presenti, ma il giocatore non lo sa fino a quando non si trova dentro la gabbia. Dopodiché, dopo essere stato bloccato, schiacciato e preso a pugni, il povero archeologo viene trasformato in un facocero:

Anticamera
I cornicioni formano, a sudest del Tempio, una curiosa anticamera.

Una gabbia di ferro, di aspetto alquanto sinistro, ha la porta aperta. È abbastanza larga da poterci entrare dall'alto e ci sono alcuni simboli sulla struttura.

>entra nella gabbia
Gli scheletri che popolano la gabbia ritornano in vita, bloccandoti con le loro mani ossute, schiacciandoti e prendendoti a pugni. Perdi conoscenza e quando ti risvegli, ti rendi conto che è accaduto qualcosa di impossibile e grottesco...

Anticamera (come facocero)
I cornicioni formano, a sudest del Tempio, una curiosa anticamera.

La lampada al sodio, posata a terra, si sta lentamente consumando.

La gabbia di ferro è chiusa.

>

Per chi non lo sapesse, il facocero è un mammifero simile al nostro cinghiale, vivente nelle boscaglie e nelle savane africane. Ha la testa grande e massiccia, con una vistosa criniera dietro la nuca, denti canini diretti lateralmente verso l'alto e in avanti, e due caratteristiche verruche sotto gli occhi. Ora che vi siete fatti un'idea più precisa di quello che siete diventati, ecco come viene definito il facocero in Ruins:

```
Object  warthog "facocero"
  with  name 'facocero' 'maiale' 'cinghiale',
        initial
            "Un facocero sbuffa e grugnisce nelle ceneri.",
        description
            "Infangato e ringhioso.",
        orders [;
            Go, Look, Examine, Smell, Taste, Touch, Search, Jump, Enter:
                rfalse;
            Eat:
                "Non hai ancora la capacit@`a di fiutare il cibo.";
            default:
                "Un facocero non pu@`o fare una cosa simile. Se non fosse
                per il peso e per la capacit@`a di vedere di notte, non
                sarebbe poi tanto peggio di molte persone.";
            ],
  has   animate proper;
```

Una volta trasformati, a parte spostarvi di locazione in locazione e vedere al buio, sono ben poche le cose che potete fare:

Anticamera (come facocero)
I cornicioni formano, a sudest del Tempio, una curiosa anticamera.

La lampada al sodio, posata a terra, si sta lentamente consumando.

La gabbia di ferro è chiusa.

>apri la gabbia
Un facocero non può fare una cosa simile. Se non fosse per il peso e per la capacità di vedere di notte, non sarebbe poi tanto peggio di molte persone.

>i
Un facocero non può fare una cosa simile. Se non fosse per il peso e per la capacità di vedere di notte, non sarebbe poi tanto peggio di molte persone.

>esamina la gabbia
Puoi leggere i seguenti simboli: Uccello Freccia Facocero.

>no

Il Tempio (come facocero)

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

Sopra il ripiano dell'altare vedi una maschera facciale in mosaico di giada.

>esamina me stesso
Infangato e ringhioso.

>

L'attributo proper è riservato a quegli oggetti il cui nome deve essere trattato come proprio di persona (non per niente a un certo punto rappresenta il giocatore stesso). Per quanto riguarda invece la "trasformazione", tutto verte sull'istruzione ChangePlayer, che deve avere come parametri il nome dell'oggetto in cui il giocatore deve essere trasformato (warthog) e i valori 0 (viene stampato a video solo il titolo della stanza in cui il giocatore si trova → ad es. Anticamera) o 1 (viene stampato a video il titolo della stanza insieme all'aspetto del giocatore → ad es. Anticamera (come facocero)).
L'istruzione self.after = 0; serve invece a far sì che gli scheletri compaiano una sola volta quando il nostro archeologo entra nella gabbia. Viene cioè, come afferma giustamente Paolo Lucchesi, "tolta" la proprietà after, in modo che non venga più invocata in futuro.

🔴 Ponendo la proprietà after dell'oggetto cage uguale a zero, risolviamo sì il problema degli scheletri, ma non possiamo più usarla. Possiamo invece ricorrere all'attributo general:

```
Object  -> cage "gabbia di ferro"
.
.
.
       after [;
            Enter:
                if (self hasnt general)
                {
                   print "Gli scheletri che popolano la gabbia...
                   move warthog to Antechamber;
                   remove skeletons;
                   give self ~open;
                   give warthog light;
                   give self general;
                   ChangePlayer(warthog, 1);
                   <<Look>>;
                }
            ],
.
.
.
has female ~general enterable transparent container openable open static;
```

mediante il quale otteniamo lo stesso risultato di prima, senza però dover necessariamente rinunciare alla proprietà in questione.

🔴 In Inform, se vogliamo nascondere un oggetto alla vista del giocatore, possiamo anche ricorrere all'attributo concealed:

```
Object  -> -> skeletons "scheletri"
  with  name 'scheletri' 'osso' 'teschio' 'ossa' 'teschi',
        initial
           "Ci sono degli scheletri qui.",
        before [;
```

```
            Examine: "Sono pieni di ragnatele.";
        ],
  has   pluralname static concealed;
```

Il risultato ottenuto è assolutamente identico a quello precedente, ma si ha il vantaggio che ora è possibile definire l'oggetto in questione come si vuole. Gli scheletri potrebbero infatti servire in qualche altro ipotetico punto del gioco, e sarebbe davvero un peccato dover obbligatoriamente rinunciare alle proprietà description e initial.

La vostra situazione è terribile: così conciati, le ragazze non vi degneranno neanche di uno sguardo. Non che prima vi guardassero poi più di tanto (una su duecento), ma adesso ogni vostra possibilità è andata persa. Delusi e disperati, decidete di farla finita e vi dirigete verso la Tana del Verme, convinti che verrete divorati quanto prima da quella bestiaccia schifosa. Ma più passa il tempo, più la tana diventa scivolosa intorno al vostro corpo di facocero, e strillate senza volerlo come annaspate nell'oscurità, precipitando infine a sud verso una cripta. Vi guardate intorno: la cripta è arredata di mensole, otturata con un sigillo di terracotta e contenente alcune figure dipinte che potrebbero rappresentare i Nove Signori Della Notte. C'è anche un favo splendidamente conservato, che forse rappresenta qualche tipo di offerta funeraria.

§3.13 La Cripta dei Nove Signori Della Notte (o sette come i nani?)

La Tana del Verme, che abbiamo già visto nel paragrafo 3.6, presenta ora delle novità:

```
Object  Wormcast "Tana Del Verme"
  with  description
           .
           .
           .
        cant_go [;
            if (player ~= warthog)
                "Cominci a capire che c'@`e qualcosa dietro la tana,
                 perch@'e questa via deve essere stata creata da qualche
                 sorta di animale strisciante, troppo stretta per il tuo
                 pancione da archeologo.";
            print "La tana diventa scivolosa intorno al tuo corpo di
                   facocero e strilli senza volerlo come annaspi
                   nell'oscurit@`a, precipitando infine a sud verso...^";
            PlayerTo(Burial_Shaft);
            rtrue;
            ],
        after [;
            Drop:
                move noun to Square_Chamber;
                print_ret (The) noun,
                          " scivola attraverso uno dei cunicoli e ",
                           (itorthem) noun, " perdi rapidamente di
                           vista.";
            ],
  has   light;
```

Quando il giocatore cerca di andare in una direzione che non porta da nessuna parte, se non è stato ancora trasformato in un facocero, Inform si limita a stampare a video il relativo messaggio. Diverso è invece il caso in cui il giocatore sia il facocero in questione. Dopo che Inform ha stampato a video il relativo messaggio di appartenenza, entra in gioco l'istruzione PlayerTo, che si usa quando si vuole spostare il giocatore in una locazione qualsiasi definita all'interno dell'avventura stessa, o eventualmente in un contenitore con l'attributo enterable.
Sono previsti poi due possibili parametri: 1 (Inform sposta il giocatore nella locazione indicata tra le parentesi tonde senza però stampare a video la descrizione della nuova locazione):

Tana Del Verme (come facocero)

>n

La tana diventa scivolosa intorno al tuo corpo di facocero e strilli senza volerlo come annaspi nell'oscurità, precipitando infine a sud verso...

>

e 2 (Inform stampa a video la descrizione della nuova locazione):

La Sala Quadrata (come facocero)

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>e

Tana Del Verme (come facocero)

>n
La tana diventa scivolosa intorno al tuo corpo di facocero e strilli senza volerlo come annaspi nell'oscurità, precipitando infine a sud verso...

La Cripta (come facocero)
Sui tuoi appunti di lavoro puoi leggere: "Una cripta arredata di mensole, otturata con un sigillo di terracotta e contenente alcune figure dipinte che potrebbero rappresentare i Nove Signori Della Notte. Delle ossa sparpagliate sembrerebbero appartenere a un uomo anziano e a diversi bambini sacrificati, mentre altri resti funerari comprendono anche le zampe di un giaguaro." (Negli appunti è importante non far notare che sei spaventato.)

C'è un favo splendidamente conservato qui.

>

Se viene chiamata la funzione con il solo nome della locazione (come accade di fatto in Ruins), il valore di default è 2 (o, più tecnicamente, true, che per Inform in questo caso è uguale a 2). A ogni modo, RICORDATEVI DI FAR SEGUIRE LA CHIAMATA ALLA FUNZIONE CON L'ISTRUZIONE RTRUE.

▲ Grazie all'istruzione PlayerTo, è possibile creare lo stesso effetto della parola magica xyzzy presente in Avventura[25]. È possibile, cioè, fare in modo che se il giocatore si trova in una specifica locazione e digita la parola xyzzy (ma può essere una qualsiasi) viene spostato in un'altra locazione e viceversa:

Il Tempio
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

Sopra il ripiano dell'altare vedi una maschera facciale in mosaico di giada.

>xyzzy

Il Grande Altopiano

[25] Adventure o Colossal Cave (tradotta in italiano da Giovanni Riccardi) è la prima avventura testuale ad essere mai stata scritta. Ulteriori informazioni potete trovarle all'indirizzo http://www.rickadams.org/adventure/.

I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>xyzzy

Il Tempio
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

La grande porta di pietra gialla è aperta.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

Sopra il ripiano dell'altare vedi una maschera facciale in mosaico di giada.

>

Come avete appena visto, le due locazioni interessate sono Forest e Shrine. Andiamo ora a vedere come questi due oggetti sono stati modificati:

```
Object  Forest "Il Grande Altopiano"
  with  description
        .
        .
        .
        before [;
            Listen: "Urla di scimmie, pipistrelli, pappagalli, macao.";
            Xyzzy:  if (Shrine hasnt visited) rfalse;
                    PlayerTo(Shrine); rtrue;
            ],
  has   light;

Object  Shrine "Il Tempio"
  with  description
            "Questo magnifico tempio mostra segni di scavi da
             preesistenti miniere di calcare, specialmente verso il lato
             occidentale, dove due lunghi cornicioni si dirigono verso
             sud.",
        before [;
             Xyzzy: PlayerTo(Forest); rtrue;
             ],
        .
        .
        .
```

Nell'oggetto Forest, l'istruzione PlayerTo viene eseguita solo se il giocatore è stato almeno una volta nella Sala del Tempio. Nell'oggetto Shrine, invece, la funzione viene eseguita sempre (la locazione Forest infatti, dal momento che è proprio quella di partenza, è sempre "visitata" almeno una volta dal giocatore). Occorre inoltre definire l'azione xyzzy che, per ovvie ragioni, non è fra quelle standard:

```
Include "ItalianG";

[ PhotographSub;
      .
      .
      .
        ];

[ XyzzySub;
        "Non accade nulla...";
        ];

Verb 'fotografa' * noun                                 -> Photograph;
Verb 'xyzzy' *                                          -> Xyzzy;
```

Nulla di complicato, come potete vedere. Dal momento che questa azione è così semplice, corrisponde allora in Inform alla proprietà before. Inoltre, se il giocatore digita la parola magica quando non è ancora andato almeno una volta nella Sala del Tempio o se non si trova nel Grande Altopiano o nella sala medesima:

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>xyzzy
Non accade nulla...

>prendi il fungo
Hai raccolto abilmente il fungo, senza staccarlo dal suo gambo sottile.

>mangia il fungo
Lo sgranocchi ad un angolo, incapace di capire l'origine di un gusto così acre, distratto dal volo di un macao sopra la tua testa che sembra quasi un'esplosione nel sole. Il battito delle sue ali è quasi assordante, e delle pietre crollano una sull'altra.

>giù

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>xyzzy
Non accade nulla...

```
>
```

ecco quello che accade (cioè nulla).

🔴 Per quanto riguarda invece la Cripta, c'è un piccolo problema: entrando (o meglio, precipitando) come facocero, non è possibile leggere gli appunti di lavoro. Eppure la descrizione di questa nuova locazione afferma proprio il contrario: ora, dal momento che, come vedremo meglio nel prossimo paragrafo, è possibile accedere alla Cripta anche in forma "umana", non basta cancellare la parte di descrizione inerente agli appunti di lavoro, ma occorre modificare l'oggetto Burial_Shaft nel seguente modo:

```
Object  Burial_Shaft "La Cripta"
  with  description [;
            if (player ~= warthog)
                "Sui tuoi appunti di lavoro puoi leggere: ~Una cripta
                 arredata di mensole, otturata con un sigillo di
                 terracotta e contenente alcune figure dipinte che
                 potrebbero rappresentare i Nove Signori Della Notte.
                 Delle ossa sparpagliate sembrerebbero appartenere a un
                 uomo anziano e a diversi bambini sacrificati, mentre
                 altri resti funerari comprendono anche le zampe di un
                 giaguaro.~ (Negli appunti @`e importante non far notare
                 che sei spaventato.)";
            "Sei in una cripta arredata di mensole, otturata con un
             sigillo di terracotta e contenente alcune figure dipinte che
             potrebbero rappresentare i Nove Signori Della Notte. Delle
             ossa sparpagliate sembrerebbero appartenere a un uomo
             anziano e a diversi bambini sacrificati, mentre altri resti
             funerari comprendono anche le zampe di un giaguaro.";
        ],
        n_to Wormcast,
        cant_go
            "Gli architetti di questa sala sono stati avari nel fornirla
             di uscite. Alcuni facoceri, comunque, sembrano aver scavato
             dei cunicoli da nord.",
  has   female light;
```

Se il giocatore non è un facocero, Inform stampa a video il messaggio "Suoi tuoi appunti di lavoro…", altrimenti viene stampato a video "Sei in una cripta arredata di mensole…". Vi ricordo ancora una volta che, PER UN'ISTRUZIONE IF, SI USANO LE PARENTESI GRAFFE APERTE ({) E CHIUSE (}) SOLO SE AD ESSA CORRISPONDONO PIÙ ISTRUZIONI; ALTRIMENTI, COME IN QUESTO CASO, SI OMETTONO.

Volete tornare da dove siete venuti ma, nonostante alcuni facoceri sembrino aver scavato dei cunicoli da nord, gli architetti di questa sala sono stati davvero avari nel fornirla di uscite. Testardi come solo un facocero del vostro calibro sa fare, con un balzo poderoso cozzate sopra il sigillo di terracotta posto al di sopra della sala, facendovi crollare addosso cenere e terra. Qualcosa privo di vita e terribilmente pesante cade su di voi. Perdete conoscenza e quando vi risvegliate vi rendete conto che… siete ritornati umani!!!
Che meraviglia. Ora le ragazze vi guarderanno di nuovo… a parte la pancia, le rughe, l'occhio di vetro, i baffi finti e la parrucca. Beh, "piccoli" dettagli a parte, vi concentrate ora sul favo. In effetti potreste anche mangiarlo, ma vecchio com'è non saprà ormai più di nulla. Forse è meglio fotografarlo, raccoglierlo e metterlo poi nella cassa da imballaggio; al massimo, potrete sempre assaporarlo quando (o meglio, se) ritornerete alla Fondazione Carneige…

§3.14 Un archeologo per un facocero (basta solo che non grugnite)

Così com'è possibile trasformare un archeologo in un facocero, è anche possibile il contrario. Osserviamo il seguente codice:

```
Object  Burial_Shaft "La Cripta"
  with  description
            "Sui tuoi appunti di lavoro puoi leggere: ~Una cripta
             arredata di mensole, otturata con un sigillo di terracotta e
             contenente alcune figure dipinte che potrebbero
             rappresentare i Nove Signori Della Notte. Delle ossa
             sparpagliate sembrerebbero appartenere a un uomo anziano e a
             diversi bambini sacrificati, mentre altri resti funerari
             comprendono anche le zampe di un giaguaro.~ (Negli appunti
             @`e importante non far notare che sei spaventato.)",
        n_to Wormcast,
        u_to [;
            move selfobj to self;
            print "Con un balzo poderoso, cozzi sopra il sigillo di
                   terracotta posto al di sopra della sala, facendoti
                   crollare addosso cenere e terra. Qualcosa privo di
                   vita e terribilmente pesante cade su di te. Perdi
                   conoscenza e quando ti risvegli ti rendi conto che @`e
                   accaduto qualcosa di impossibile e grottesco...^";
            ChangePlayer(selfobj);
            give warthog ~light ~proper;
            <<Look>>;
            ],
        cant_go
            "Gli architetti di questa sala sono stati avari nel fornirla
             di uscite. Alcuni facoceri, comunque, sembrano aver scavato
             dei cunicoli da nord.",
        before [; Jump: <<Go u_obj>>; ],
  has   female light;
```

Quando il giocatore si dirige verso l'alto, entra in azione la parola chiave selfobj che viene dapprima trasferita a self (il giocatore è tornato ad essere un archeologo) per poi finire nelle "mani" dell'istruzione ChangePlayer (che effettua di fatto la trasformazione). Occorre anche negare gli attributi light e proper:

Tana Del Verme (come facocero)

>n
La tana diventa scivolosa intorno al tuo corpo di facocero e strilli senza volerlo come annaspi nell'oscurità, precipitando infine a sud verso...

La Cripta (come facocero)
Sui tuoi appunti di lavoro puoi leggere: "Una cripta arredata di mensole, otturata con un sigillo di terracotta e contenente alcune figure dipinte che potrebbero rappresentare i Nove Signori Della Notte. Delle ossa sparpagliate sembrerebbero appartenere a un uomo anziano e a diversi bambini sacrificati, mentre altri resti funerari comprendono anche le zampe di un giaguaro." (Negli appunti è importante non far notare che sei spaventato.)

C'è un favo splendidamente conservato qui.

>esamina me stesso
Infangato e ringhioso.

>su
Con un balzo poderoso, cozzi sopra il sigillo di terracotta posto al di sopra della sala, facendoti crollare addosso cenere e terra. Qualcosa privo di vita e terribilmente pesante cade su di te. Perdi conoscenza e quando ti risvegli ti rendi conto che è accaduto qualcosa di impossibile e grottesco...

La Cripta
Sui tuoi appunti di lavoro puoi leggere: "Una cripta arredata di mensole, otturata con un sigillo di terracotta e contenente alcune figure dipinte che potrebbero rappresentare i Nove Signori Della Notte. Delle ossa sparpagliate sembrerebbero appartenere a un uomo anziano e a diversi bambini sacrificati, mentre altri resti funerari comprendono anche le zampe di un giaguaro." (Negli appunti è importante non far notare che sei spaventato.)

Un facocero sbuffa e grugnisce nelle ceneri.

C'è un favo splendidamente conservato qui.

>esamina me stesso
Hai sempre lo stesso bell'aspetto.

>

Dal momento che alcuni facoceri sembrano aver scavato dei cunicoli da nord, Nelson ha (giustamente) pensato di inserirne uno di essi non appena il giocatore ridiventa un archeologo. L'oggetto in questione è sempre il solito warthog, ma questa volta è privo dell'attributo proper (non è più, quindi, un nome proprio di persona e Inform può così trattarlo come un comune "oggetto" associandogli, quando occorre, il relativo articolo). Se invece vogliamo eliminarlo del tutto, basta sostituire la riga give warthog ~light ~proper; con un semplice remove warthog; come accade nel seguente esempio:

Tana Del Verme (come facocero)

>n
La tana diventa scivolosa intorno al tuo corpo di facocero e strilli senza volerlo come annaspi nell'oscurità, precipitando infine a sud verso...

La Cripta (come facocero)
Sui tuoi appunti di lavoro puoi leggere: "Una cripta arredata di mensole, otturata con un sigillo di terracotta e contenente alcune figure dipinte che potrebbero rappresentare i Nove Signori Della Notte. Delle ossa sparpagliate sembrerebbero appartenere a un uomo anziano e a diversi bambini sacrificati, mentre altri resti funerari comprendono anche le zampe di un giaguaro." (Negli appunti è importante non far notare che sei spaventato.)

C'è un favo splendidamente conservato qui.

>su
Con un balzo poderoso, cozzi sopra il sigillo di terracotta posto al di sopra della sala, facendoti crollare addosso cenere e terra. Qualcosa privo di vita e terribilmente pesante cade su di te. Perdi conoscenza e quando ti risvegli ti rendi conto che è accaduto qualcosa di impossibile e grottesco...

La Cripta
Sui tuoi appunti di lavoro puoi leggere: "Una cripta arredata di mensole, otturata con un sigillo di terracotta e contenente alcune figure dipinte che potrebbero rappresentare i Nove Signori Della Notte. Delle ossa sparpagliate sembrerebbero appartenere a un uomo anziano e a diversi bambini sacrificati, mentre altri resti funerari comprendono anche le zampe di un giaguaro." (Negli appunti è importante non far notare che sei spaventato.)

C'è un favo splendidamente conservato qui.

>esamina il facocero
Non vedi nulla del genere.

>esamina me stesso
Hai sempre lo stesso bell'aspetto.

>

⚠ In Inform è possibile stabilire che, dopo un certo numero di mosse effettuate dal giocatore, accada qualcosa. Possiamo allora provare a modificare il favo nel seguente modo:

```
Treasure -> Honeycomb "favo"
  with  name 'antico' 'vecchio' 'miele' 'favo' 'cera',
        initial
            "C'@`e un favo di cera splendidamente conservato qui.",
        description
            "Forse qualche tipo di offerta funeraria.",
        after [;
            Eat:
                "Questo @`e probabilmente il pasto pi@`u costoso della
                 tua vita. Il miele ha uno strano gusto, forse perch@'e
                 @`e stato usato per conservare i visceri dei sovrani
                 sepolti qui, ma sembra ancora miele.";
            Drop:
                StartTimer(self, 2);
                "Il favo comincia immediatamente ad ammorbidirsi.";
            Take:
                StopTimer(self);
                "La cera, sebbene sia molto molle, conserva ancora la
                 forma di un favo.";
            ],
        time_left 0,
        time_out [;
            if (self in location)
                print "^Il favo, diventato ormai della cera liquida, si
                       espande come una pozzanghera ed evapora.^";
            remove self;
            ],
  has  edible;
```

Inform ha un timer interno che viene attivato dall'istruzione StartTimer che richiede, come parametri, l'oggetto stesso (self) e il numero di mosse (2) dopo le quali accade l'evento da noi stabilito.

L'attivazione del timer avviene quando il giocatore posa da qualche parte il favo di cera che, dopo due mosse, evapora se non viene più raccolto:

La Cripta
Sui tuoi appunti di lavoro puoi leggere: "Una cripta arredata di mensole, otturata con un sigillo di terracotta e contenente alcune figure dipinte che potrebbero rappresentare i Nove Signori Della Notte. Delle ossa sparpagliate sembrerebbero appartenere a un uomo anziano e a diversi bambini sacrificati, mentre altri resti funerari comprendono anche le zampe di un giaguaro." (Negli appunti è importante non far notare che sei spaventato.)

C'è un favo splendidamente conservato qui.

>posa tutto eccetto la macchina
dizionario maya di Waldeck: Posato.
lampada al sodio: Posata.
mappa di Quintana Roo: Posata.

>fotografa il favo
Prepari l'elefantiaca macchina fotografica a lastre, sistemi la lampada al sodio e metti pazientemente in posa il favo.

>prendi il favo
La cera, sebbene sia molto molle, conserva ancora la forma di un favo.

>posa il favo
Il favo comincia immediatamente ad ammorbidirsi.

>aspetta
Il tempo passa.

>aspetta
Il tempo passa.

Il favo, diventato ormai della cera liquida, si espande come una pozzanghera ed evapora.

>g

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

La lampada al sodio è posata saldamente a terra.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

Puoi anche vedere la mappa di Quintana Roo e un dizionario maya di Waldeck qui.

>

Se invece viene raccolto prima che vengano effettuate le due fatidiche mosse, allora il favo è "salvo":

La Cripta
Sui tuoi appunti di lavoro puoi leggere: "Una cripta arredata di mensole, otturata con un sigillo di terracotta e contenente alcune figure dipinte che potrebbero rappresentare i Nove Signori Della Notte. Delle ossa sparpagliate sembrerebbero appartenere a un uomo anziano e a diversi bambini sacrificati, mentre altri resti funerari comprendono anche le zampe di un giaguaro." (Negli appunti è importante non far notare che sei spaventato.)

C'è un favo splendidamente conservato qui.

>posa tutto eccetto la macchina
dizionario maya di Waldeck: Posato.
lampada al sodio: Posata.
mappa di Quintana Roo: Posata.

>fotografa il favo
Prepari l'elefantiaca macchina fotografica a lastre, sistemi la lampada al sodio e metti pazientemente in posa il favo.

>prendi il favo
La cera, sebbene sia molto molle, conserva ancora la forma di un favo.

>posa il favo
Il favo comincia immediatamente ad ammorbidirsi.

>aspetta
Il tempo passa.

>prendi il favo
La cera, sebbene sia molto molle, conserva ancora la forma di un favo.

>i
Stai portando:
  un favo
  una macchina fotografica a lastre

>

Ricordatevi infine, che l'istruzione StartTimer ha bisogno, per poter funzionare, della variabile locale time_left posta uguale a zero, e della funzione time_out (nella quale viene di fatto eliminato l'oggetto in questione). Un esercizio efficace (ma anche contorto) per esplorare a fondo un'altra utilissima caratteristica di questo straordinario linguaggio di programmazione.

Vi dirigete verso l'alto e vi ritrovate dentro la gabbia di ferro (questa volta senza scheletri). Dal pavimento della gabbia, un pozzo di terracotta scende nella cripta, ma voi non ne volete assolutamente sapere di tornare là sotto; c'è troppa umidità, che fa solo male alle vostre povere ossa. Uscite dalla gabbia e spingete la lampada al sodio verso nordovest. Dovete fare presto, perché la luce da essa emessa diventa sempre più debole: procedete allora verso nord più velocemente che potete, fino poi a ritrovarvi nella Sala Quadrata. Qui, la lampada al sodio si affievolisce e improvvisamente si spegne. Siete letteralmente disperati: la vostra cara lampada è morta e non potete neanche farle un funerale. La raccogliete con le lacrime agli occhi e salite finalmente verso la superficie. Vi dirigete verso la vostra cassa da imballaggio, depositate tutti i tesori che avete trovato e vi lasciate infine cadere, esausti come non mai, per terra.
Un macao dalla coda rossa fluttua giù dalle cime degli alberi, con le piume appesantite dalla pioggia recente. Il battito delle sue ali è quasi assordante, e delle pietre crollano una sull'altra. Come il cielo si schiarisce, una luna crescente sorge sopra una tranquilla giungla. È la fine di marzo del 1938, ed è l'ora di tornare a casa…

§3.15 La fine di Ruins è giunta (ma con Inform siete solo all'inizio)

Spesso, non basta assegnare una semplice direzione a una stanza. È quello che accade in questa "nuova" versione della Cripta:

```
Object  Burial_Shaft "La Cripta"
  with  description
            "Sui tuoi appunti di lavoro puoi leggere...";
        n_to Wormcast,
        u_to [;
            cage.floor_open = true;
            self.u_to = self.opened_u_to;
            move selfobj to self;
            print "Con un balzo poderoso, cozzi sopra il sigillo di
                   terracotta posto…";
            ChangePlayer(selfobj);
            give warthog ~light ~proper;
            <<Look>>;
            ],
        cant_go
            "Gli architetti di questa sala sono stati avari nel fornirla
             di uscite. Alcuni facoceri, comunque, sembrano aver scavato
             dei cunicoli da nord.",
        before [; Jump: <<Go u_obj>>; ],
        opened_u_to [; PlayerTo(cage); rtrue; ],
  has   female light;
```

Quando il giocatore si dirige verso l'alto (quando tenta, cioè, di uscire dalla cripta), accadono diverse cose:

- la variabile locale floor_open (vedremo fra poco a cosa serve) appartenente all'oggetto cage (la gabbia di ferro) vale true;
- l'istruzione u_to punta alla funzione opened_u_to;
- il giocatore viene, come abbiamo già visto nel paragrafo precedente, trasformato da facocero in archeologo.

La funzione opened_u_to, a sua volta, "teletrasporta" il giocatore nella gabbia di ferro[26]:

[26] Quest'azione è possibile perché l'oggetto cage ha l'attributo enterable.

```
Object  -> cage "gabbia di ferro"
  with  name 'ferro' 'gabbia' 'sbarre' 'struttura' 'simboli',
        description
            "Puoi leggere i seguenti simboli: Uccello Freccia Facocero.",
        inside_description [;
            if (self.floor_open)
                "Dal pavimento della gabbia, un pozzo di terracotta
                 scende nella cripta.";
            "Sei circondato dalle sbarre della gabbia.";
            ],
        when_open
            "Una gabbia di ferro, di aspetto alquanto sinistro, ha la
             porta aperta. @`E abbastanza larga da poterci entrare
             dall'alto e ci sono alcuni simboli sulla struttura.",
        when_closed
            "La gabbia di ferro @`e chiusa.",
        after [;
            Enter:
                print "Gli scheletri che popolano la gabbia ritornano in
                       vita, bloccandoti con le loro mani ossute,
                       schiacciandoti e prendendoti a pugni. Perdi
                       conoscenza e quando ti risvegli, ti rendi conto
                       che @`e accaduto qualcosa di impossibile e
                       grottesco...^";
                move warthog to Antechamber;
                remove skeletons;
                give self ~open;
                give warthog light;
                self.after = 0;
                ChangePlayer(warthog, 1);
                <<Look>>;
            ],
        react_before [;
            Go:
                if (noun == d_obj && self.floor_open) {
                    PlayerTo(Burial_Shaft);
                    rtrue;
                    }
            ],
        floor_open false,
  has   female enterable transparent container openable open static;
```

Per quanto riguarda la proprietà inside_description, se la variabile locale floor_open vale true, allora Inform stampa a video il messaggio "Dal pavimento della gabbia, un pozzo di terracotta scende nella cripta.". Altrimenti (se, cioè, floor_open vale false), stampa a video il messaggio alternativo ("Sei circondato dalle sbarre della gabbia."):

Anticamera (dentro la gabbia di ferro)
I cornicioni formano, a sudest del Tempio, una curiosa anticamera.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Dal pavimento della gabbia, un pozzo di terracotta scende nella cripta.

>

Dal momento che la proprietà after vale zero gli scheletri (come abbiamo già visto nel paragrafo 3.12), non appaiono più, mentre la proprietà react_before fa sì che, quando il giocatore si dirige verso il basso con la variabile locale floor_open = true, sia possibile accedere alla Cripta.

Una volta raccolti e depositati tutti i tesori nella cassa da imballaggio, viene assegnato il valore 2 alla variabile di libreria deadflag nella classe Treasure:

```
Class   Treasure
  with  before [;
      .
      .
      .
      if (score == MAX_SCORE) {
          deadflag = 2;
          print_ret "Come depositi con attenzione ", (the) noun,
                    " un macao dalla coda rossa fluttua gi@`u
                     dalle cime degli alberi, con le piume appesantite
                     dalla pioggia recente. Il battito delle sue ali @`e
                     quasi assordante, e delle pietre crollano una
                     sull'altra.^
                     Come il cielo si schiarisce, una luna crescente sorge
                     sopra una tranquilla giungla. @'E la fine di marzo
                     del 1938, ed @`e l'ora di tornare a casa.";
      }
      .
      .
      .
      cultural_value 5,
      photographed_in_situ false;
```

e l'avventura termina con una bella vittoria:

Il Grande Altopiano
La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>posa il favo nella cassa
Come depositi con attenzione il favo un macao dalla coda rossa fluttua giù dalle cime degli alberi, con le piume appesantite dalla pioggia recente. Il battito delle sue ali è quasi assordante e delle pietre crollano una sull'altra.
Non appena il cielo si schiarisce, una luna crescente sorge sopra una tranquilla
giungla. È la fine di marzo del 1938, ed è l'ora di tornare a casa.

*** Hai vinto ***

In questa partita hai totalizzato 30 punti su 30 possibili, in 117 turni, guadagnando il rango di Direttore della Fondazione Carneige.

Vuoi RICOMINCIARE, CARICARE una partita salvata o USCIRE ?
> uscire

⚠Un altro esercizio che può essere utile a scopo didattico, è la creazione di un trono su cui potersi sedere sopra:

```
Object  Square_Chamber "La Sala Quadrata"...

Object  -> "iscrizioni scolpite"...
```

```
Object  -> sunlight "raggio di sole"...

Object -> throne "trono di pietra"
  with name 'trono' 'pietra',
       initial
           "Alcuni gradini di pietra, bassi e squadrati, portano verso un
            trono maestoso, anch'esso di pietra.",
       after [;
          Enter: print_ret "Ti sei seduto sul trono.";
          Exit: print_ret "Ora sei di nuovo in piedi.";
       ],
  has supporter enterable;
```

Questo oggetto, come potete facilmente vedere, si trova nella Sala Quadrata. Ecco come funziona:

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

Alcuni gradini di pietra, bassi e squadrati, portano verso un trono maestoso, anch'esso di pietra.

>siediti sul trono
Ti sei seduto sul trono.

>g

La Sala Quadrata (sopra il trono di pietra)
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>s
Dovresti toglierti da sopra il trono di pietra prima.

>alzati dal trono
Questo è un verbo che non conosco.

>

Il primo problema da risolvere è che in Infit (v2.5) manca di fatto la definizione del verbo alzati, ma, come dice un vecchio detto, a tutto c'è rimedio:

```
Verb 'alzati' = 'esci';
```

Grazie a questa riga di codice (da posizionare alla fine del listato di Ruins, subito sotto la riga di codice Verb 'fotografa' * noun…, abbiamo il nostro verbo funzionante e pronto per essere usato:

La Sala Quadrata (sopra il trono di pietra)
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>s
Dovresti toglierti da sopra il trono di pietra prima.

>alzati dal trono
Ora sei di nuovo in piedi.

>g

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

Alcuni gradini di pietra, bassi e squadrati, portano verso un trono maestoso, anch'esso di pietra.

>

Volendo, è possibile fare in modo che il giocatore non sia costretto a digitare tutte le volte i comandi SIEDITI SUL TRONO e ALZATI DAL TRONO ma, più semplicemente, SIEDITI e ALZATI. Ecco allora le modifiche da apportare all'oggetto throne:

```
Object -> throne "trono di pietra"
  with name 'trono' 'pietra',
       initial
           "Alcuni gradini di pietra, bassi e squadrati, portano verso un
            trono maestoso, anch'esso di pietra.",
       after [;
          Enter: print_ret "Ti sei seduto sul trono.";
          Exit: print_ret "Ora sei di nuovo in piedi.";
       ],
       react_before [;
          Enter: if (noun == nothing) <<Enter self>>;
          Exit:  if (noun == nothing && player in self) <<Exit self>>;
       ],
  has supporter enterable;
```

L'uso più comune della proprietà react_before, come abbiamo gia visto nel paragrafo 3.4, è quello di evitare le azioni senza oggetto. Per quanto riguarda l'azione Enter, se noun == nothing (se cioè il giocatore digita solo SIEDITI), Inform fa sedere l'archeologo sul trono; per quanto riguarda invece l'azione Exit, se noun == nothing (se cioè il giocatore digita solo ALZATI), Inform fa alzare l'archeologo dal trono, solo però se è seduto su di esso (&& player in self). Ma c'è ancora un piccolo problema:

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

Alcuni gradini di pietra, bassi e squadrati, portano verso un trono maestoso, anch'esso di pietra.

>siediti
Su cosa vuoi sederti?

>siediti sul trono
Ti sei seduto sul trono.

>alzati

Ora sei di nuovo in piedi.

>

Come potete vedere, il comando SIEDITI non funziona ancora da solo. Il problema è relativo alla definizione del verbo siediti in Infit:

```
Verb 'siedi' 'siediti' 'sdraiati'
                * 'su'/'sul'/'sullo'/'sull^'/'sulla'/'sui'/
                  'sugli'/'sulle'/'sopra' noun       -> Enter
            * 'a'/'ad'/'all^'/'allo'/'alla'/'al'/'agli'/'ai'/
                  'alle' noun                        -> Enter
                * 'dentro'/'in'/'nel'/'nello'/'nell^'/'nella'/
                  'negli'/'nelle'/'nei' noun         -> Enter;
```

Manca, in poche parole, la seguente riga di codice:

```
Extend 'siediti' first *                                -> Enter;
```

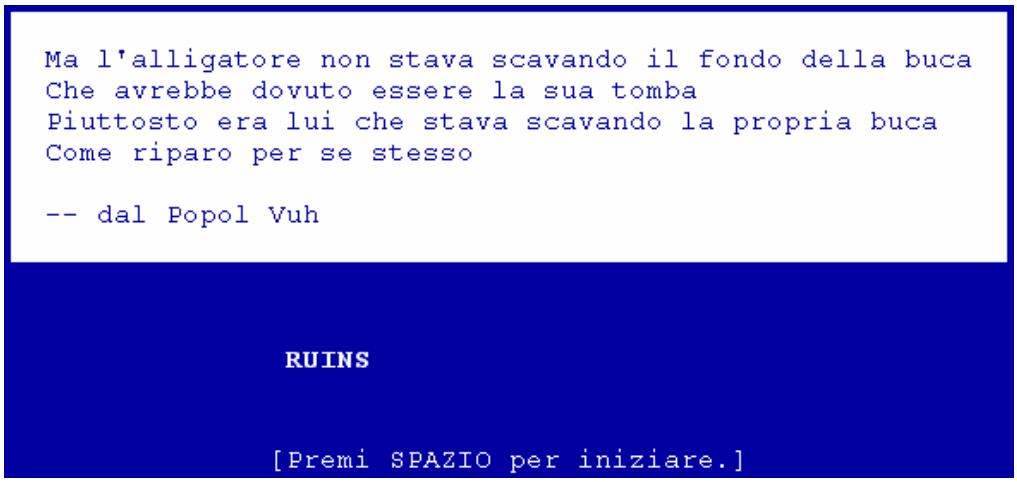
che va inserita alla fine del listato di Ruins subito sotto la riga di codice Verb 'alzati' = 'esci';
Provare per credere...

* *

L'ultima parte di codice da esaminare in Ruins, è la funzione TitlePage:

```
[ TitlePage i;
        ClearScreen(); print "^^^^^^^^^^^^^";
        i = ScreenWidth(); if (i > 30) i = (i-30)/2;
        style bold; font off; spaces(i);
        print " RUINS^";
        style roman; print "^^"; spaces(i);
        print "[Premi SPAZIO per iniziare.]^";
        font on;
        box "Ma l'alligatore non stava scavando il fondo della buca"
            "Che avrebbe dovuto essere la sua tomba"
            "Piuttosto era lui che stava scavando la propria buca"
            "Come riparo per se stesso"
            ""
            "-- dal Popol Vuh";
        KeyCharPrimitive();
        ClearScreen();
        ];
```

con la quale si ottiene il seguente risultato:



Siamo quindi di fronte a una sorta di "presentazione" dell'avventura. L'istruzione ClearScreen, spiegata nel capitolo 2, pulisce lo schermo. Per quanto riguarda invece gli stili[27], Inform ne mette a disposizione cinque:

che corrispondono alle seguenti istruzioni:

```
style roman;     ! (NORMALE)
style bold;      ! (GRASSETTO)
style underline; ! (CORSIVO)
style reverse;   ! (REVERSE)
style fixed;     ! (PREFORMATTATO) – equivalente a font off
```

L'istruzione font off attiva il font a spaziatura fissa, dove i singoli caratteri hanno la stessa distanza fra di loro (il Times New Roman, uno dei font che ho utilizzato per scrivere questo manuale, è ad esempio uno di quelli che non ha questa caratteristica). Detto questo, occorre capire che IN INFORM, AZIONI COME LA CENTRATURA DEL TESTO RISPETTO ALLO SCHERMO SONO POSSIBILI SOLO CON UN FONT A SPAZIATURA FISSA, E QUINDI SOLO IN MODALITÀ FONT OFF[28].
L'istruzione spaces(i) sposta il testo alla destra dello schermo di n posizioni. L'istruzione if relativa alla variabile i è un sistema piuttosto contorto per far sì che il testo appaia sempre in una specifica posizione tenendo però conto della larghezza dello schermo (che è di fatto il valore restituito alla variabile locale i dall'istruzione ScreenWidth), che varia al variare della dimensione della finestra di Windows Frotz 2002 (o di qualsiasi altro interprete Z-code). La larghezza della mia finestra di Windows Frotz 2002 è, tanto per fare un esempio, 70: questo significa che il numero massimo di posizioni consentite per un singolo carattere è 69, mentre per quanto riguarda una parola o addirittura un'intera frase, il numero massimo di posizioni consentite cambia a seconda del numero di caratteri che compongono queste ultime. A ogni modo, se il numero di posizioni supera il "limite" consentito, Inform continua la visualizzazione del testo alla riga successiva, come accade nel seguente esempio:

```
                                        [Premi SPAZIO per
iniziare.]
```

Per questa frase il limite massimo di posizioni onde evitare che una parte del testo possa andare a capo, è 43. Comunque, vi consiglio di posizionare il vostro testo nella posizione in cui vi piace, mantenendo però l'istruzione if come riferimento:

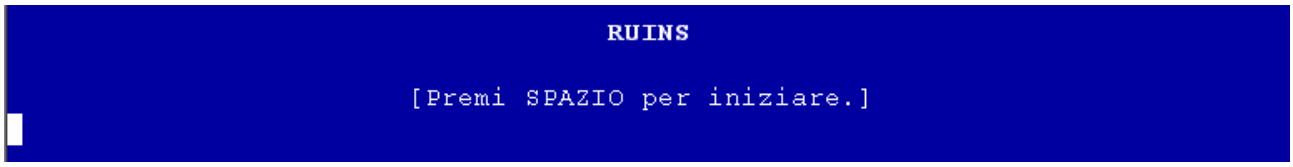
[27] Gli stili del testo possono essere gestiti più comodamente con la libreria style.h. Ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 4.9.
[28] In realtà queste azioni sono teoricamente possibili anche con i font a spaziatura variabile ma, poiché i valori delle coordinate di riferimento cambiano di volta in volta a seconda del font utilizzato, non sono fattibili da un punto di vista "pratico" perché in Inform non si può stabilire uno specifico font da utilizzare di default.

```
[ TitlePage i;
        ClearScreen(); print "^^^^^^^^^^^^^^";
        i = ScreenWidth(); if (i > 30) i = (i-30)/2;
        style bold; font off; spaces(i+10);
        print " RUINS^";
        style roman; print "^^"; spaces(i);
        print "[Premi SPAZIO per iniziare.]^";
        font on;
         .
         .
         .
```

aggiungendo ad esempio il valore 10 alla variabile locale i nella prima istruzione spaces, la scritta "RUINS" risulterà spostata un po' più a destra, ma verrà sempre presa in considerazione la larghezza dello schermo. Se la si vuole invece centrare si deve allora ricorrere all'istruzione Centre:

```
[ TitlePage;
        ClearScreen(); print "^^^^^^^^^^^^^^";
        style bold; font off;
        Centre("RUINS");
        style roman; print "^^";
        Centre("[Premi SPAZIO per iniziare.]");
        print "^";
        font on;
         .
         .
         .
```

ottenendo così il seguente risultato:



Quando invece non si vuole più usare il font a spaziatura fissa previsto da Inform, l'istruzione font on ripristina il font precedentemente usato da Windows Frotz 2002 per visualizzare il testo del gioco. L'istruzione box "..." apre un riquadro al cui interno viene visualizzato il testo racchiuso tra le virgolette[29], mentre l'istruzione KeyCharPrimitive, spiegata nel capitolo 2, aspetta la pressione di un tasto qualsiasi da parte dell'utente (o meglio, del giocatore).
Se vogliamo fare in modo che il giocatore, anziché premere un tasto qualsiasi, debba aspettare un certo periodo di tempo prima di procedere con l'avventura, si deve ricorrere all'utilizzo dell'istruzione KeyDelay. Sostituendo quindi l'istruzione KeyCharPrimitive con KeyDelay(40) il giocatore aspetterà 40 decimi di secondo; a ogni modo, se è "impaziente", potrà comunque premere un tasto qualsiasi per andare avanti.

* *

🔴 È anche possibile fare in modo che Inform stampi a video il numero di versione e il numero di serie da noi voluto:

```
Constant Story "RUINS";
```

[29] La libreria boxclever.h permette di creare dei riquadri con, oltre al testo normale, anche il grassetto e il corsivo. Ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 4.9.

```
Constant Headline
        "^Un esempio di lavoro interattivo.^
         Copyright (c) 1999 di Graham Nelson.^
         Traduzione e adattamenti di Vincenzo Scarpa e Raffaello Valesio
         (c) 2002-2003 su permesso dell'autore.^^";

Constant MAX_CARRIED = 7;
Constant MAX_SCORE = 30;

Release 2;
Serial "025207";

Include "Parser";
.
.
.
```

così come di prevedere, ad esempio, il comando AIUTO[30]:

```
[ PhotographSub...

[ AiutoSub;
  print "Sei proprio sicuro di volere un aiuto sul gioco? (s/n)^^>";
  if (yesorno())
  {
     if (player in Forest && steps.rubble_filled == true)
       "Non riesci a rimuovere le macerie? Prova a mangiare il fungo...";
     "Spiacente... nessun aiuto qui.";
  }
];

Verb 'fotografa'...
Verb 'aiuto' 'help' *                                   -> Aiuto;
```

Se il giocatore digita questo comando quando si trova nel Grande Altopiano e l'entrata della piramide è bloccata dalle macerie, Inform stampa a video il relativo consiglio, altrimenti lo informa che nessun aiuto è disponibile in quel punto del gioco:

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>aiuto
Sei proprio sicuro di volere un aiuto sul gioco? (s/n)

>s
Non riesci a rimuovere le macerie? Prova a mangiare il fungo...

[30] I suggerimenti possono essere gestiti anche attraverso le librerie flags_it.h e tutor.h. Ulteriori informazioni sul loro utilizzo potete trovarle al paragrafo 4.9.

>

Volendo essere estremamente pignoli, possiamo addirittura prevedere l'inserimento di una password:

```
[ AiutoSub password;
  print "Sei proprio sicuro di volere un aiuto sul gioco? (s/n)^^>";
  if (yesorno())
  {
     print "Inserisci la password: ";
     keyboardPrimitive(buffer, parse);
     password = parse-->1;
     if (password == 'fungo')
     {
        if (player in Forest && steps.rubble_filled == true)
            "Non riesci a rimuovere le macerie? Prova a mangiare il
             fungo...";
        "Spiacente... nessun aiuto qui.";
     }
     else "^Non sembra essere quella giusta...";
  }
];
```

Alla richiesta di aiuto, se il giocatore non digita la password corretta il suggerimento non viene visualizzato. Naturalmente, oltre alla password corretta, il giocatore si deve trovare nel Grande Altopiano e l'entrata della piramide deve essere bloccata dalle macerie:

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>aiuto
Sei proprio sicuro di volere un aiuto sul gioco? (s/n)

>s
Inserisci la password: funghi

Non sembra essere quella giusta...

>aiuto
Sei proprio sicuro di volere un aiuto sul gioco? (s/n)

>s
Inserisci la password: fungo
Non riesci a rimuovere le macerie? Prova a mangiare il fungo...

>prendi il fungo
Hai raccolto abilmente il fungo, senza staccarlo dal suo gambo sottile.

>mangia il fungo

Lo sgranocchi ad un angolo, incapace di capire l'origine di un gusto così acre, distratto dal volo di un macao sopra la tua testa che sembra quasi un'esplosione nel sole. Il battito delle sue ali è quasi assordante, e delle pietre crollano una sull'altra.

>aiuto
Sei proprio sicuro di volere un aiuto sul gioco? (s/n)

>s
Inserisci la password: fungo
Spiacente... nessun aiuto qui.

>

Il comando INFORMAZIONI:

```
[ PhotographSub...

[ AiutoSub...

[InformazioniSub;
        "Ruins è un esempio interattivo scritto da Graham Nelson per
         l'Inform Designer's Manual.^
         Traduzione in italiano a cura di Vincenzo Scarpa e Raffaello
         Valesio.^
         Per segnalazioni e contatti scrivete all'indirizzo e-mail
         scarvin@@64libero.it. ";
         ];

Verb 'fotografa'...
Verb 'aiuto' 'help'...
Verb 'informazioni' 'info' 'crediti' 'credits' *      -> Informazioni;
```

può essere utile al giocatore per sapere come contattare l'autore nel caso in cui gli voglia chiedere, ad esempio, qual è la soluzione del gioco, se è possibile reperire il listato, o altro ancora:

Dopo giorni di inutili ricerche, passati senz'acqua attraversando i rovi della foresta, alla fine la tua pazienza è stata ricompensata: hai fatto una scoperta!

RUINS
Un esempio di lavoro interattivo.
Copyright (c) 1999 di Graham Nelson.
Traduzione e adattamenti di Vincenzo Scarpa e Raffaello Valesio (c) 2002-2003 su permesso dell'autore.

Versione 1 -- Numero di serie 041109
Inform v6.30 -- Libreria 6/11 -- Infit v2.5   S

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>informazioni
Ruins è un esempio interattivo scritto da Graham Nelson per l'Inform Designer's Manual.
Traduzione in italiano a cura di Vincenzo Scarpa e Raffaello Valesio.
Per segnalazioni e contatti scrivete all'indirizzo e-mail scarvin@libero.it.

>

Un altro comando piuttosto importante da implementare è ISTRUZIONI, con lo scopo di spiegare ai non esperti come giocare a un'avventura testuale, argomento che non deve mai essere dato per scontato. Il codice è praticamente identico al comando precedente tranne, ovviamente, che per il nome.

CAPITOLO 4

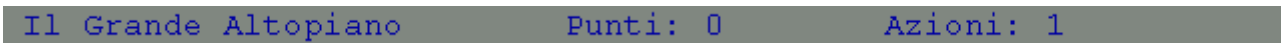
INFORM E ANCORA INFORM

§4.1 La status line

Se qualcuno di voi ha pensato che Inform si possa fermare al solo Capitolo 3, beh… inutile dire che si è sbagliato di grosso; d'altra parte, un linguaggio di programmazione non si pone dei limiti particolari (anche se qualcuno ha fatto giustamente notare che Inform non appartiene alla famiglia dei linguaggi di programmazione "general-purpouse" - con Inform, cioè, potete scrivere solo delle avventure testuali[1]).

Iniziamo subito con la status line. Con questo termine si intende la barra in alto che compare nella finestra di Windows Frotz 2002 (o in qualsiasi altro interprete Z-code) quando iniziate a giocare ad un'avventura testuale[2]:

Di default, appaiono il nome della stanza in cui si trova il giocatore e due numeri che indicano rispettivamente il punteggio e il numero di mosse che si stanno effettuando durante il gioco. Tuttavia, non è così chiaro al giocatore quale dei due numeri indica il punteggio e quale il numero di mosse[3]. È molto meglio una status line del tipo:



che si può tranquillamente ottenere grazie al seguente codice:

```
Constant Story "RUINS";
.
.
.
Constant MAX_CARRIED = 7;
Constant MAX_SCORE = 30;

Replace DrawStatusLine;

Include "Parser";

[DrawStatusLine width;
     StatusLineHeight(1);
     width = ScreenWidth(); if (width == 0) width = 80;
     MoveCursor(1, 1); spaces width;
     MoveCursor(1, 2);
     width = width - 35;
     MoveCursor(1, width); print (string) SCORE__TX, sline1;
     width = width + 15;
     MoveCursor(1, width); print (string) MOVES__TX, sline2;
     MainWindow();
```

[1] In realtà, alcuni programmatori si sono divertiti a scrivere degli "abuse", ovvero dei listati Inform che non sono però delle avventure testuali. Devono tuttavia essere considerati - soprattutto in virtù delle limitate capacità grafiche di questo linguaggio - degli esperimenti di programmazione piuttosto che dei programmi veri e propri, che meritano comunque di essere studiati (anche se parzialmente) se non altro per alcuni effetti particolarmente interessanti. Li potete scaricare, gratuitamente, all'indirizzo ftp://ftp.ifarchive.org/if-archive/games/source/inform/. Paolo Lucchesi, inoltre, ha scritto una libreria (la cyoa.h) che permette di usare Inform per creare delle avventure testuali a scelte multiple (più conosciute da alcuni come libri-game).

[2] Tutti i listati di questo capitolo si trovano nella directory Capitolo4 del file esercizi_manuale.zip.

[3] Questo inconveniente si verifica o meno a seconda della grandezza della finestra (o, se preferite, dello schermo) di Windows Frotz 2002. Di "default", infatti, la status line visualizza entrambe le scritte ma – come potete ben vedere – questo non rappresenta affatto una garanzia e di conseguenza è sempre meglio inizializzarla.

```
];
.
.
.
```

Come si può facilmente intuire, la funzione che si occupa dell'inizializzazione della status line si chiama in Inform DrawStatusLine. L'istruzione StatusLineHeight stabilisce di quante righe la status line deve essere costituita (nel nostro caso una), mentre l'istruzione MoveCursor posiziona il cursore del testo ad una posizione ben precisa della status line (visualizzata sullo schermo grazie all'istruzione spaces width;[4]) dopo aver specificato un certo numero di riga (nel nostro caso 1) e di colonna (il numero contenuto nella variabile locale width, che viene inizializzata la prima volta – tramite l'istruzione ScreenWidth() – alla lunghezza dell'intero schermo). Sottraendo poi un certo valore alla variabile locale width, spostiamo di n posizioni a sinistra il testo che viene successivamente stampato a video (il punteggio); viceversa, sommando un certo valore alla variabile omonima, spostiamo di n posizioni più a destra il testo che viene successivamente stampato a video (il numero di mosse o d'azioni). Fatto questo, segue l'istruzione MainWindow che fa in modo che il cursore punti alla finestra principale.
Tuttavia, come fa giustamente notare Paolo Lucchesi, la funzione DrawStatusLine così com'è porta una semplificazione pericolosa nel non andare a cercare il Visibility Level. Se il giocatore si trova ad esempio in una cassa chiusa, sulla status line viene mostrata solo la cassa e non il nome della stanza (che non risulta visibile al giocatore). Ecco allora come riscrivere la funzione:

```
[DrawStatusLine width;
     StatusLineHeight(1);
     width = ScreenWidth(); if (width == 0) width = 80;
     MoveCursor(1, 1); spaces width;
     MoveCursor(1, 2);
     if (location == thedark) {
         print (name) location;
     }
     else {
         FindVisibilityLevels();
         if (visibility_ceiling == location)
             print (name) location;
         else
             print (The) visibility_ceiling;
     }

     width = width - 35;
     MoveCursor(1, width); print (string) SCORE__TX, sline1;
     width = width + 15;
     MoveCursor(1, width); print (string) MOVES__TX, sline2;
     MainWindow();
];
```

Se vogliamo invece cambiare le parole Punti e Azioni rispettivamente in Punteggio e Turni, dobbiamo ricorrere all'utilizzo delle variabili di libreria score (per il punteggio) e turns (per il numero d'azioni):

```
[DrawStatusLine width;
      .
```

[4] Per creare una status line costituita da più righe (ad esempio 2) occorre specificare il valore 2 per l'istruzione StatusLineHeight e aggiungere la riga MoveCursor(1, 2); spaces width; dopo la riga MoveCursor(1, 1); spaces width;

```
      .
      .
     width = width - 35;
     MoveCursor(1, width); print "Punteggio: ", score;
     width = width + 20;
      MoveCursor(1, width); print "Turni: ", turns;
      MainWindow();
];
```

RICORDATEVI INOLTRE, CHE PER ESSERE UTILIZZATA, LA FUNZIONE DrawStatusLine DEVE ESSERE SEMPRE MESSA DOPO L'ISTRUZIONE Include "Parser", A SUA VOLTA PRECEDUTA DALL'ISTRUZIONE Replace DrawStatusLine.
E se la status line non ci piace e vogliamo eliminarla? Basta definire la funzione DrawStatusLine come segue:

```
[ DrawStatusLine; ];
```

Provare per credere…

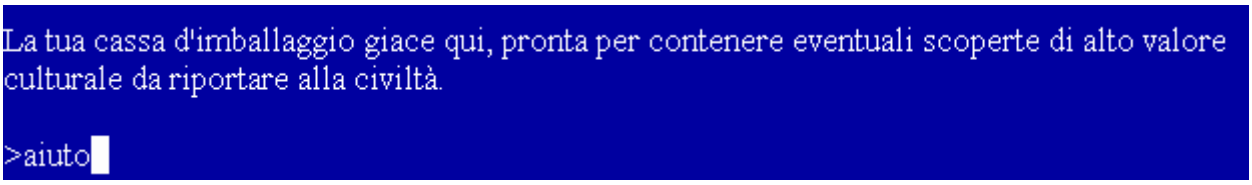
§4.2 I menu

Passiamo adesso ai menu:

```
.
.
.
[ DeathMessage...

[ HelpSub;
  DoMenu("Ecco il menu:^
         ^     Informazioni
         ^     Scopo del gioco
         ^     L'autore
         ^     Ringraziamenti^",
         HelpMenu, HelpInfo);
];

[ HelpMenu;
  if (menu_item == 0)  { item_width=2; item_name="RUINS"; return 4; }
  if (menu_item == 1)  { item_width=7; item_name="Informazioni"; }
  if (menu_item == 2)  { item_width=8; item_name="Scopo del gioco"; }
  if (menu_item == 3)  { item_width=4; item_name="L'autore"; }
  if (menu_item == 4)  { item_width=7; item_name="Ringraziamenti"; }
];

[ HelpInfo;
  if (menu_item == 1)
     {print "Sei un intrepido archeologo alla ricerca di tesori perduti.
             Indiana Jones non @`e nessuno paragonato a te.^^"; }
  if (menu_item == 2)
     {print "Riportare il numero pi@`u alto possibile di manufatti alla
             fondazione Carneige...^^"; }
  if (menu_item == 3)
     {print "Quest'avventura @`e stata scritta da Graham Nelson che
```

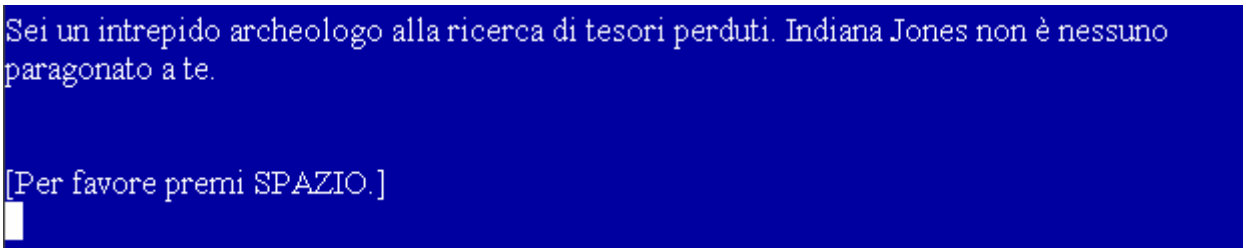
```
            ringrazier@`o in eterno per avermi dato il permesso di
            tradurla in italiano.^^"; }
  if (menu_item == 4)
     {print "Ringrazio immensamente Raffaello Valesio per l'aiuto datomi
            nella traduzione di Ruins; Massimo Pinna e Fulvio
            Ghiringhello per il betatesting.^^"; }
];
.
.
.
Include "ItalianG";
.
.
.
Verb 'fotografa' * noun                                    -> Photograph;
Verb 'aiuto' 'help' *                                      -> Help;
```

Inform dà la possibilità di usufruire d'aiuti più complessi (e completi) rispetto a quelli visti alla fine del paragrafo 3.15. Un menu è costituito da tre funzioni principali: HelpSub (che rappresenta l'interfaccia principale attraverso la quale è possibile scegliere l'opzione interessata), HelpMenu (al cui interno vengono definite le caratteristiche d'ogni singola opzione) e HelpInfo (al cui interno vengono definiti i contenuti di ogni singola opzione). Per capire meglio quanto sto dicendo, vediamolo "all'opera":



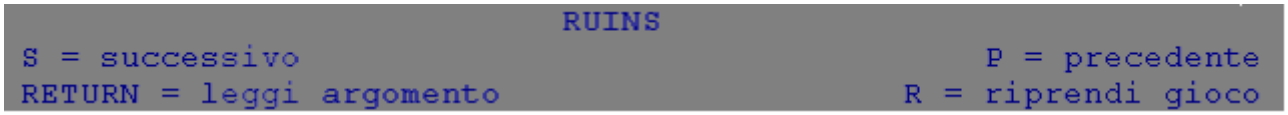
Dopo aver digitato il comando AIUTO e aver premuto il tasto di Invio, ecco quello che appare:

Questa è l'interfaccia principale. Adesso abbiamo a disposizione quattro opzioni (ma ce ne possono essere ovviamente molte di più) da selezionare ed esplorare. A livello di codice, bisogna assicurarsi che alla fine della prima riga della funzione HelpMenu venga restituito il numero di opzioni definite (nel nostro caso quattro). Una volta scelta l'opzione attraverso i tasti cursore[5] e aver premuto il tasto di Invio, è possibile leggere il suo contenuto:



[5] Per chi non lo sapesse, sono i quattro tasti presenti sulla vostra tastiera contrassegnati dalle freccette.

Premendo la barra spaziatrice si ritorna al menu principale. A questo punto è possibile continuare a selezionare delle altre opzioni o ritornare al gioco:



Ricordatevi che il valore numerico contenuto nella variabile locale item_width determina la posizione del testo contenuto nella variabile locale item_name; OCCORRE INOLTRE NOTARE, CHE IL SISTEMA DI CREAZIONE "STANDARD" DEI MENU CHE ABBIAMO APPENA VISTO È ORMAI OBSOLETO (ANCHE SE COMUNQUE PERFETTAMENTE FUNZIONANTE E PRESENTE IN DIVERSI LISTATI) E CONVIENE QUINDI RIFARSI ALL'UTILIZZO DELLA LIBRERIA DMENUS.H[6].

§4.3 I labirinti e i nomi plurali

Per quanto riguarda i labirinti, altro tema molto caro e dibattuto nell'ambito delle avventure testuali, ecco una delle possibili implementazioni all'interno di Ruins:

```
Object  Square_Chamber "La Sala Quadrata"...
.
.
.
Object  -> sunlight "raggio di sole"...

Class   Maze
  with  short_name "Labirinto",
        out_to "Sembra facile, vero?",
        has light;

Maze    Inner1_Web
  with  describe
           "Sei in un labirinto di piccoli passaggi tutti uguali.",
        w_to Inner2_Web,
        s_to Square_Chamber,
        n_to Inner1_Web, e_to Inner1_Web, nw_to Inner1_Web,
        ne_to Inner1_Web, sw_to Inner1_Web, se_to Inner1_Web;

Maze    Inner2_Web
  with  describe
            "Sei in un labirinto di stretti passaggi, tutti uguali.",
        n_to Inner3_Web,
        e_to Inner1_Web,
        w_to Inner2_Web, s_to Inner2_Web, nw_to Inner2_Web,
        sw_to Inner2_Web, ne_to Inner2_Web, se_to Inner2_Web;

Maze    Inner3_Web
  with  describe
            "Sei in un labirinto di tortuosi passaggi, tutti uguali.",
        s_to Inner2_Web, n_to Inner3_Web, w_to Inner3_Web,
        e_to Inner3_Web, nw_to Inner3_Web, sw_to Inner3_Web,
        ne_to Inner3_Web, se_to Inner3_Web;
```

[6] Ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 4.9.

La prima cosa che possiamo subito notare è che questo è un labirinto a tre livelli o, per dirla in parole più semplici, a tre stanze. Come potete però facilmente osservare, quasi tutte le direzioni puntano alle locazioni di appartenenza. Nella locazione Inner1_Web, ad esempio, il giocatore cambia stanza solo se si dirige o a sud (ritornando nella Sala Quadrata), oppure procedendo verso ovest (spostandosi alla locazione Inner2_Web, il secondo livello del labirinto). Se cerca invece di andare in tutte le altre direzioni, rimane sempre lì dov'è:

```
La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio. Un passaggio, stretto e illuminato conduce a nord.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>n

Labirinto
Sei in un labirinto di piccoli passaggi tutti uguali.

>n

Labirinto

>ne

Labirinto

>o

Labirinto
Sei in un labirinto di stretti passaggi, tutti uguali.

>
```

Questo "trucco" fa sì che il giocatore abbia l'impressione di trovarsi in una miriade di stanze (un labirinto appunto). Ma occorre anche notare un'altra cosa: come mai le due locazioni si chiamano entrambe Labirinto anziché Inner1_Web e Inner2_Web?
Il merito va attribuito all'istruzione short_name della classe Maze. In poche parole, i nomi di tutte le locazioni appartenenti a questa classe vengono stampati a video con la scritta Labirinto; tuttavia, a livello di codice restano sempre gli stessi:

```
Object  Square_Chamber "La Sala Quadrata"
  with  name 'sala' 'quadrata' 'architrave' 'architravi' 'est' 'sud'
             'entrata',
        description
            "Sei in una sala di pietra oscura e profonda, larga circa
             dieci metri. Un raggio di sole, proveniente dalla cima della
             scalinata, la illumina diffusamente, ma le ombre del livello
             pi@`u basso rivelano dei passaggi verso est e sud, che
             conducono verso la pi@`u profonda oscurit@`a del Tempio.
             Un passaggio, stretto e illuminato conduce a nord.",
        u_to Forest,
        n_to Inner1_Web,
         .
         .
         .
```

Approfittiamo di questo esempio per introdurre altre peculiarità molto interessanti di questo linguaggio:

```
Class  Maze...

Class   Bar
  with  name 'lingotto' 'lingotti//p',
        article "un",
        plural "lingotti",
        description "A prima vista sembra essere d'oro, ma osservandolo
                     meglio scopri che @`e di ottone.";

Maze    Inner1_Web...

Maze    Inner2_Web...

Maze    Inner3_Web...

Bar   "lingotto" Inner3_web;
Bar   "lingotto" Inner3_web;
```

Così facendo, se il giocatore riesce a raggiungere il terzo livello del labirinto trova due lingotti (apparentemente d'oro):

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio. A nord, una stretta e illuminata fenditura nella roccia funge da entrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>n

Labirinto
Sei in un labirinto di piccoli passaggi tutti uguali.

>o

Labirinto
Sei in un labirinto di stretti passaggi, tutti uguali.

>n

Labirinto
Sei in un labirinto di tortuosi passaggi, tutti uguali.

Puoi vedere due lingotti qui.

>prendi un lingotto
Preso.

>g

Labirinto
Sei in un labirinto di tortuosi passaggi, tutti uguali.

Puoi vedere un lingotto qui.

```
>i
Stai portando:
  un lingotto
  un dizionario maya di Waldeck
  una lampada al sodio
  la mappa di Quintana Roo

>prendi un lingotto
Preso.

>i
Stai portando:
  due lingotti
  un dizionario maya di Waldeck
  una lampada al sodio
  la mappa di Quintana Roo
```

Per Inform ogni lingotto è un oggetto a sé stante, e questo è possibile grazie all'istruzione `bar "lingotto" Inner3_web` ripetuta due volte. È chiaro allora che per avere n lingotti bisogna ripetere n volte questa istruzione. All'interno della classe `Bar`, l'unica vera novità è il nome `'lingotti//p'`, un modo per dire a Inform che l'oggetto in questione può essere chiamato al singolare (`lingotto`) o al plurale (`lingotti`):

```
Labirinto
Sei in un labirinto di tortuosi passaggi, tutti uguali.

Puoi vedere due lingotti qui.

>prendi i lingotti
lingotto: Preso.
lingotto: Preso.

>posa i lingotti
lingotto: Posato.
lingotto: Posato.

>prendi il lingotto
Preso.

>
```

In poche parole, senza quel nome il giocatore non può chiamarlo al plurale:

```
Labirinto
Sei in un labirinto di tortuosi passaggi, tutti uguali.

Puoi vedere due lingotti qui.

>prendi i lingotti
Non vedi nulla del genere.

>prendi il lingotto
Preso.

>g

Labirinto
Sei in un labirinto di tortuosi passaggi, tutti uguali.
```

Puoi vedere un lingotto qui.

>

con tutte le conseguenze del caso.

§4.4 Il tempo

Andiamo ora avanti con la gestione del tempo tramite l'orologio interno di Inform. Per attivarlo occorre inserire, all'inizio del listato, l'istruzione Statusline time:

```
Statusline time;
Constant Story "RUINS";
Constant Headline
.
.
.
```

In questo modo, il valore del tempo appare automaticamente sulla status line. Dobbiamo anche decidere a che ora farlo partire, inserendo nella funzione Initialise l'istruzione Settime secondo la seguente formula:

```
SetTime (60 x [ore] + [minuti], [avanzamento]);
```

Se vogliamo ad esempio far sì che Ruins inizi alle 8:00 del mattino, il calcolo da fare è quindi 60 x 8 per un totale di 480:

```
[ Initialise;
    TitlePage();
    location = Forest;
    Settime (480, 1);
    .
    .
    .
```

come d'altra parte la stessa statusline conferma:

Il parametro avanzamento stabilisce, come dice il nome stesso, di quanti minuti il tempo deve avanzare per ogni mossa effettuata dal giocatore (in questo esempio avanza di un minuto per ogni mossa effettuata). Volendo invece inizializzare la status line con questo parametro:

```
[DrawStatusLine width;
   StatusLineHeight(1);
   width = ScreenWidth(); if (width == 0) width = 80;
   MoveCursor(1, 1); spaces width;
   MoveCursor(1, 2);
   if (location == thedark) {
        print (name) location;
   }
   else {
          FindVisibilityLevels();
          if (visibility_ceiling == location)
              print (name) location;
          else
              print (The) visibility_ceiling;
   }

   width = width - 15;
   MoveCursor(1, width); print (string) TIME__TX;
```

```
    LanguageTimeOfDay(sline1, sline2);
    MainWindow();
];
```

ecco quello che si deve fare. Se vogliamo invece fare in modo che a una data ora si verifichi un certo evento, basta testare la variabile di libreria the_time all'ora desiderata:

```
[ Initialise;
        TitlePage();
        location = Forest;
        move map to player;
        move sodium_lamp to player;
        move dictionary to player;
        Settime (480, 1);
        Startdaemon (Timerdaemon);
         .
         .
         .
      ];

Object Timerdaemon
  with  daemon [;
           if (the_time == 485) {
             steps.rubble_filled = false;
             print "Improvvisamente, vieni distratto dal volo di un
                    macao sopra la tua testa che sembra quasi
                    un'esplosione nel sole. Il battito delle sue ali
                    @`e quasi assordante, e delle pietre crollano una
                    sull'altra.^";
            StopDaemon(self);
           }
        ];
```

Il test viene effettuato all'interno del daemon Timerdaemon. Il valore della variabile di libreria the_time è il tempo stesso che, come abbiamo già detto, s'incrementa di uno a ogni mossa effettuata dal giocatore:

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>giù
Le macerie ostruiscono il passaggio dopo appena pochi passi.

>z
Il tempo passa.

>z
Il tempo passa.

>z
Il tempo passa.

>z
Il tempo passa.
Improvvisamente, vieni distratto dal volo di un macao sopra la tua testa che sembra quasi un'esplosione nel sole. Il battito delle sue ali è quasi assordante, e delle pietre crollano una sull'altra.

>giù

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio. A nord, una stretta e illuminata fenditura nella roccia funge da entrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>

Alle 8:05 (dopo cioè cinque mosse), l'entrata della piramide sepolcrale si libera da sola grazie al daemon, indipendentemente da quello che il giocatore sta facendo. Ma non è finita: il valore del tempo può essere visualizzato anche durante il gioco, come accade nel seguente esempio:

```
Statusline time;
Constant Story "RUINS";
Constant Headline
.
.
.
Object  tiny_claws "suono di piccoli artigli" thedark...

Object orologio "orologio"
  with name 'orologio' 'ora',
       description [;
       print "Un orologio da polso a carbone che segna le ore ",
              the_time/60, ":";
       if ((the_time%60) < 10) print "0";
       print the_time%60, ".^";
       ],
has worn;

[ Initialise;
        TitlePage();
        location = Forest;
        move map to player;
        move sodium_lamp to player;
        move dictionary to player;
        move orologio to player;
        Settime (480, 1);
        Startdaemon (Timerdaemon);
         .
         .
         .
```

Tutte le volte che il giocatore esamina l'orologio, Inform stampa a video l'ora:

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>i
Stai portando:
  un orologio (indossato)
  un dizionario maya di Waldeck
  una lampada al sodio
  la mappa di Quintana Roo

>esamina l'orologio
Un orologio da polso a carbone che segna le ore 8:01.

>z
Il tempo passa.

>esamina l'orologio
Un orologio da polso a carbone che segna le ore 8:03.

L'attributo worn fa sì che l'orologio sia già indossato dal giocatore (come potete facilmente vedere dall'elenco dell'inventario). Inoltre, per evitare che l'ora venga visualizzata anche sulla statusline basta inizializzare quest'ultima eliminando le istruzioni width = width – 15, MoveCursor (1, width); print (string) TIME__TX e LanguageTimeOfDay(sline1, sline2) dalla funzione DrawStatusLine. Se invece si vuole visualizzare solo l'ora (come in Flamel di Francesco Cordella), basta eliminare l'istruzione print (string) TIME__TX lasciando inalterato tutto il resto. Se si vuole infine cambiare la parola Tempo in Ora basta sostituire l'istruzione print (string) TIME__TX con print "Ora: ".

§4.5 I colori

Per quanto riguarda la gestione dei colori, l'istruzione SetColour per la selezione del colore del testo e dello sfondo richiede, nel suo utilizzo, certe attenzioni particolari. IL PRIMO "CAMBIO DI COLORE" DEVE AVVENIRE ALL'INTERNO DELLA FUNZIONE Initialise, anche se in effetti questa regola non è esclusiva poiché quest'istruzione può essere usata in un punto qualsiasi del listato. Tuttavia, per essere completamente sicuri che avvenga il cambio di colore almeno una volta, è meglio seguire la regola sopra citata:

```
[ Initialise;
        print "Vuoi supportare i colori (s/n)? ";
        if (yesorno()) clr_on = 1;
         else clr_on = 0;

        SetColour(9,2); ClearScreen();
        TitlePage();
        location = Forest;
         .
         .
```

.

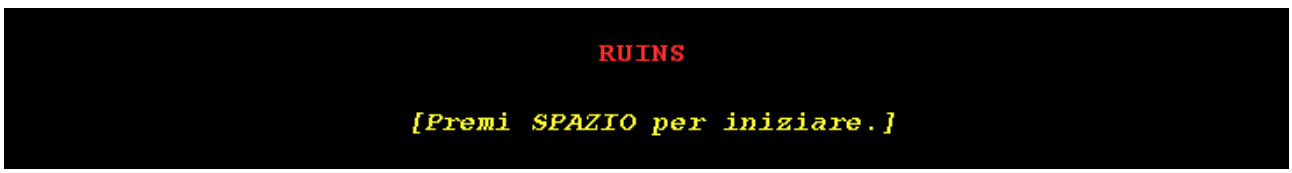
AFFINCHÉ L'ISTRUZIONE SETCOLOUR ABBIA EFFETTO, OCCORRE IMPOSTARE A 1 LA VARIABILE DI LIBRERIA clr_on. Dal momento però, che non è detto che tutti gli interpreti supportino correttamente i colori (non bisogna mai dimenticare che Inform è linguaggio multipiattaforma - palmari compresi), e meglio far sì che Inform chieda prima al giocatore se desidera o no il loro supporto all'inizio del gioco; se quest'ultimo risponde alla domanda premendo i tasti s e INVIO, i colori vengono attivati (la variabile di libreria clr_on viene cioè posta uguale a 1), altrimenti (se il giocatore preme cioè i tasti n e INVIO) rimangono attivi quelli di default (la variabile di libreria clr_on vale 0 e vengono di conseguenza ignorate da Inform tutte le istruzioni relative ad essi)[7].
IN INFORM, INOLTRE, A OGNI COLORE CORRISPONDONO UN NUMERO E UNA COSTANTE, come mostrato nella seguente tabella:

DEFAULT	1	CLR_DEFAULT
NERO	2	CLR_BLACK
ROSSO	3	CLR_RED
VERDE	4	CLR_GREEN
GIALLO	5	CLR_YELLOW
BLU	6	CLR_BLUE
MAGENTA	7	CLR_MAGENTA
CIANO	8	CLR_CYAN
BIANCO	9	CLR_WHITE
VIOLA	7	CLR_PURPLE
AZZURRO	8	CLR_AZURE

I due valori assegnati all'istruzione SetColour rappresentano rispettivamente il colore del testo (il 9 o CLR_WHITE) e dello sfondo (il 2 o CLR_BLACK); non è difficile, a questo punto, capire che il numero nove rappresenta il colore bianco e il numero due il colore nero. Volendo poi cambiare il colore di una o più parole, ecco cosa si deve fare:

```
[ TitlePage;
        ClearScreen(); print "^^^^^^^^^^^^^^";
        style bold; font off;
        SetColour(3,2); Centre("RUINS");
        style underline; print "^^";
        SetColour(5,2); Centre("[Premi SPAZIO per iniziare.]");
        SetColour(9,2); print "^^";
        font on;
        box "Ma l'alligatore non stava scavando il fondo della buca"
            "Che avrebbe dovuto essere la sua tomba"
            "Piuttosto era lui che stava scavando la propria buca"
            "Come riparo per se stesso"
            ""
            "-- dal Popol Vuh";
        KeyCharPrimitive();
        ClearScreen();
        ];
```

ottenendo così il seguente risultato:

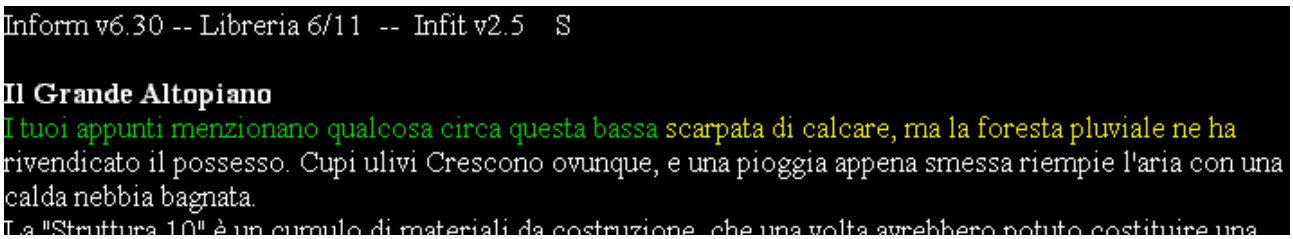


[7] L'istruzione yesorno risponde solo alle lettere s (o si) e n (o no). Se si vogliono usare anche le altre, occorre rifarsi all'istruzione ReadChar, da me creata e descritta nel paragrafo 2.15.

Per cambiare il colore del testo della descrizione di un oggetto occorre modificare la proprietà description nel seguente modo:

```
Object  Forest "Il Grande Altopiano"
  with  description [;
            SetColour(4,2,2); print "I tuoi appunti menzionano qualcosa
               circa questa bassa ";
            SetColour(5,2,2); print "scarpata di calcare, ma la foresta
              pluviale ne ha ";
            SetColour(9,2,2); print "rivendicato il possesso. Cupi ulivi
              Crescono ovunque, e una pioggia appena smessa riempie
              l'aria con una calda nebbia bagnata.^
              La ~Struttura 10~ @`e un cumulo di materiali da
              costruzione, che una volta avrebbero potuto costituire una
              piramide sepolcrale, ma della quale ora nulla rimane,
              eccetto alcuni gradini scolpiti nella nuda roccia che
              portano gi@`u, nell'oscurit@`a.";
             ],
        d_to steps,
        in_to steps,
        .
        .
        .
```

Il risultato potete vederlo nella figura che segue:



Partendo dal presupposto che i colori di default siano il bianco per il testo e il nero per lo sfondo, all'interno della proprietà description dell'oggetto interessato si richiama una o più volte l'istruzione SetColour mantenendo il colore dello sfondo di default (2) e cambiando invece il colore del testo (in questo caso prima 4 e poi 5), ripristinando poi il tutto al bianco e al nero (evitando così che il testo che segue resti in giallo). QUANDO SI LAVORA CON I COLORI, OCCORRE QUINDI DEFINIRE **SEMPRE** QUELLI DI DEFAULT ALL'INTERNO DELLA FUNZIONE Initialise. Il terzo numero (il 2), presente in entrambe le istruzioni, dice a Inform che il cambio di colore deve riguardare solo la finestra principale. Usando invece il numero 1, il cambio di colore avviene solo nella status line; se invece non si mette nulla, il cambio di colore avviene per entrambe (la status line e la finestra principale).

Un altro utilizzo abbastanza comune dei colori nelle avventure testuali, è quello di cambiare il colore dello sfondo e del testo al cambio di locazione o di sezione. Ecco allora una possibile implementazione:

```
Object  Square_Chamber "La Sala Quadrata"
  with  name...
        description...
        s_to [;
               Changecolor(Corridor);
               return Corridor;
        ],
```

```
        .
        .
        .
```

Se il giocatore si trova nella Sala Quadrata e si dirige verso sud, viene effettuata la chiamata alla funzione Changecolor che cambia il colore del testo da bianco a giallo e il colore dello sfondo da nero a blu:

```
.
.
.
[ DeathMessage...

[Changecolor object;
        if (object == Corridor) SetColour(CLR_YELLOW,CLR_BLUE);
        if (object == Square_Chamber) SetColour(CLR_WHITE,CLR_BLACK);
        if (clr_on == 1) ClearScreen();
];
```

Quando il giocatore torna indietro verso la Sala Quadrata, viene effettuata una seconda chiamata alla funzione Changecolor che ripristina i colori di default:

```
Object  Corridor "Corridoio in pendenza"
  with  description
            "Un corridoio basso e squadrato va da nord verso sud,
             inclinandosi verso la fine.",
        n_to [;
               Changecolor(Square_Chamber);
               return Square_Chamber;
        ],
        s_to StoneDoor;
```

L'istruzione ClearScreen, si rende necessaria nella funzione Changecolor perché viene cambiato anche il colore dello sfondo (a condizione, ovviamente, che i colori siano attivi). Se dobbiamo invece cambiare solo il colore del testo, occorre allora riscrivere la funzione Changecolor nel seguente modo:

```
[Changecolor object;
        if (object == Corridor) SetColour(CLR_YELLOW,CLR_BLACK,2);
        if (object == Square_Chamber) SetColour(CLR_WHITE,CLR_BLACK,2);
];
```

Eliminando l'istruzione ClearScreen e mantenendo il nero come colore di sfondo, il problema è risolto; provare per credere…

§4.6 I mezzi di trasporto

Lasciamo ora da parte Ruins per esaminare l'esempio che segue:

MONTACARICHI
di Vincenzo Scarpa
Esempio di spostamento sopra un altro oggetto

Versione 1 -- Numero di serie 050425

```
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

Basamento Ovest
Direzioni: sud, est

Puoi vedere un vecchio montacarichi spento qui.

>esamina il montacarichi
È tutto arrugginito.

>
```

Tutto quello che per ora riusciamo a capire, è che siamo in una stanza con un vecchio montacarichi a nostra disposizione. Proviamo allora a salirci sopra e ad accenderlo:

```
>sali sul montacarichi
Ti trovi sopra il montacarichi.

>accendi il montacarichi
Dopo un frastuono assordante, il montacarichi non si mette in moto. Forse dovresti riprovare...

>ancora
Dopo un frastuono assordante, il montacarichi non si mette in moto. Forse dovresti riprovare...

>ancora
Dopo un frastuono assordante, il montacarichi si mette in moto.

>g

Basamento Ovest (sopra il montacarichi)
Direzioni: sud, est

>
```

Dopo qualche tentativo, riusciamo nel nostro intento. Ora proviamo a spostarci:

```
>e
Il motore sbuffa e singhiozza…

Basamento Est (sopra il montacarichi)
Direzioni: ovest

Ci sono delle scale poco illuminate qui, che portano al piano superiore.

>su
Hai mai visto un montacarichi salire su una scala?

>
```

Come avete appena visto, in Inform ci si può spostare di stanza in stanza anche attraverso un mezzo di trasporto; inutile dire quanto questo possa essere di estrema utilità (provate, ad esempio, a immaginare una stanza piena di casse che nascondono un passaggio). Naturalmente, come tutti i mezzi di trasporto che si rispettino, è anche possibile spegnerlo:

```
>spegni il montacarichi
Il montacarichi si è spento. Non sei sicuro che riesca a mettersi di nuovo in moto.

>o
Non puoi andare da nessuna parte se il montacarichi è spento.
```

>

rendendo ovviamente impossibile qualsiasi ulteriore spostamento con esso (occorre accenderlo di nuovo, con tutte le difficoltà del caso). A questo punto, possiamo allora scendere per proseguire "a piedi":

>scendi dal montacarichi
Sei sceso dal montacarichi.

Basamento Est
Direzioni: ovest

Puoi vedere un vecchio montacarichi spento qui.

Ci sono delle scale poco illuminate qui, che portano al piano superiore.

>o

Basamento Ovest
Direzioni: sud, est

>

Interessante, vero? Vediamo allora la descrizione del montacarichi:

```
Object forklift "montacarichi" Aerobsm
  with name 'montacarichi' 'sollevatore',
       describe [;
          print "^Puoi vedere un vecchio montacarichi ";
          if (self has on) "acceso qui.";
          "spento qui.";
           ],
       before [;
          Climb: <<Enter self>>;
          Examine: print_ret "@`E tutto arrugginito.";
          Go: if (self has on) {
                   if ((noun == u_obj) && (location == Bse)) {
                       print "Hai mai visto un montacarichi salire su una
                              scala?^";
                       return 2; ! Non si muove
                   }
                   else {
                      print "Il motore sbuffa e singhiozza...^";
                      return 1; ! Vai pure
                   }
              }
              else {
                 print "Non puoi andare da nessuna parte se il
                        montacarichi @`e spento.^";
                 return 2;
              }
          SwitchOn: if (player in self) {
                       if (self hasnt on) {
                          if (random(4)==2) {
                              give self on;
                              print "Dopo un frastuono assordante, il
                                     montacarichi si mette in moto.^";
```

```
                        }
                        else print "Dopo un frastuono assordante, il
                                    montacarichi non si mette  in
                                    moto. Forse dovresti
                                    riprovare...^";
                    }
                    else rfalse;
                }
                else print "Senza salirci sopra? Forse sei un
                            telepate...^";
                rtrue;
        SwitchOff: if (player in self) {
                        if (self has on) {
                            give self ~on;
                            print "Il montacarichi si @`e spento. Non
                                   sei sicuro che riesca a mettersi di
                                   nuovo in moto.^";
                        }
                        else rfalse;
                    }
                    else print "Prima devi salirci sopra.^";
                    rtrue;
        ],
has supporter static enterable switchable ~on;
```

Le uniche novità, rispetto a quanto visto finora, sono i valori 1 e 2 che vengono ritornati col verbo Go. Il primo indica a Inform che il montacarichi può muoversi (nel nostro caso solo quando è acceso) mentre il secondo lo ferma (nel nostro caso quando è spento o se vogliamo salire sulle scale). L'istruzione random genera invece un numero casuale (in questo caso da 1 a 4) per rendere più "difficoltosa" l'accensione del mezzo.

§4.7 Su e giù con l'ascensore

La gestione di un ascensore è un altro "classico" delle avventure testuali. Ecco allora come funziona l'esempio da me proposto:

```
Ascensore
Un semplice esempio che gestisce un ascensore a quattro piani
di Vincenzo Scarpa

Versione 1 -- Numero di serie 050704
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

Piano terra
Sei al piano terra.

Puoi vedere delle porte scorrevoli qui.

>
```

All'inizio abbiamo davanti a noi delle porte scorrevoli. Naturalmente, occorre aprirle per entrare poi nella cabina:

```
>esamina le porte
C'è un pulsante accanto alle porte.
```

```
>premi pulsante
Le porte si aprono silenziosamente.

>entra

Cabina dell'ascensore
Sei nella cabina dell'ascensore.

Puoi vedere un pannello di controllo con quattro tasti numerati da zero a tre.

>
```

Eccoci dunque dentro la cabina del nostro ascensore. Se non soffrite di claustrofobia, potete allora decidere a quale piano andare premendo l'apposito pulsante:

```
>premi 2
Le porte si chiudono e la cabina si mette in movimento.
Improvvisamente, le porte si aprono di nuovo.

>esci
Non appena esci dall'ascensore, le porte scorrevoli si chiudono silenziosamente alle tue spalle.

Secondo piano
Sei al secondo piano.

Puoi vedere delle porte scorrevoli qui.

>
```

Una volta arrivati a destinazione, potete uscire dalla cabina quando volete, ricordandovi però di riaprire le porte se dovete usarla di nuovo:

```
>entra
Le porte sono ancora chiuse: credi forse di essere un fantasma?

>premi il pulsante
Le porte si aprono silenziosamente.

>entra

Cabina dell'ascensore
Sei nella cabina dell'ascensore.

Puoi vedere un pannello di controllo con quattro tasti numerati da zero a tre.

>
```

A questo punto, il suo utilizzo dovrebbe ormai essere abbastanza chiaro. Occorre però ancora notare che, se il giocatore preme il tasto corrispondente al piano su cui si trova l'ascensore, non accade nulla:

```
>premi 2
Non accade nulla.

>
```

come d'altra parte avviene in tutti gli ascensori che si rispettino. Per quanto riguarda invece il codice, occorre fare alcune considerazioni. Iniziamo dalla funzione Muovi_cabina:

```
[Muovi_cabina x y;
    Ascensore.piano_asc = x;
    if (parent(Porte_scorrevoli) ~= y) {
        move Porte_scorrevoli to y;
        move Pulsante_chiamata to y;
        print_ret "Le porte si chiudono e la cabina si mette in
                   movimento. ^Improvvisamente, le porte si aprono di
                   nuovo.";
        }
    else print_ret "Non accade nulla.";
];
```

I parametri che devono esserle passate sono rispettivamente il numero del piano che l'ascensore deve raggiungere (x) e il nome dell'oggetto al quale corrisponde il piano stesso (y). È facile allora capire che una chiamata del tipo:

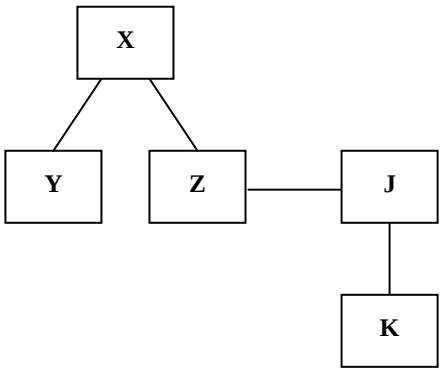
```
Button -> Pulsante_2 "terzo pulsante"
  with name 'terzo' 'due' '2//',
       before
       [;
          Push:   Muovi_cabina(2, Piano_2);
                  rtrue;
       ];
```

fa sì che, se il giocatore preme il pulsante due, viene effettuata una chiamata alla funzione Muovi_cabina con, come parametri, 2 per la variabile locale x e Piano_2 per la variabile locale y. A questo punto, viene dapprima assegnata alla variabile locale piano_asc il valore contenuto in x, utile all'oggetto Ascensore per sapere a quale stanza deve indirizzare il giocatore quando questi esce dalla cabina:

```
Object Ascensore "Cabina dell'ascensore"
  with description "Sei nella cabina dell'ascensore.",
       out_to [;
          <<Go n_obj>>;
       ],
       n_to [;
          give Porte_scorrevoli ~open;
          print "Non appena esci dall'ascensore, le porte scorrevoli si
                 chiudono silenziosamente alle tue spalle.^";
          if (self.piano_asc == 0) return Piano_terra;
          if (self.piano_asc == 1) return Piano_1;
          if (self.piano_asc == 2) return Piano_2;
          if (self.piano_asc == 3) return Piano_3;
       ],
       piano_asc 0;
```

Segue poi, una condizione che verifica se il nome dell'oggetto al quale appartiene Porte_scorrevoli è o meno uguale al nome dell'oggetto passato come parametro alla funzione stessa. Se lo è, allora non accade nulla (il giocatore ha cioè premuto il tasto corrispondente al piano stesso in cui si trova la cabina dell'ascensore), altrimenti vengono spostati gli oggetti Porte_scorrevoli e Pulsante_chiamata al relativo piano.

Per capire meglio quanto ho appena detto, occorre rifarsi al concetto di albero; Inform, cioè, raggruppa gli oggetti con uno schema di questo tipo:



dove gli oggetti Y e Z appartengono all'oggetto X (e sono di conseguenza i child di X), mentre quest'ultimo è il parent di Y e Z. L'oggetto J è il sibling di Z (e non appartiene a X) ma anche parent di K e così via.
Sembrerebbe tutto a posto, ma c'è un'ultima considerazione da fare. Definendo il primo pulsante come segue:

```
Button -> Pulsante_1 "secondo pulsante"
  with name 'secondo' 'uno' '1//',
       before
       [;
          Push:   Muovi_cabina(1, Piano_1);
                  rtrue;
       ];
```

tutto sembra essere a posto. Tuttavia, quando il giocatore preme il pulsante 1:

```
Cabina dell'ascensore
Sei nella cabina dell'ascensore.

Puoi vedere un pannello di controllo con tre tasti numerati come zero, uno e due.

>premi 1
(il pannello di controllo)
È fisso al suo posto.

>
```

Inform assegna il nome 1 al pannello di controllo anziché all'oggetto Pulsante_1. Questo bug, che non so se imputare a Inform stesso o a Infit, è possibile evitarlo con un po' d'astuzia. Iniziamo a esaminare l'oggetto Pannello_controllo:

```
Object -> Pannello_controllo "pannello di controllo"
  with name 'pannello' 'controllo',
       description "Lo osservi attentamente, ma non noti nulla di
                    speciale.",
       initial "Puoi vedere un pannello di controllo con quattro tasti
                numerati da zero a tre.",
   has static;
```

La prima modifica che occorre fare è l'inserimento dell'azione Push (che in questo caso deve essere analoga a quella dell'oggetto Pulsante_1):

```
Object -> Pannello_controllo "pannello di controllo"
  with name 'pannello' 'controllo',
       description "Lo osservi attentamente, ma non noti nulla di
                    speciale.",
       initial "Puoi vedere un pannello di controllo con quattro tasti
                numerati da zero a tre.",
       before
       [;
          Push:    Muovi_cabina(1, Piano_1);
                   rtrue;
       ],
   has static;
```

Agendo in questo modo, l’ascensore si può spostare al primo piano:

```
Cabina dell'ascensore
Sei nella cabina dell'ascensore.

Puoi vedere un pannello di controllo con quattro tasti numerati da zero a tre.

>premi 1
(il pannello di controllo)
Le porte si chiudono e la cabina si mette in movimento.
Improvvisamente, le porte si aprono di nuovo.

>
```

Come potete però vedere, prima che l’ascensore si sposti viene anche stampato il nome dell’oggetto, che non è ovviamente il pannello di controllo. Occorre allora fare una seconda modifica:

```
Object -> Pannello_controllo "secondo pulsante"
  with name 'pannello' 'controllo',
       description "Lo osservi attentamente, ma non noti nulla di
                    speciale.",
       initial "Puoi vedere un pannello di controllo con quattro tasti
                numerati da zero a tre.",
       before
       [;
          Push:    Muovi_cabina(1, Piano_1);
                   rtrue;
       ],
   has static;
```

Cambiando il nome dell’oggetto come secondo pulsante, abbiamo risolto anche il secondo problema:

```
Cabina dell'ascensore
Sei nella cabina dell'ascensore.

Puoi vedere un pannello di controllo con quattro tasti numerati da zero a tre.

>premi 1
(il secondo pulsante)
Le porte si chiudono e la cabina si mette in movimento.
Improvvisamente, le porte si aprono di nuovo.
```

>

Rimane tuttavia un altro problema; se il giocatore prova a premere il pannello di controllo:

Cabina dell'ascensore
Sei nella cabina dell'ascensore.

Puoi vedere un pannello di controllo con quattro tasti numerati da zero a tre.

>premi pannello
Le porte si chiudono e la cabina si mette in movimento.
Improvvisamente, le porte si aprono di nuovo.

>

Inform continua a eseguire il codice relativo all'azione Push associata (la stessa, cioè, dell'oggetto Pulsante_1). Per risolvere questo terzo problema, bisogna ricorrere a una soluzione drastica:

```
Object -> Pannello_controllo "secondo pulsante"
  with name 'pannello' 'controllo',
       initial "Puoi vedere un pannello di controllo con quattro tasti
                numerati da zero a tre.",
       parse_name [;
           if (NextWord() == 'pannello' or 'controllo') return 0;
       ],
       before
       [;
          Push:    Muovi_cabina(1, Piano_1);
                   rtrue;
       ],
   has static;
```

Utilizzando la proprietà parse_name in questo modo, "obbligo" Inform a ignorare totalmente i nomi "pannello" e "controllo":

Cabina dell'ascensore
Sei nella cabina dell'ascensore.

Puoi vedere un pannello di controllo con quattro tasti numerati da zero a tre.

>premi il pannello
Non vedi nulla del genere.

>premi 1
(il secondo pulsante)
Le porte si chiudono e la cabina si mette in movimento.
Improvvisamente, le porte si aprono di nuovo.

>esci
Non appena esci dall'ascensore, le porte scorrevoli si chiudono silenziosamente alle tue spalle.

Primo piano
Sei al primo piano.

Puoi vedere delle porte scorrevoli qui.

>

Certo, la risposta di sistema non è delle migliori (il pannello di controllo infatti esiste, nonostante Inform affermi il contrario), ma d'altra parte non si può sempre avere tutto.

§4.8 Gli NPC (o PNG in italiano)

Sostanzialmente, i Non Player Characters o Personaggi Non Giocatori (da qui in poi PNG) altro non sono che tutti quei personaggi inclusi nel gioco all'infuori del giocatore stesso (il sacerdote mummificato in Ruins, tanto per fare un esempio, è uno di questi). Osserviamo ora il funzionamento del seguente esempio:

```
La signora Bindi
di Vincenzo Scarpa e Paolo Lucchesi
Avventura d'esempio per gestire un NPC.

Versione 1 -- Numero di serie 051016
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

Corridoio
Sei in un lungo corridoio che prosegue verso nord.

>n

Corridoio
Sei in un lungo corridoio. A ovest c'è una porta aperta.

>o

Soggiorno
Questa è una stanza molto grande. Alla tua sinistra, una finestra dà sulla strada e davanti a te puoi vedere
un grosso tavolo in noce circondato ai lati da quattro sedie.

C'è la signora Bindi qui, seduta sul sofà.

Puoi anche vedere un giornale qui.

>esamina la signora Bindi
È molto bella.

>bacia la signora Bindi
Non credi che la signora Bindi sia interessata alle tue attenzioni.

>chiedi alla signora Bindi del giornale
"È il giornale di oggi."
```

La signora Bindi è il nostro PNG per eccellenza, poiché reagisce ai comandi "sociali" (grazie alla proprietà life che in un oggetto di questo tipo deve essere sempre presente). Ma non è finita:

```
>z
Il tempo passa.

>z
Il tempo passa.

>z
Il tempo passa.

>z
```

Il tempo passa.

>z
Il tempo passa.

La signora Bindi si dirige verso est.

>

dopo altri cinque turni, la signora Bindi esce dal soggiorno e si dirige verso est. Proviamo allora a seguirla:

>e

Corridoio

Puoi vedere la signora Bindi qui.

La signora Bindi si dirige verso nord.

>n

Corridoio
Sei in un lungo corridoio. Verso nord puoi vedere una porta aperta che dà sulla cucina.

Puoi vedere la signora Bindi qui.

La signora Bindi si dirige verso nord.

>n

Cucina
Questa è la cucina della signora Bindi, piuttosto stretta ma completa.

La signora Bindi è qui, intenta a fare le pulizie. Non sembra essere molto contenta.

Puoi anche vedere un frigorifero (che è chiuso) e una forchetta qui.

>esamina il frigo
È tutto sporco di sugo di pomodoro.

>chiedi alla signora Bindi del pomodoro
"Dovrei fare più attenzione a quello che faccio. A volte sono così distratta..."

>

In poche parole, dopo dieci turni dall'inizio del gioco la signora Bindi si sposta dal soggiorno in cucina, indipendentemente da quello che il giocatore sta facendo. E questo accade naturalmente anche se quest'ultimo non si trova nel soggiorno:

Soggiorno
Questa è una stanza molto grande. Alla tua sinistra, una finestra dà sulla strada e davanti a te puoi vedere un grosso tavolo in noce circondato ai lati da quattro sedie.

C'è la signora Bindi qui, seduta sul sofà.

Puoi anche vedere un giornale qui.

>prendi il giornale
Preso.

```
>e

Corridoio

>n

Corridoio
Sei in un lungo corridoio. Verso nord puoi vedere una porta aperta che dà sulla cucina.

>leggi il giornale
Nessuna notizia interessante.

>z
Il tempo passa.

>z
Il tempo passa.

>z
Il tempo passa.

>z
Il tempo passa.

>z
Il tempo passa.

La signora Bindi arriva qui.

>parla alla signora Bindi del giornale
"È il giornale di oggi."

La signora Bindi si dirige verso nord.

>
```

Interessante, vero? Ma non è ancora finita. Guardate cosa accade se ci troviamo in cucina quando la signora Bindi arriva:

```
Cucina
Questa è la cucina della signora Bindi, piuttosto stretta ma completa.

Puoi vedere un frigorifero (che è chiuso) e una forchetta qui.

>esamina il frigorifero
Sembra essere piuttosto robusto.

>apri il frigorifero
Ora hai aperto il frigorifero.

>g

Cucina
Questa è la cucina della signora Bindi, piuttosto stretta ma completa.

Puoi vedere un frigorifero (che è vuoto) e una forchetta qui.

>esamina la forchetta
Una normale forchetta.
```

```
>prendi la forchetta
Presa.

>i
Stai portando:
  una forchetta

>z
Il tempo passa.

>z
Il tempo passa.

>z
Il tempo passa.

La signora Bindi arriva qui.

La signora Bindi chiude il frigorifero sporcandolo con le mani imbrattate di sugo di pomodoro.

>g

Cucina
Questa è la cucina della signora Bindi, piuttosto stretta ma completa.

La signora Bindi è qui, intenta a fare le pulizie. Non sembra essere molto contenta.

Puoi anche vedere un frigorifero (che è chiuso) qui.

>esamina il frigorifero
È tutto sporco di sugo di pomodoro.

>
```

Prima del suo arrivo, il frigorifero in questione è pulito e chiuso. Noi lo apriamo, e quando lei arriva in cucina, lo chiude e lo sporca. Un PNG può quindi modificare anche lo stato di uno o più oggetti che non devono, tra l'altro, trovarsi necessariamente nella stanza in cui esso si trova (grazie all'istruzione objectloop spiegata verso la fine del paragrafo 3.10). Senza contare che, tramite l'istruzione move, può addirittura prenderli e spostarli (si potrebbe, per assurdo, far sì che al suo arrivo in cucina, la signora Bindi prendesse - se presente - la forchetta e la mettesse nel frigorifero). A ogni modo, al ventiduesimo turno il nostro PNG esce dalla cucina ritornando nuovamente al soggiorno:

```
Cucina
Questa è la cucina della signora Bindi, piuttosto stretta ma completa.

La signora Bindi è qui, intenta a fare le pulizie. Non sembra essere molto contenta.

Puoi anche vedere un frigorifero (che è chiuso) e una forchetta qui.

La signora Bindi si dirige verso sud.

>esamina il frigorifero
Sembra essere piuttosto robusto.

>s

Corridoio
```

```
>s

Corridoio

>o

Soggiorno
Questa è una stanza molto grande. Alla tua sinistra, una finestra dà sulla strada e davanti a te puoi vedere
un grosso tavolo in noce circondato ai lati da quattro sedie.

La signora Bindi è qui, intenta a guardare la televisione.

Puoi anche vedere un giornale qui.

>chiedi alla signora Bindi del frigorifero
È di nuovo pulito ora.

>
```

Il frigorifero adesso è di nuovo pulito e sono tutti felici e contenti[8].
Diamo ora un'occhiata al codice (molto meno difficile di quello che a prima vista potrebbe forse sembrare):

```
[ Initialise;
    location = Corridoio_sud;
    move Bindi to Soggiorno;
    Startdaemon (Timerdaemon);
];
```

All'inizio, diamo la locazione di partenza del gioco, la posizione iniziale del nostro PNG e facciamo partire il daemon Timerdaemon:

```
Object Timerdaemon
    with tm 0,
         daemon [;
           switch (++self.tm) {
              10: Bindi.path = 1; Startdaemon(Bindi);
              22: Bindi.path = 2; Startdaemon(Bindi);
              38: StopDaemon(self); deadflag = 3;
           }
    ];
```

quest'ultimo, conta i turni ed esegue certe azioni in turni ben precisi. In particolare, al decimo turno attiva il daemon della signora Bindi sul primo percorso (alla decima mossa cioè del giocatore, la signora Bindi si dirige dal salotto alla cucina), mentre al ventiduesimo turno essa si dirige dalla cucina nuovamente al soggiorno. Al trentottesimo turno, poi, viene fermato il gioco. Per quanto riguarda invece la signora Bindi:

```
Object Bindi "signora Bindi"
  with name 'signora' 'bindi' 'Bindi',
       article "la",
       describe [;
```

[8] La creazione di movimenti relativamente sofisticati per i personaggi non giocatori può essere semplificata con l'utilizzo della libreria MoveClass.h. Ulteriori informazioni sul suo utilizzo potete trovarle nel prossimo paragrafo.

```
   switch (location) {
     Soggiorno: if (self.path < 2) "^C'@`e la signora Bindi qui,
                   seduta sul sof@`a.";
                "^La signora Bindi @`e qui, intenta a guardare la
                 televisione.";
     Cucina: "^La signora Bindi @`e qui, intenta a fare le
              pulizie. Non sembra essere molto contenta.";
   }
],
description "@`E molto bella.",
life [;
  Attack, ThrowAt: "Non oseresti mai far del male ad una povera
        signora indifesa.";
  Kiss: "Non credi che la signora Bindi sia interessata alle tue
        attenzioni.";
  Show: "~@`E ", (a) noun, "~, dice la signora Bindi.";
  Give: "~No, grazie, ", (the) noun, " non mi serve.~";
  Ask, Tell: switch(second) {
        'forchetta': "~@`E l'unica rimasta di tutto il servito.~";
        'giornale': "~@`E il giornale di oggi.~";
        'frigo', 'frigorifero', 'sugo', 'pomodoro':
               if ((TimerDaemon.tm >=12) && (self.path < 2))
                  "~Dovrei fare pi@`u attenzione a quello che
                   faccio. A volte sono cos@`i distratta...~";
               if (self.path >= 2) "~@`E di nuovo pulito ora.~";
       }
       "~Non ne so niente, mi spiace.~";
],
path 0,
daemon [;
   switch (self.path) {
      1: switch(parent(self)) {
            Soggiorno: percorso_npc(self, e_obj); rtrue;
            Corridoio_nord: percorso_npc(self, n_obj); rtrue;
            Corridoio_cucina: percorso_npc(self, n_obj);
        ! se il giocatore è in cucina e il frigorifero è aperto...
        if ((player in Cucina) && (frigorifero has open))
             print "^La signora Bindi chiude il frigorifero
                    sporcandolo con le mani imbrattate di sugo di
                    pomodoro.^";
        ! se il giocatore è in cucina e il frigorifero è chiuso...
        if ((player in Cucina) && (frigorifero hasnt open))
             print "^La signora Bindi apre il frigorifero
                   (sporcandolo con le mani imbrattate di sugo di
                   pomodoro) e, dopo un attimo di esitazione, lo
                   richiude .^";
              give frigorifero ~open;
              StopDaemon(self); rtrue;
        }
     2: switch(parent(self)) {
           Cucina: percorso_npc(self, s_obj); rtrue;
           Corridoio_cucina: percorso_npc(self, s_obj); rtrue;
           Corridoio_nord: percorso_npc(self, w_obj);
          StopDaemon(self); rtrue;
        }
     }
],
```

```
  has female animate;
```

la proprietà describe permette a Inform di stampare a video delle frasi personalizzate a seconda di dove il nostro PNG si trova. La proprietà life, come abbiamo già detto, stabilisce come esso reagisce ai comandi sociali; in particolare, occorre notare le diverse risposte che esso dà quando il giocatore gli chiede delle informazioni inerenti al frigorifero: se, infatti, la signora Bindi non è ancora uscita dalla cucina viene stampato a video il messaggio “"Dovrei fare più attenzione a quello che faccio. A volte sono così distratta..."”, mentre se è invece uscita viene stampato a video il messaggio “"È di nuovo pulito ora."”. La proprietà daemon gestisce invece il movimento del PNG dove, a seconda del valore della variabile path e della locazione in cui esso si trova, viene chiamata la funzione percorso_npc alla quale occorre passare come parametri l’oggetto stesso (self) e la direzione verso la quale il PNG deve dirigersi. Per quanto riguarda, infine, il frigorifero:

```
Object -> frigorifero "frigorifero"
  with name 'frigorifero' 'frigo',
       description [;
      ! Se la signora Bindi non è ancora in cucina o se ne è già
      ! andata...
      if ((TimerDaemon.tm < 12) || (Bindi.path >=2)) print "Sembra essere
           piuttosto robusto.^";
      ! Se la signora Bindi si trova in cucina...
      if ((TimerDaemon.tm >=12) && (Bindi.path < 2)) print "@`E tutto
           sporco di sugo di pomodoro.^";
       ],
   has static container ~open openable;
```

quando questo oggetto viene esaminato, Inform stampa a video messaggi diversi a seconda che la signora Bindi si trovi o meno in cucina.

§4.9 Le estensioni (o librerie aggiuntive)

Nonostante Inform sia un linguaggio di programmazione orientato alle avventure testuali, a mano mano che si diventa più bravi nella programmazione (unita ovviamente alla stesura delle storie), si arriva a un certo punto che Inform, di default, non basta più. Si vuole ottenere sempre di più, arrivando a voler gestire eventi sempre più complessi e a questo punto si è obbligati a dover scegliere fra due strade diverse: la prima, quella forse più facile e intuitiva, è quella di scrivere del codice "ad hoc"; la seconda, invece, è quella di utilizzare le estensioni (chiamate anche librerie aggiuntive) [9]. Fra queste ultime, mi preme ricordare:

- **la scenic.h** (scritta da Richard Barnett, Joe Mason, Roger Firth e Stefano Gaburri), che ci dà la possibilità di attribuire a un oggetto dei nomi "scenici" che estendono la descrizione di default (evitando così di dover creare un nuovo oggetto per ogni nuova descrizione, come abbiamo già visto nel paragrafo 3.5). Per usarla, occorre scaricare il file estensioni_inform.zip [10], decomprimere quest'ultimo in una directory qualsiasi, andare nella directory scenic, copiare il file Italian.h[11] in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Base" (se state usando Jif), copiare il file scenic_it.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare il file scenic_it_test.inf in "C:\Inform" (se state usando WIDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Vediamo adesso come funziona:

  SCENIC_IT - TEST
  Un esempio di utilizzo della scenic.h
  Adattato alla lingua italiana da Paolo Maroncelli 030104
  Traduzione commenti e istruzioni di Paolo Lucchesi 021010

  Versione 1 -- Numero di serie 050813
  Inform v6.30 -- Libreria 6/11 -- Infit v2.5 S

  Foresta
  La foresta sembra stendersi in ogni direzione. Attorno a te i cespugli, intricati in modo quasi impenetrabile, combattono silenziosamente per il possesso del sottobosco, mentre sopra di te gli alberi si stagliano alti e spettrali attraverso l'aria umida e soffocante.

  >esamina gli alberi
  Ricoperti di licheni, gli alberi non offrono nessun appiglio oltre ai rami che si trovano molti metri sopra di te.

  >esamina i cespugli
  Rovi e spine spuntano dai cespugli avviluppandosi in pericolose spire.

  >esamina il sottobosco
  L'unico colore vivo in questo scenario desolato. Il sottobosco verde smeraldo è soffice sotto i tuoi piedi.

  >

  Qualcuno di voi si starà di sicuro chiedendo che cosa ci sia di nuovo in tutto questo. Beh, andiamo a vedere la descrizione dell'oggetto Foresta:

```
Object  foresta "Foresta"
  with  description
           "La foresta sembra stendersi in ogni direzione. Attorno a
```

[9] Per alcune di queste, il compilatore produce una serie di warning durante la fase di compilazione; non dovete tuttavia preoccuparvi perché l'eseguibile prodotto è comunque perfettamente funzionante.
[10] Questo file (così come listati_inform.zip) è soggetto a continui aggiornamenti da parte del sottoscritto. Quindi, di tanto in tanto, scaricatelo per verificare al suo interno i nuovi contenuti (se presenti).
[11] Per usare correttamente la scenic_it.h è necessario sostituire il file Italian.h originale di Infit 2.5 con questo.

```
            te i cespugli, intricati in modo quasi impenetrabile,
            combattono silenziosamente per il possesso del sottobosco,
            mentre sopra di te gli alberi si stagliano alti e
            spettrali attraverso l'aria umida e soffocante.",
        name 'foresta' 'bosco',
        scenic
          'albero' 'alberi' 'alti' 'spettrali' 0 "Ricoperti di
            licheni, gli alberi non offrono nessun appiglio oltre
            ai rami che si trovano molti metri sopra di te."
          'cespugli' 'cespuglio' 'intricati' 0 "Rovi e spine spuntano
            dai cespugli avviluppandosi in pericolose spire."
          'sottobosco' 'terreno' 0 "L'unico colore vivo in questo
            scenario desolato. Il sottobosco verde smeraldo @`e
            soffice sotto i tuoi piedi."
          'aria' 'umida' 'soffocante' 0 "Densa, quasi soffocante,
            l'atmosfera pesante ti opprime."
          'cielo' 0 "Il cielo non @`e visibile attraverso le foglie."
          'ramo' 'rami' 0 NULL,
         scenicnull
           'lichene' 'licheni' 'spine' 'spina' 'rovo'
           'rovi' 'spire' 'spira' 'colore' 'verde' 'smeraldo' 'foglia'
           'foglie' 'atmosfera',
        before [;
            Listen: "Urla di scimmie, pipistrelli, pappagalli,
                     macao.";
            ],
  has   light female;
```

State dunque iniziando a capire i grandi vantaggi che porta questa libreria?
Di default, per ottenere tutte queste descrizioni dovremmo creare un oggetto per ognuna di esse (dovremmo cioè creare un oggetto cielo, albero, cespuglio e così via) mentre qui basta associare dei nomi “scenici” fatti poi seguire da uno zero. Osserviamo anche un’altra cosa:

>esamina le foglie
Non è importante ai fini del gioco.

Normalmente, la risposta di sistema sarebbe "Non vedi nulla del genere." (cosa non vera, poiché è alquanto improbabile che un albero non abbia almeno qualche foglia) ma anche qui la libreria ci viene in aiuto; basta far seguire lo zero da NULL o, eventualmente, usare la proprietà scenicnull (che non è di fatto obbligatoria ma, come potete ben vedere, estremamente utile).
Ci sono ancora due ultime cose da notare:

```
.
.
.
[ Initialise;
        location = foresta;
];

Global ScenicFlag=4;
Constant ScenicError NULL;
Include "scenic_it";
Include "ItalianG";
```

Impostando la varibile globale ScenicFlag a 4, facciamo sì che quando il giocatore esamina il titolo della stanza:

```
>esamina la foresta
La foresta sembra stendersi in ogni direzione. Attorno a te i cespugli, intricati in modo quasi impenetrabile, combattono silenziosamente per il possesso del sottobosco, mentre sopra di te gli alberi si stagliano alti e spettrali attraverso l'aria umida e soffocante.
```

non appare il classico messaggio di default ("Non è importante ai fini del gioco.") ma la descrizione della stanza stessa. Definendo, infine, la costante ScenicError e impostandola a NULL facciamo in modo che, al verificarsi di un "errore scenico", segua il messaggio di default ("Non è importante ai fini del gioco."). Ulteriori informazioni potete comunque trovarle all'interno della libreria stessa (che potete aprire con un normalissimo text-editor);

- **la fnote.h** (scritta da L. Ross Raszewski), che ci permette di integrare delle note a pie' di pagina nel gioco. Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory qualsiasi, andare nella directory fnote, copiare il file fnote_it.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare il file fnote_it_test.inf in "C:\Inform" (se state usando WIDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Vediamo adesso come funziona:

```
FNOTE_IT - TEST
Un esempio di utilizzo dell'fnote_it.h
Traduzione commenti e istruzioni di Paolo Lucchesi

Versione 1 -- Numero di serie 050813
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

Foresta
La foresta [Nota 1] sembra stendersi in ogni direzione. Attorno a te i cespugli, intricati in modo quasi impenetrabile, combattono silenziosamente per il possesso del sottobosco, mentre sopra di te gli alberi [Nota 2] si stagliano alti e spettrali attraverso l'aria umida e soffocante.

>
```

Come potete facilmente vedere, nella descrizione della foresta appaiono due note racchiude tra le parentesi quadre. Ecco allora come leggerle:

```
>nota 1
[Nota 1]
Accidenti, che caldo. Nessuno di voi ha un ventilatore da prestarmi?

>nota 2
[Nota 2]
Altissimi e con i tronchi nodosi! Ma dove diavolo sono finito?
```

Bello, vero? Proviamo allora ad andare verso est:

```
>e

Piramide sepolcrale
La piramide sepolcrale [Nota 3] è ormai ridotta a un cumulo di materiali da costruzione. Alcuni gradini scolpiti nella nuda roccia portano giù, nell'oscurità.

>nota 3
[Nota 3]
Qualcuno di voi è superstizioso? Io no...

>nota 4
La nota [4] non è stata menzionata.
```

```
>nota 2
[Nota 2]
Altissimi e con i tronchi nodosi! Ma dove diavolo sono finito?

>
```

Se vogliamo leggere delle note non ancora menzionate (non ancora, cioè, apparse nel gioco), la libreria ce lo impedisce. In compenso, però, è possibile (giustamente) leggere le note già citate da un punto qualsiasi dell'avventura. Per quanto riguarda invece il suo utilizzo all'interno di un listato ecco quello che si deve fare:

```
[ Initialise...
.
.
.

[ PrintNote n;
     switch(n) {
         1: "Accidenti, che caldo. Nessuno di voi ha un ventilatore da
             prestarmi?";
         2: "Altissimi e con i tronchi nodosi! Ma dove diavolo sono
             finito?";
         3: "Qualcuno di voi @`e superstizioso? Io no...";
     };
];

Include "ItalianG";
Constant MAX_FOOTNOTES 5;
Include "fnote_it";
```

La libreria deve essere inclusa DOPO l'ItalianG.h, ma PRIMA devono essere definiti il numero massimo di note da visualizzare (dichiarando la costante MAX_FOOTNOTES con un valore che può andare da un minimo di 1 a un massimo 256) e i testi delle note stesse (con la funzione PrintNote che, come credo abbiate già capito, deve essere sempre presente).
Le note devono infine essere richiamate con l'istruzione (note) n dalla descrizione di un qualsiasi oggetto:

```
Object  foresta "Foresta"
  with  name 'foresta' 'bosco',
        description [;
            "La foresta", (note) 1, " sembra stendersi in ogni
             direzione. Attorno a te i cespugli, intricati in modo
             quasi impenetrabile, combattono silenziosamente per il
             possesso del sottobosco, mentre sopra di te gli alberi",
             (note) 2, " si stagliano alti e spettrali attraverso
             l'aria umida e soffocante.";
        ],
        e_to piramide,
        before [;
            Listen: "Urla di scimmie, pipistrelli, pappagalli,
                     macao.";
            ],
  has   light female;
```

Le note, inoltre, sono sempre numerate in sequenza ascendente, indipendentemente dall'ordine in cui sono chiamate;

- **la dmenus.h** (scritta da Dave Robinson), che ci viene incontro nella gestione semplificata dei menu. Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory qualsiasi, andare nella directory dmenus, copiare il file dmenus_it.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare i file dmenus_it_test.inf e Ralph.inf in "C:\Inform" (se state usando WIDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Vediamo adesso come funziona:

```
Constant Story "DMENUS_IT - TEST";
Constant Headline
        "^Un esempio di utilizzo della DMenus_it.h^
         Traduzione di Alessandro Schillaci^
         Testo dei menu di Paolo Lucchesi^^
         (digitare help per aiuto, info per informazioni sul
          gioco)^^";

Include "Parser";
Include "VerbLib";
Include "Dmenus_it";
Include "replace";
.
.
.

[ Initialise;
    location = foresta;
];

[ HelpSub;  ShowMenu(helpmenu); ];

Menu helpmenu "Aiuto e Informazioni:";

Object -> menuat "Introduzione alle Avventure Testuali"
  with description [; ... ];

Object -> menuhow "Come si giocano le Avventure Testuali"
  with description [; ... ];

Object -> menucom "Comandi particolari"
  with description [; ... ];

Object -> menualtre "Dove trovare altre Avventure Testuali"
  with description [; ... ];

Object -> menucred "Crediti, Ringraziamenti e Licenza"
  with description [; ... ];

! ----------------------------------------------------------------
Include "ItalianG";

Verb meta 'help' 'info' 'informazioni' 'istruzioni' 'crediti' 'aiuto'
          'hints' 'about' 'credits' *                        -> Help;
```

Come potete ben vedere, ogni sezione del nostro menu viene trattata come un vero e proprio oggetto; questo rende meno macchinosa la sua creazione, ottenendo tra l'altro lo stesso e identico risultato visto per il paragrafo 4.2. Questa estensione, inoltre, ci dà la possibilità di creare voci nascoste o "bloccate" (come spiegato all'interno della libreria stessa) e dei menu annidati (come mostrato nella libreria command_it.h);

- **la command_it.h**, da me scritta, è un menu (annidato) di aiuto per i comandi. Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory qualsiasi, andare nella directory command, copiare i file command_it.h e dmenus_it.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare il file command_it_test.inf in "C:\Inform" (se state usando WIDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Il suo funzionamento è molto semplice:

  COMMAND_IT - TEST
  Un esempio di utilizzo della command_it.h

  (digitare comandi per l'aiuto sui comandi)

  Versione 1 -- Numero di serie 050921
  Inform v6.30 -- Libreria 6/11 -- Infit v2.5 S

  Foresta
  La foresta sembra stendersi in ogni direzione. Attorno a te i cespugli, intricati in modo quasi impenetrabile, combattono silenziosamente per il possesso del sottobosco, mentre sopra di te gli alberi si stagliano alti e spettrali attraverso l'aria umida e soffocante.

  > comandi

  ATTENZIONE: i comandi elencati sono quelli previsti di default dalla libreria italiana (Infit 2.5 (c) Giovanni Riccardi). Ulteriori comandi del gioco (come ad esempio 'FOTOGRAFA' in Ruins) non sono qui compresi.

  [Premi un tasto per continuare]

  Come potete facilmente notare, questa estensione mostra "solo" (si fa per dire) i comandi di default. A ogni modo, una volta premuto un tasto qualsiasi, appare un menu contenente tutte (o quasi) le lettere dell'alfabeto, all'interno delle quali trovate l'elenco dei relativi comandi e, per ognuno di essi, il relativo esempio d'utilizzo;

- **la wtalk.h** (scritta da Paolo Lucchesi), ci viene in aiuto nei dialoghi con i PNG (i Personaggi Non Giocatori, ovvero tutti i personaggi che appaiono nel gioco all'infuori del giocatore stesso). Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory qualsiasi, andare nella directory wtalk, copiare il file wtalk_it.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare il file wtalk_it_test.inf in "C:\Inform" (se state usando WIDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Vediamo adesso come funziona:

  La libreria wtalk.h (e wtalk_it.h)
  Una dimostrazione interattiva
  di Paolo Lucchesi

  Versione 1 -- Numero di serie 050822
  Inform v6.30 -- Libreria 6/11 -- Infit v2.5 S

  Sala da pranzo
  La sala da pranzo, larga e spaziosa, è ben illuminata. L'unica porta conduce a nord.

  Puoi vedere Maria qui.

  > parla a Maria

  se digitiamo il comando PARLA A MARIA, appare nella metà superiore dello schermo di Windows Frotz 2002 la seguente finestra di dialogo:

```
Sala da pranzo                                        (Parlando con Maria)
 1) Buongiorno, Maria. Come sta?
 2) Sa per caso dirmi dove posso trovare il sig. Toleni?
 3) Quando verrà servito il pranzo?
```

digitando ad esempio il numero 1, Inform chiede a Maria come sta:

```
>1
"Buongiorno, Maria. Come sta?"
"Abbastanza bene, grazie."

>
```

e così via. Per terminare il dialogo potete dirigervi a nord o, eventualmente, fare tutte le domande mostrate. Inutile dire i grandi vantaggi che offre questa libreria, che prevede tra l'altro anche la possibilità di "nascondere" certe domande fino a quando non capita un certo evento. Ulteriori informazioni sul suo utilizzo potete trovarle nei commenti dell'esempio stesso;

- **la flags.h**, scritta da Alessandro Schillaci, ci dà la possibilità di stampare a video dei messaggi di aiuto sulla base del numero dei capitoli[12] e delle mosse effettuate. Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory qualsiasi, andare nella directory flags, copiare il file flags_it.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare il file flags_it_test.inf in "C:\Inform" (se state usando WIDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Vediamo adesso come funziona:

```
Esempio di utilizzo della libreria flagsit.h
Esercizio 1
Alessandro Schillaci

Versione 1 -- Numero di serie 050823
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

Oscurità
È completamente buio, e non riesci a vedere niente.

>i
Stai portando:
  un accendino

>z
Il tempo passa.

>z
Il tempo passa.

Messaggio di aiuto per il 3 turno del capitolo 1

>
```

Come potete facilmente notare, alla terza mossa qualsiasi effettuata dal giocatore, appare il primo messaggio di aiuto. Alla decima mossa, Inform stampa a video un secondo messaggio di aiuto:

[12] Un'avventura testuale può essere intesa come costituita da un'insieme di locazioni (se consideriamo la sola struttura del programma) o anche da capitoli (se consideriamo invece la storia di per sé). In Ruins, per esempio, il primo capitolo potrebbe essere costituito dall'inizio del gioco fino all'apertura dell'entrata della piramide, il secondo dalla discesa dei gradini fino al ritrovamento della chiave e così via…

```
>z
Il tempo passa.

Forse dovrei accendere l'accendino

>
```

che ci suggerisce di accendere l'accendino. Proviamo allora a farlo:

```
>accendi l'accendino
Con uno scatto del pollice, accendi l'accendino.
Ora riesci a vedere la stanza intorno a te...

Una stanza vuota
Questa è la stanza dell'esercitazione.
La stanza non ha porte, se non quelle della tua immaginazione.

Puoi vedere un foglio di carta qui.

>
```

Questo sistema di helping è molto utile nel caso in cui il giocatore sia bloccato in un punto qualsiasi del gioco e non sappia più come andare avanti. Occorre tuttavia notare che i due messaggi di aiuto sono di due tipi diversi: il primo, infatti, appare indipendentemente da quello che il giocatore sta facendo, mentre il secondo appare solo se dopo 10 mosse il giocatore non ha ancora acceso l'accendino. D'altra parte, i due messaggi vengono "attivati" da due diverse istruzioni:

```
setMessage(1,3,"Messaggio di aiuto per il 3 turno del capitolo 1^");
setMessageFlag(1,10,"Forse dovrei accendere l'accendino^",2);
```

Nel primo caso, il parametro 1 è il numero del capitolo, mentre il parametro 3 è il numero dei turni; nel secondo, come prima, il valore 1 è il numero del capitolo, il valore 10 è il numero dei turni mentre l'ultimo valore (il 2) è il numero del flag successivo. Volendo, potete scegliere se usare uno solo o entrambi i tipi di messaggi; a ogni modo, ulteriori informazioni sul suo utilizzo potete trovarle nei commenti dell'esempio stesso;

- **la pname.h** (scritta da Neil Cerutti), che ci permette di risolvere le ambiguità dell'input del giocatore quando la stessa parola di dizionario si riferisce a più di un oggetto. Viene cioè definita una nuova proprietà pname per gli oggetti, con un aspetto simile a quello della normale proprietà name ma al cui interno l'ordine delle parole è importante. Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory qualsiasi, andare nella directory pname e copiare il file pname.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif). Ulteriori informazioni sul suo utilizzo potete trovarle al capitolo 12 della Guida A Inform Per Principianti (consultabile anche dall'indirizzo http://www.inform-italia.org/docs/gip) o, per chi conosce l'inglese, il file pname.txt contenuto della directory pname;
- **la smartcantgo.h** (scritta da David Wagner e Roger Firth), che può essere usata con la proprietà cant_go di una locazione per stampare a video le uscite possibili anziché il messaggio standard "Non puoi andare in quella direzione.". Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory qualsiasi, andare nella directory smartcantgo, copiare il file smartcantgo_it.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare il file

smartcantgo_it_test.inf in "C:\Inform" (se state usando WIDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Vediamo adesso come funziona:

```
SMARTCANTGO_IT - TEST
Un esempio di utilizzo della smartcantgo_it.h

Versione 1 -- Numero di serie 050823
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

Foresta
La foresta sembra stendersi in ogni direzione. Attorno a te i cespugli, intricati in modo quasi impenetrabile, combattono silenziosamente per il possesso del sottobosco, mentre sopra di te gli alberi si stagliano alti e spettrali attraverso l'aria umida e soffocante.

>o
Puoi andare solo a est.

>e

Piramide sepolcrale
La piramide sepolcrale è ormai ridotta a un cumulo di materiali da costruzione. Alcuni gradini scolpiti nella nuda roccia portano giù, nell'oscurità.

>n
Puoi andare solo a ovest o giù.

>giù
Le macerie ostruiscono il passaggio dopo appena pochi passi.

>o

Foresta

>
```

Molto utile, non trovate? Ecco ora il listato d'esempio:

```
Constant Story "SMARTCANTGO_IT - TEST";
Constant Headline
        "^Un esempio di utilizzo della smartcantgo_it.h^^";

Include "Parser";
Include "smartcantgo_it";
Include "VerbLib";
Include "replace";

! ---------------------------------------------------------------------
Class   Room
 with   cant_go [; SmartCantGo(); ],
  has   light;

Room    foresta "Foresta"
  with  name 'foresta' 'bosco',
        description [;
            "La foresta sembra stendersi in ogni direzione. Attorno a
             te i cespugli, intricati in modo quasi impenetrabile,
             combattono silenziosamente per il possesso del
             sottobosco, mentre sopra di te gli alberi si
             stagliano alti e spettrali attraverso l'aria umida e
```

```
             soffocante.";
        ],
        e_to piramide
        before [;
            Listen: "Urla di scimmie, pipistrelli, pappagalli,
                     macao.";
            ],
  has   female;

Room    piramide "Piramide sepolcrale",
  with  name 'piramide' 'sepolcrale',
        description [;
           "La piramide sepolcrale @`e ormai ridotta a un cumulo di
            materiali da costruzione. Alcuni gradini scolpiti nella
            nuda roccia portano gi@`u, nell'oscurit@`a.";
        ],
        w_to foresta,
        d_to [; "Le macerie ostruiscono il passaggio dopo appena pochi
                 passi."; ],
   has  female;

! ---------------------------------------------------------------------
[ Initialise;
        location = foresta;
];

! ---------------------------------------------------------------------
Include "ItalianG";
```

Si deve definire una classe che contenga l'istruzione cant_go [; SmartCantGo();] da associare poi a tutte quelle locazioni in cui si voglia che la libreria "operi"[13]. Tutto qui. Se siamo al buio, viene ripristinato il messaggio di sistema e se non ci sono uscite viene stampato a video il messaggio "Non ci sono uscite.". Se abbiamo invece bisogno di associare un nostro messaggio alla proprietà cant_go di una certa locazione, basta semplicemente definirlo all'interno di quest'ultima, poiché la libreria se ne "accorge" e gli dà la priorità massima;

- **la doors.h**, scritta da L. Ross Raszewski, che ci permette di semplificare la gestione di una door. Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory qualsiasi, andare nella directory doors, copiare il file doors.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare il file doors_test.inf in "C:\Inform" (se state usando WIDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Vediamo adesso come funziona:

```
Corridoio in pendenza
Un corridoio basso e squadrato va da nord verso sud, inclinandosi verso la fine.

Il passaggio è bloccato da una massiccia porta di pietra gialla.

C'è una preziosa statuetta maya qui!

>apri la porta
Sembra essere chiusa a chiave.

>apri la porta con la chiave
Ora la porta di pietra non è più chiusa a chiave.
```

[13] Tutte quelle locazioni, cioè, in cui siano previste delle direzioni; le altre, invece, si definiscono normalmente.

>apri la porta
Ora hai aperto la porta di pietra.

>spingi la lampada a sud

Il Tempio
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

Sopra il ripiano dell'altare vedi una maschera facciale in mosaico di giada.

>spingi la lampada a nord

Corridoio in pendenza

La grande porta di pietra gialla è aperta.

C'è una preziosa statuetta maya qui!

>

Anche qui, come per la scenic.h, qualcuno potrebbe giustamente chiedersi qual è la differenza. A livello di gioco, sicuramente nessuna, ma osserviamo per un attimo il codice della porta di pietra gialla:

```
Connector  StoneDoor "porta di pietra"
  with  name 'porta' 'massiccia' 'grande' 'pietra' 'gialla',
        description
            "@`E solo una grossa porta di pietra.",
        when_closed
            "Il passaggio @`e bloccato da una massiccia porta di
             pietra gialla.",
        when_open
            "La grande porta di pietra gialla @`e aperta.",
        s_to Shrine,
        n_to Corridor,
        with_key stone_key,
        found_in Corridor Shrine,
  has   female static openable lockable locked;
```

non vi sembra decisamente più semplice e intuitivo rispetto all'originale (che abbiamo visto nel paragrafo 3.5)? Ulteriori informazioni sul suo utilizzo potete comunque trovarle studiando il listato d'esempio ad essa allegato;

- **la easydoors.h**, scritta da Andrew MacKinnon, ci permette anch'essa di semplificare la gestione di una door, ma con delle caratteristiche in più. Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory qualsiasi, andare nella directory easydoors, copiare il file easydoors_it.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare il file

easydoors_it_test.inf in "C:\Inform" (se state usando IF-IDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Vediamo adesso come funziona:

```
EASYDOORS_IT - TEST
Un esempio di utilizzo della easydoors_it.h

Versione 1 -- Numero di serie 050825
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

La Stanza A
Sei nella stanza A.

La porta di legno è chiusa.

>esamina la porta
È solo una vecchia porta di legno.

>apri la porta
Sembra essere chiusa a chiave.

>i
Stai portando:
  una grossa chiave

>apri la porta con la chiave
Ora la porta di legno non è più chiusa a chiave.

>o
(aprendo la porta di legno)

La Stanza B
Sei nella stanza B.

La porta di legno è aperta.

>
```

Avete notato cosa accade quando il giocatore si dirige a ovest? La porta, pur essendo ancora chiusa, viene aperta da Inform in automatico. Vediamo allora il codice relativo a quest'oggetto:

```
Doorway Porta "porta di legno"
   with name 'porta' 'vecchia' 'legno',
        description
            "@`E solo una vecchia porta di legno.",
        side1_to StanzaB,
        side1_dir w_to,
        side2_to StanzaA,
        side2_dir e_to,
        opendesc "aperta",
        closeddesc "chiusa",
        isconcealed 0,
        autoopen 1,
        with_key chiave,
        found_in StanzaA StanzaB,
    has female static openable lockable locked;
```

per far sì che l'apertura della porta avvenga in maniera automatica, la proprietà autoopen deve necessariamente valere 1; se vale 0, si deve aprire "manualmente" (inserendo cioè, come già sappiamo, il comando APRI LA PORTA dopo averla sbloccata con la chiave). La proprietà

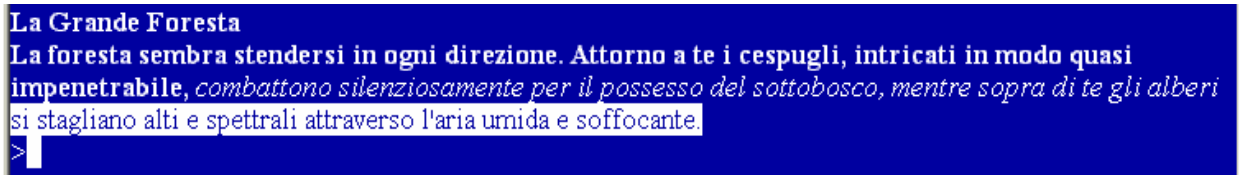
isconcealed, invece, stabilisce se la descrizione della porta debba apparire nella stanza (valore 0) oppure no (valore 1). Le proprietà opendesc e closeddesc rappresentano infine i messaggi di default relativi allo stato della porta (se è cioè aperta o chiusa) e non devono mai essere modificati tranne quando l'oggetto di tipo door a cui fanno riferimento è maschile (come ad esempio un cancello); in questo caso, occorre assegnare la stringa "aperto" alla proprietà opendesc e "chiuso" alla proprietà closeddesc.
E se volessimo ampliare la descrizione dello stato della nostra porta? Basta ricorrere alla proprietà describe:

```
Doorway Porta "porta di legno"
   with name 'porta' 'vecchia' 'legno',
        description
            "@`E solo una vecchia porta di legno.",
        describe [;
            print "^Puoi vedere una porta di legno ";
            if (self has open) print "aperta ";
            if (self hasnt open) print "chiusa ";
            print "qui.^";
        ],
        side1_to StanzaB,
        side1_dir w_to,
        side2_to StanzaA,
        side2_dir e_to,
        autoopen 1,
        with_key chiave,
        found_in StanzaA StanzaB,
    has female static openable lockable locked;
```

che porta al seguente risultato:

```
EASYDOORS_IT - TEST
Un esempio di utilizzo della easydoors_it.h

Versione 1 -- Numero di serie 050825
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

La Stanza A
Sei nella stanza A.

Puoi vedere una porta di legno chiusa qui.

>esamina la porta
È solo una vecchia porta di legno.

>o
Non puoi, la porta di legno è in quella direzione.

>apri la porta con la chiave
Ora la porta di legno non è più chiusa a chiave.

>o
(aprendo la porta di legno)

La Stanza B
Sei nella stanza B.

Puoi vedere una porta di legno aperta qui.

>
```

- **la style.h**, scritta da Chris Klimas, che ci aiuta nella gestione degli stili del testo. Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory



qualsiasi, andare nella directory style, copiare il file style.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare il file style_test.inf in "C:\Inform" (se state usando WIDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Vediamo adesso come funziona:

Per ottenere in Inform standard un effetto di questo tipo dobbiamo ricorrere, come ormai sapete, all'istruzione style:

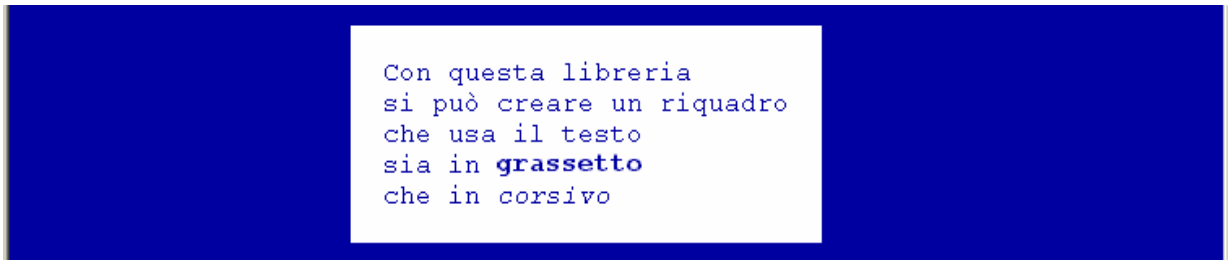
```
Object  Forest "La Grande Foresta"
  with   description [;
            style bold;
            print "La foresta sembra stendersi in ogni direzione.
                   Attorno a te i cespugli, intricati in modo quasi
                   impenetrabile,";
            style roman; style underline;
            print " combattono silenziosamente per il possesso del
                   sottobosco, mentre sopra di te gli alberi ";
            style roman; style reverse;
            print "si stagliano alti e spettrali attraverso l'aria
                   umida e soffocante.";
            style roman;
         ],
  has    light;
```

che è sì efficace ma molto poco funzionale. Passiamo invece al codice che fa riferimento all'estensione:

```
Object  Forest "La Grande Foresta"
  with  description [;
           print (b) "La foresta sembra stendersi in ogni direzione.
                      Attorno a te i cespugli, intricati in modo
                      quasi impenetrabile, ";
           print (i) "combattono silenziosamente per il possesso del
                      sottobosco, mentre sopra di te gli alberi ";
           print (r) "si stagliano alti e spettrali attraverso l'aria
                      umida e soffocante.";
         ],
  has    light;
```

È un'altra cosa, non è vero? La lettera b sta per bold (o grassetto in italiano) la i sta per italic (o corsivo in italiano) e la r sta per reverse. L'unica nota dolente è che questa estensione, a differenza di tutte le altre viste finora, non funziona sotto Glulx[14];

[14] Glulx (pienamente supportato da Infit e da Jif) è un linguaggio di programmazione, scritto dal grande Andrew Plotkin, che risulta essere perfettamente compatibile con Inform (tanto da poter essere considerato a tutti gli effetti un'estensione di quest'ultimo) e offre la possibilità di creare delle avventure testuali grafiche e sonore eseguibili con il programma WinGlulxe. Ulteriori informazioni potete trovarle al capitolo 5.



- **la boxclever.h**, scritta da Roger Firth, che ci permette di creare dei riquadri in grado di contenere, oltre al testo normale, anche il grassetto e il corsivo. Per usarla, occorre scaricare il file estensioni_inform.zip, decomprimere quest'ultimo in una directory qualsiasi, andare nella directory boxclever, copiare il file boxclever.h in "C:\Inform\libraries" (se state usando WIDE) o "C:\Programmi\Jif\lib\Contrib" (se state usando Jif) e, infine, copiare il file boxclever_test.inf in "C:\Inform" (se state usando WIDE) o "C:\Programmi\Jif\Games" (se state usando Jif). Come potete ben vedere, abbiamo un riquadro di testo contenente anche del testo in grassetto e in corsivo. Per quanto riguarda il codice:

  ```
  box "Con questa libreria" "si pu@`o creare un riquadro" "che usa il
       testo" "sia in <grassetto>"  "che in [corsivo]";
  ```

  non bisogna fare altro che usare l'istruzione box (come in Inform standard) con il testo in grassetto racchiuso tra i segni minore e maggiore e il testo in corsivo racchiuso tra le parentesi quadre. Purtroppo, anche questa estensione non funziona sotto Glulx.

Bene. Le estensioni appena viste possono essere considerate le "principali". Ne esistono naturalmente molte altre[15] che, anche se non indispensabili, possono comunque essere molto utili. Fra queste, mi preme almeno ricordare:

- la betatest.h, scritta da Marnie Parker, che facilita il betatesting del gioco dando la possibilità di commentare l'avventura mentre la si gioca, registrando il tutto in un file di testo;
- la cyoa.h, scritta da Paolo Lucchesi, che permette di creare delle avventure testuali a scelte multiple (più conosciute da alcuni come libri-game);
- la daemons.h, scritta da Andrew Plotkin e Roger Firth, che permette di creare dei daemon da poter eseguire in ordine di priorità;
- l'easyout.h, scritta da Gabriele Brami, che verifica la presenza di proprietà direzionali in una stanza stampando poi la direzione e la destinazione;
- la Io.h, scritta dal sottoscritto che, come avete già avuto modo di vedere dal capitolo 2, ci viene incontro nella gestione dell'input-output dei dati;
- l'Italian3.h, una modifica (realizzata da Paolo Lucchesi) del file Italian.h contenuto in Infit che permette l'uso della narrativa in prima persona (oltre alla normale narrativa in seconda persona). È anche possibile passare in qualsiasi momento dalla prima alla seconda persona e viceversa;
- la math.h, scritta da Matt Albrecht, che mette a disposizione una serie di utilissime funzioni matematiche;
- le mfs library, scritte da Giancarlo Niccolai per tutti coloro che sono interessati a creare dei giochi di ruolo in Inform;
- la mininf.h, scritta da Ignazio Di Napoli e Marco Falcinelli, appositamente studiata per creare avventure giocabili facilmente su palmari, cellulari e console;
- la moveclass.h, scritta da Neil James Brown e Alan Trewartha, che permette di spostare i personaggi non giocatori lungo percorsi prestabiliti, calcolati oppure random;

[15] Quelle qui elencate sono tutte contenute all'interno del file estensioni_inform.zip. Le altre, invece, sono tutte scaricabili all'indirizzo http://www.inform-fiction.org/extensions/index.html; molte di queste però, prima di essere usate, devono essere necessariamente tradotte in italiano (su permesso, ovviamente, dei rispettivi autori).

- la printslow.h, scritta da David Cornelson, che ci permette di modificare la velocità di stampa del testo;
- la scanner.h, scritta da Alessandro Schillaci e Paolo Lucchesi, che elenca tutte le "istanze" di una classe presenti all'interno delle locazioni adiacenti alla locazione attuale senza spostarsi "fisicamente";
- la tutor.h, scritta da Paolo Lucchesi per gestire i suggerimenti nel corso di un'avventura (per capire come utilizzarla date un'occhiata al listato di Villa Morgana);
- l'untouchable.h, che permette di utilizzare un nuovo attributo denominato "untouchable" per caratterizzare quegli oggetti che pur essendo in vista non possono essere maneggiati dal giocatore perchè irraggiungibili (ad esempio un uccello in cima un albero);
- l'utility.h, scritta da L. Ross Raszewski, che contiene delle funzioni molto utili;
- la wtellask.h, scritta da Paolo Lucchesi, utilissima per definire dei Personaggi Non Giocatori che possano rispondere in maniera più efficace e funzionale a dei comandi come "chiedi al personaggio del [argomento]" e "parla al personaggio del [argomento]".

Ricordatevi però che, anche se gratuite e libere di essere utilizzate da chiunque, queste estensioni sono state scritte da persone che hanno speso tempo e fatica; di conseguenza, TUTTI COLORO CHE LE USANO HANNO L'OBBLIGO MORALE DI RINGRAZIARE I RISPETTIVI AUTORI NEI CREDITI DEI LORO GIOCHI. Inoltre, SE L'ESTENSIONE UTILIZZATA IMPLICA DI PER SÉ UN MODO D'UTILIZZO DIVERSO DA QUELLO SOLITO (COME AD ESEMPIO LA WTALK_IT.H), OCCORRE ALLORA FORNIRE NEGLI AIUTI DEL GIOCO LA RELATIVA DOCUMENTAZIONE.

§4.10 Varie, listati e decompilatore

Ci sono ancora diverse caratteristiche di questo linguaggio che meritano di essere esaminate in dettaglio, nell'ordine:

- è possibile supportare il set di caratteri in modalità estesa (come ad esempio il simbolo dell'euro "€"[16]). Vediamo allora questo semplice esempio:

```
Include "Parser";
Include "VerbLib";
Include "Replace";

Global charext_on = 0;

Zcharacter table + '@{20ac}';

[ Stampa;
  ClearScreen(); ! pulisce lo schermo
  print "Questa maglietta costa 70"; Euro(); print ".^";
  print "Queste scarpe costano 60"; Euro(); print ".^";
  print "Questa camicia costa 80"; Euro(); print ".^";
  KeyCharPrimitive(); ! legge un carattere dalla tastiera
];

! ----------------------------------------------------------------
[ Initialise;
  print "Vuoi supportare i caratteri in modalit@`a estesa (s/n)? ";
  if (yesorno()) charext_on = 1;
  Stampa();
```

[16] Per ottenere questo simbolo, premete contemporaneamente i tasti ALT (DESTRO) + E della vostra tastiera.

```
  quit; ! fine del programma
];

[ Euro;
  if (charext_on == 1) print " @{20ac}";
  else print " euro";
];

! ----------------------------------------------------------------

Include "ItalianG";
```

Come per i colori, anche qui è meglio chiedere all'inizio se si vogliono o meno supportare i caratteri in modalità estesa (il motivo è sempre lo stesso: non è detto che tutti gli interpreti Z-code supportino correttamente questa modalità). Dopodiché, vengono stampate a video tre frasi che fanno tutte riferimento alla funzione Euro, la quale stampa la parola euro o il simbolo € a seconda del valore memorizzato nella variabile globale charext_on (a seconda, cioè, che l'utente abbia accettato o meno il supporto dei caratteri in modalità estesa). Occorre tuttavia puntualizzare che, per far sì che Inform "capisca" se un dato carattere esteso debba essere supportato o meno, bisogna ricorrere all'istruzione Zcharacter table:

```
Zcharacter table + '@{20ac}';
```

dove @{20ac} è proprio il simbolo dell'euro. Ulteriori informazioni potete trovarle all'indirizzo http://www.firthworks.com/roger/informfaq/aa20.html;

- il contatore dei turni parte di default da zero. Definendo tuttavia una costante START_MOVE con un valore diverso da zero:

```
Constant Story "RUINS";
Constant Headline
        "^Un esempio di lavoro interattivo.^
         Copyright (c) 1999 di Graham Nelson.^
         Traduzione e adattamenti di Vincenzo Scarpa e Raffaello
         Valesio (c) 2002-2003 su permesso dell'autore.^^";

Constant MAX_CARRIED = 7;
Constant START_MOVE = 1;
.
.
.
```

il contatore parte da quel valore (in questo esempio 1 anziché zero);

- aggiungendo la proprietà compass_look a un oggetto:

```
Object  Square_Chamber "La Sala Quadrata"
  with  name 'sala' 'quadrata' 'architrave' 'architravi' 'entrata',
        description
            "Sei in una sala di pietra oscura e profonda, larga circa
             dieci metri. Un raggio di sole, proveniente dalla cima
             della scalinata, la illumina diffusamente, ma le ombre
             del livello pi@`u basso rivelano dei passaggi verso est e
             sud, che conducono verso la pi@`u profonda oscurit@`a del
             Tempio.",
        compass_look [ obj;
            if (obj == u_obj) "Puoi vedere le scale che ti riportano
```

```
                            alla foresta.";
        if (obj == n_obj or w_obj) "Vedi il muro.";
    ],
    u_to Forest,
    e_to Wormcast,
    s_to Corridor,
```

è possibile usare il comando GUARDA A [DIREZIONE]:

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>guarda a nord
Vedi il muro.

>guarda a est
Non vedi niente di strano in quella direzione.

>guarda su
Puoi vedere le scale che ti riportano alla foresta.

>

- è possibile, mediante la proprietà before_implicit, mangiare un oggetto prima ancora di possederlo:

```
Object  -> mushroom "fungo macchiato"
.
.
.
after...

before_implicit [;
     Take: if (action_to_be == ##Eat) return 2;
],
.
.
.
```

Per capire meglio quanto è stato detto proviamo a vedere quello che accade normalmente (senza, cioè, la proprietà before_implicit):

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi Crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.
Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>mangia il fungo

(prima prendi il fungo macchiato)
Hai raccolto abilmente il fungo, senza staccarlo dal suo gambo sottile.

>mangia il fungo
Lo sgranocchi ad un angolo, incapace di capire l'origine di un gusto così acre, distratto dal volo di un macao sopra la tua testa che sembra quasi un'esplosione nel sole. Il battito delle sue ali è quasi assordante, e delle pietre crollano una sull'altra.

>

Se il giocatore vuole mangiare il fungo senza ancora possederlo nell'inventario, Inform esegue automaticamente il comando PRENDI IL FUNGO in modo tale che poi il giocatore lo possa finalmente mangiare. Con la proprietà before_implicit, invece:

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi Crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.
Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>mangia il fungo
Lo sgranocchi ad un angolo, incapace di capire l'origine di un gusto così acre, distratto dal volo di un macao sopra la tua testa che sembra quasi un'esplosione nel sole. Il battito delle sue ali è quasi assordante, e delle pietre crollano una sull'altra.

>

il problema non si pone;

- l'istruzione ScreenHeight restituisce l'altezza della finestra principale (o meglio, il numero di righe che la costituiscono);
- l'istruzione ClearScreen può essere usata in diversi modi; ClearScreen(1) pulisce solo la status line, ClearScreen(2) solo la finestra principale e, infine, ClearScreen(0) - o più semplicemente ClearScreen - che le pulisce entrambe;
- se un oggetto di tipo contenitore, come ad esempio un comunissimo sacco di tela, si trova in una locazione:

```
.
.
.
Object  Forest "Il Grande Altopiano"...

Object  -> mushroom "fungo macchiato"...

Object  -> packing_case "cassa d'imballaggio"...

Object -> Sacco "sacco di tela"
  with name 'sacco' 'tela',
  has  container open;

Object  -> -> camera "macchina fotografica a lastre"...

Object  -> -> newspaper "giornale di un mese fa"...
```

```
.
.
.
```

ecco quello che accade se durante il gioco appare la descrizione della stanza:

```
Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

Puoi anche vedere un sacco di tela (nel quale ci sono una macchina fotografica a lastre e un giornale di un mese fa) qui.

>
```

Come potete vedere, il sacco di tela viene stampato a video insieme a tutto il suo contenuto. Tramite la proprietà invent, invece, è possibile “personalizzare” la sua descrizione:

```
Object -> Sacco "sacco di tela"
  with name 'sacco' 'tela',
       invent [;
           if (inventory_stage == 1) switch (children(self)) {
               0: print "un sacco vuoto";
               1: print "un sacco contenente ", (a) child(self);
               default: print "un sacco pieno di oggetti";
           }
           rtrue;
       ],
has container open;
```

Così facendo, se il sacco è vuoto viene stampato a video il messaggio Puoi anche vedere un sacco vuoto qui.; se contiene un oggetto (ad esempio il giornale), viene stampato a video il messaggio Puoi anche vedere un sacco contenente un giornale di un mese fa qui.; se contiene invece due o più oggetti viene stampato a video il messaggio Puoi anche vedere un sacco pieno di oggetti qui.:

```
Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

Puoi anche vedere un sacco pieno di oggetti qui.
```

>esamina il sacco
Dentro il sacco di tela vedi una macchina fotografica a lastre e un giornale di un mese fa.

>svuota il sacco
macchina fotografica a lastre: Posata.
giornale di un mese fa: Posato.

>g

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

Puoi anche vedere un giornale di un mese fa, una macchina fotografica a lastre e un sacco vuoto qui.

>prendi il giornale
Preso.

>metti il giornale nel sacco
Hai messo il giornale di un mese fa dentro il sacco di tela.

>g

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

Puoi anche vedere una macchina fotografica a lastre e un sacco contenente un giornale di un mese fa qui.

>prendi il sacco
Preso.

>i
Stai portando:
  un sacco contenente un giornale di un mese fa
  un dizionario maya di Waldeck
  una lampada al sodio
  la mappa di Quintana Roo

>

> Naturalmente il numero degli oggetti contenuti nel sacco è facilmente verificabile all'interno del ciclo switch-case della proprietà invent: al numero 0 corrisponde nessun oggetto, al numero 1 corrisponde un oggetto e così via…

È tutto. Ovviamente ci sarebbero molte altre cose da dire, ma dal momento che questo manuale è rivolto principalmente ai non esperti della programmazione, è meglio concludere qui il capitolo[17]. Ad ogni modo, vi consiglio vivamente di prendere la sana abitudine di andare a studiare i listati degli altri autori (se sono ovviamente disponibili) poiché solo così si impara a programmare (e a migliorare, di conseguenza, le proprie avventure testuali). Oltre a Ruins, altri due listati da studiare sono sicuramente Avventura e Negozio; altrimenti, potete sempre rifarvi anche agli altri listati reperibili sul mio sito o sul portale di If-Italia[18].
Perché però non giocarle? Le avventure testuali si "conoscono" anche imparando a risolvere i vari enigmi che ci vengono proposti durante lo svolgimento del gioco: è indubbio quindi, che un buon programmatore di avventure testuali deve anche essere a sua volta un buon giocatore. Personalmente consiglio quindi d'inizare con Flamel (di Francesco Cordella), WarMage (di Giancarlo Niccolai), la Pietra della Luna (di Paolo Lucchesi), Filaments (di JB Ferrant – tradotta in italiano da Marco Totolo) ed Enigma di Marco Vallarino. Altre avventure italiane potete trovarle sul portale di If-Italia, mentre all'indirizzo http://www.wurb.com/if/ potete trovare tutte (o quasi) quelle scritte nelle altre lingue (inglese in primis e poi, a seguire, spagnolo, francese, tedesco e - ovviamente - italiano).

[17] Per ulteriori informazioni su questo linguaggio vi consiglio di consultare, oltre all'Inform Beginner Guide e alla Guida ad Inform per Principianti - più volte citati nel manuale, la Guida a Inform-Glulx di Marco Falcinelli, la Guida a Inform-Glulx di Andrew Plotkin, le Inform Release Notes e le pratiche mini-guide InfoLib e Inform in four minutes scritte entrambe dal bravissimo Roger Firth.

[18] Esiste anche la possibilità di usare Reform, un decompilatore in grado di estrarre un listato da un file eseguibile Z-code (come spiegato nell'ottimo manuale contenuto all'interno dell'archivio stesso): il risultato ottenuto non è sempre ottimale ma d'altra parte è sempre meglio che niente.

CAPITOLO 5
INFORM E GLULX

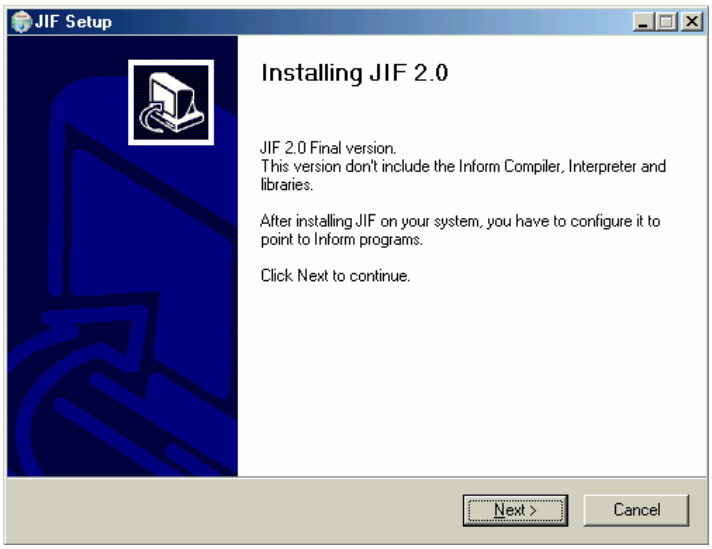
§5.1 Cos'è Glulx

È presto detto: Glulx è una virtual machine a 32 bit, scritta dal grande Andrew Plotkin, i cui compilatori sono stati disegnati per prendere il codice Z-code e convertirlo in codice Glulx (evitando così ai programmatori Inform di dover apprendere i vantaggi e le capacità extra di questo linguaggio iniziando tutto da capo). QUINDI, PER APPRENDERE GLULX, È NECESSARIO CONOSCERE (E ANCHE BENE) INFORM.
In termini più semplici, Glulx è sostanzialmente "un'estensione" di Inform che permette di ottenere delle avventure testuali grafiche e sonore. Dal momento però che, come già detto, il codice ottenuto è in formato Glulx, PER ESEGUIRE UN'AVVENTURA TESTUALE SCRITTA IN QUESTO LINGUAGGIO OCCORRE UTILIZZARE IL PROGRAMMA WINGLULXE (o - se preferite - GARGOYLE) ANZICHÉ WINDOWS FROTZ 2002 (che, come già sapete, esegue solo il codice in formato Z-code) [1].

§5.2 L'installazione di Jif

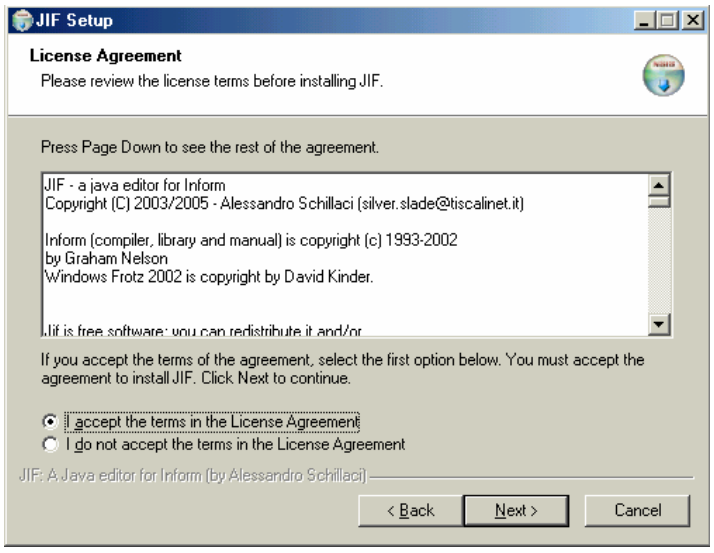
Cercare di utilizzare Glulx così com'è, non è affatto semplice come a prima vista potrebbe sembrare. JIF, DA QUESTO PUNTO DI VISTA, È UN OTTIMO EDITOR MULTIPIATTAFORMA PER PROGRAMMARE IN QUESTO LINGUAGGIO[2]. Vediamo allora come installarlo:

- la prima cosa da fare è, come già detto nel paragrafo 1.4, quella d'installare il JRE (v1.4 o successiva);
- una volta che si è installato il Java Runtime Environment (permettendo così a Windows di eseguire i programmi scritti in Java), possiamo finalmente decomprimere il file Jif20me.zip ed eseguire il file JIF2_win32.exe:

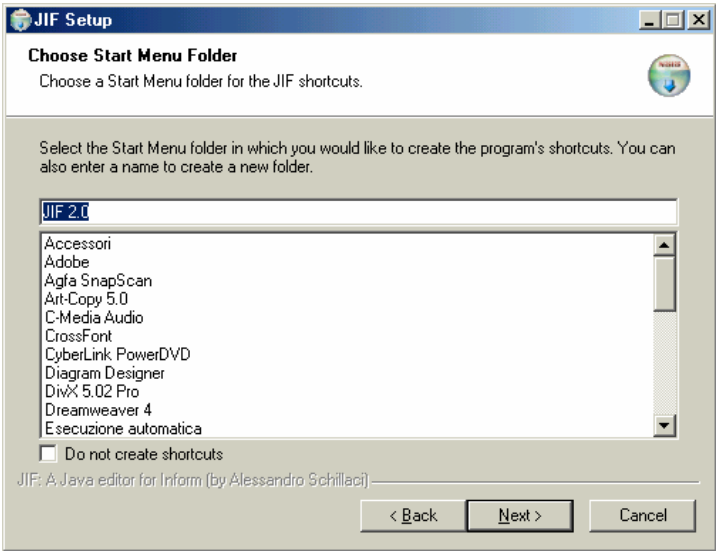


Questa è la videata iniziale di presentazione, nella quale viene espressamente specificato che in questa versione non vengono inclusi il compilatore Inform, l'interprete e le librerie e che dopo aver installato Jif sul nostro sistema occorre configurarlo per far sì che possa "elaborare" i programmi in Inform. Non preoccupatevi di tutto questo (vedremo fra poco come effettuare quest'operazione) e cliccate, con il tasto sinistro del mouse, il pulsante Next per continuare:

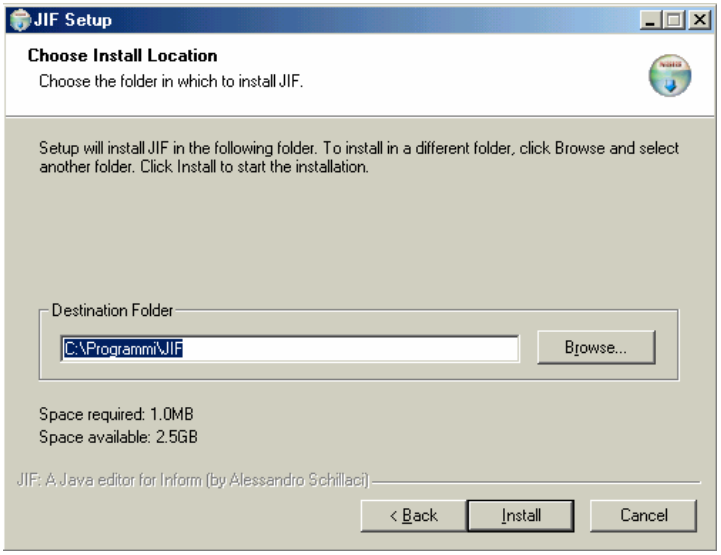
[1] In realtà, Windows Frotz 2002 è in grado di eseguire anche alcuni file nel formato .blb, ma per questi ultimi è molto meglio utilizzare allo scopo WinGlulxe. Quest'ultimo, tra l'altro, è disponibile anche nelle versioni Linux e Mac-OS.
[2] Anche WIDE può essere usato allo scopo (senza, tra l'altro, essere costretti ad installare il JRE) ma non è al momento così completo come JIF.



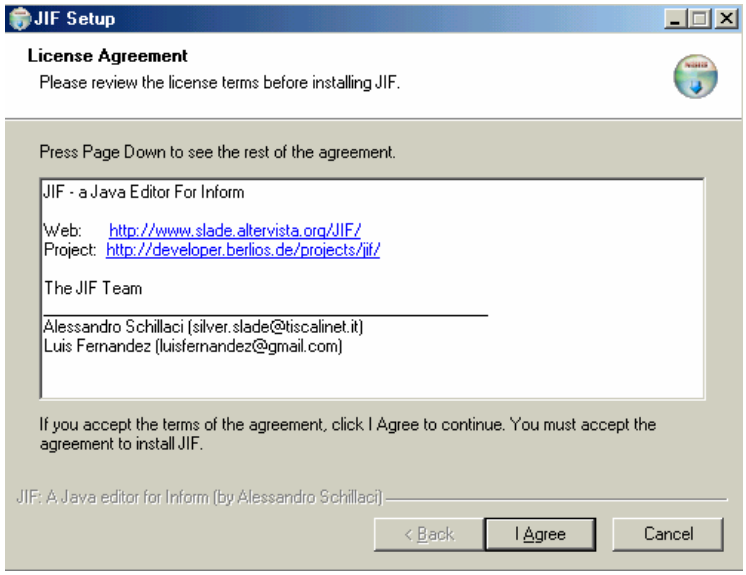
Qui dovete invece accettare gli accordi di licenza e cliccare il pulsante Next per continuare:



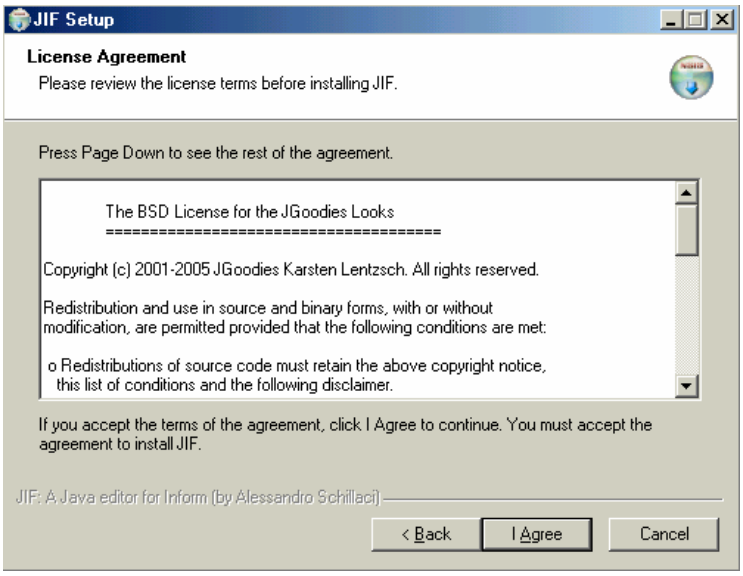
Cliccate sul pulsante Next per continuare:



In questa quarta videata, dovete specificare la directory in cui installare Jif. Lasciate quella di default e proseguite cliccando, con il tasto sinistro del mouse, il pulsante Install:



Cliccate il tasto I Agree per continuare:



Anche qui, come prima, cliccate il tasto I Agree per continuare. Nella videata che segue (qui non visualizzata), cliccate il tasto Finish per poter così terminare l'installazione;

- occorre ora copiare le directory compiler, games, interpreter, lib e tools in "C:\Programmi\JIF", le directory manuale e Jif_Glulx[3] in "C:\Programmi\JIF\Doc" e il file config.ini in "C:\Programmi\JIF\config";[4]
- ora si può, infine, avviare Jif dall'apposita icona presente sul desktop del vostro computer.

[3] La directory manuale contiene il manuale ufficiale di Jif, mentre la directory Jif_Glulx contiene delle informazioni molto utili su come compilare in Glulx con Jif.
[4] Se avete installato Jif in un'altra directory, modificate a mano i path di questo file prima di copiarlo.

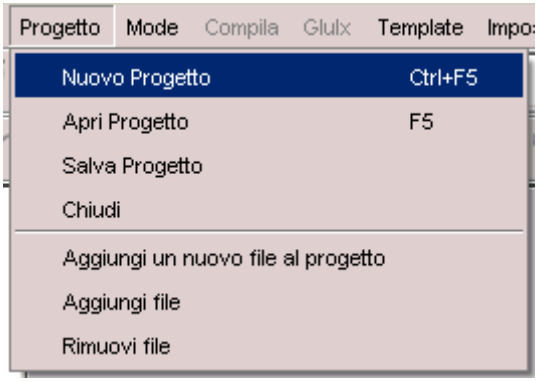
§5.3 La compilazione

Ora che abbiamo installato Jif, possiamo finalmente passare alla compilazione di un listato in Glulx. Scaricate allora il file estensioni_glulx.zip, decomprimetelo in "C:" e seguite attentamente le seguenti istruzioni:

- la prima cosa da fare è creare un file di risorse provvisorio con l'estensione .res[5]. Basta allora aprire un qualsiasi editor di testo (tra cui ovviamente Jif), creare un nuovo file e scrivere:

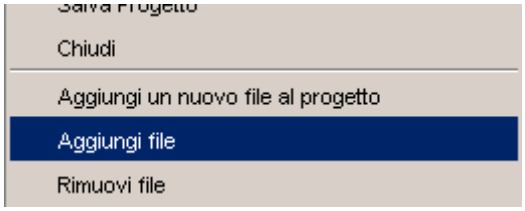
```
CODE C:\sgw_it\sgw_test_it.ulx

PICTURE room1 C:\sgw_it\IMG\room1.jpg
PICTURE room2 C:\sgw_it\IMG\room2.jpg
PICTURE room3 C:\sgw_it\IMG\room3.jpg

SOUND rumble C:\sgw_it\SND\rumble.aif
SOUND heart C:\sgw_it\SND\heart.aif
```

  salvatelo poi in "C:\sgw_it" con il nome sgw_test_it.res. Come potete facilmente notare, all'interno di questo file sono definiti i path delle immagini, dei suoni e del file con estensione .ulx (che vedremo nel prossimo paragrafo a cosa serve);
- la seconda cosa da fare è quella di creare un file progetto con l'estensione .jpf. Da Jif, create un nuovo progetto dal menu Progetto:



  e chiamatelo come sgw_test_it:

  Ora, sempre dal menu Progetto, cliccate con il tasto sinistro del mouse sulla funzione Aggiungi file:



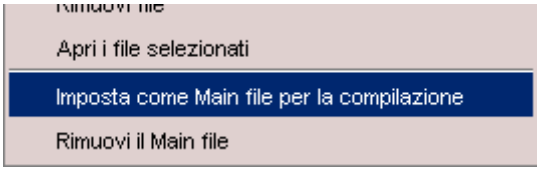
  andate in "C:\sgw_it" e selezionate, tenendo premuti contemporaneamente i tasti shift e sinistro rispettivamente della tastiera e del mouse[6], i tre file evidenziati in figura, cliccando poi con il

[5] Tutti i file necessari per la compilazione potete trovarli in "C:\sgw_it\Help". Cercate, tuttavia, di crearli da soli.
[6] Se i file interessati sono sparsi, occorre allora premere il tasto Control (o Ctrl) anziché lo Shift.

tasto sinistro del mouse sul pulsante Apri (non visibile in figura ma comunque presente all'interno della finestra):

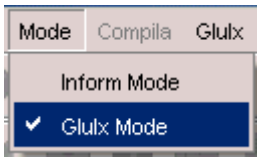
Occorre ora assegnare il Main file. Posizionatevi allora con il puntatore del mouse sulla finestra Project:

selezionate il file sgw_test_it.inf, premete il tasto destro del mouse e selezionate la funzione Imposta come Main file per la compilazione:

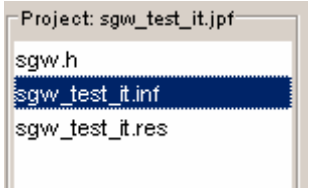


Bene. Ora salvate il progetto selezionando la funzione Salva Progetto dal menu Progetto. Il file in questione verrà automaticamente salvato - da Jif - nella directory "C:\Programmi\JIF\Projects" come sgw_test_it.jpf;[7]

- la terza cosa da fare è assicurarsi che Jif sia in Glulx Mode; per farlo, andate con il puntatore del mouse sul menu Mode e selezionate la funzione Glulx Mode:



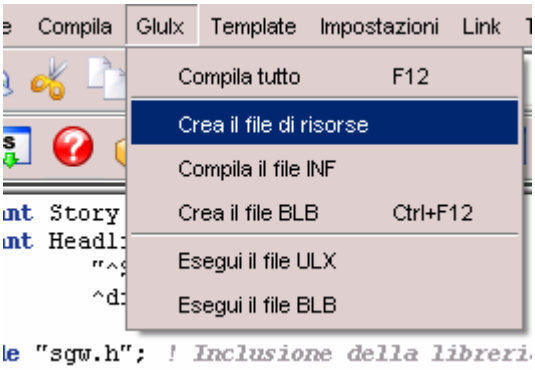
- posizionatevi ora, con il puntatore del mouse, sul nome del file sgw_test_it.inf della finestra Project e cliccate due volte consecutive il tasto sinistro del mouse:



[7] Nel caso in cui il file di progetto sia già stato creato in precedenza, occorre allora aprirlo premendo il tasto F5 oppure selezionando la funzione Apri Progetto nel menu Progetto. Assicuratevi inoltre che il Main file sia riferito a un file .inf del progetto stesso.

verrà automaticamente aperto, nella prima finestra in alto a destra, il programma in questione;

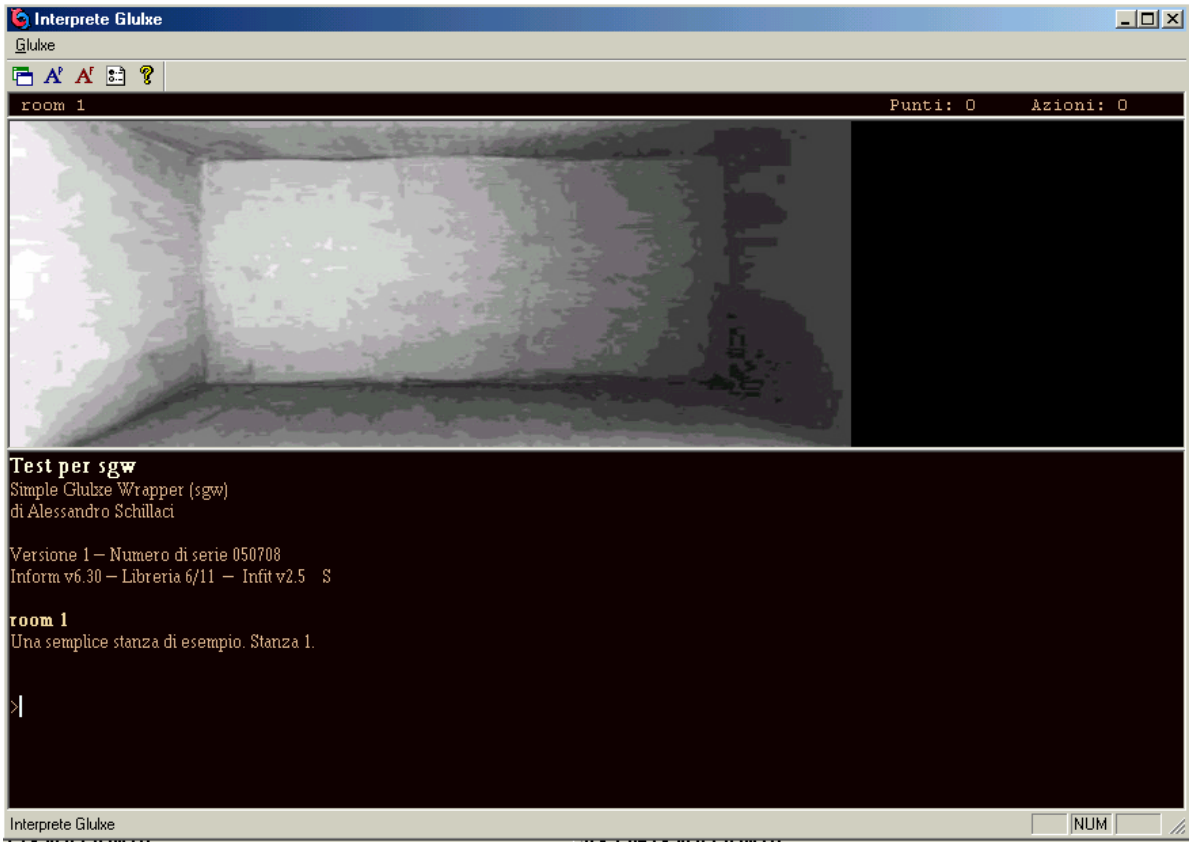
- ora siamo davvero a buon punto. Le ultime tre cose rimaste da fare sono quelle di creare il file di risorse definitivo cliccando, con il tasto sinistro del mouse, sulla funzione Crea il file di risorse del menu Glulx:



A questo punto Jif crea, partendo dal file sgw_test_it.res, i file sgw_test_it.blc e sgw_test_it.bli. Andando poi su Compila il file INF otteniamo il file sgw_test_it.ulx e infine, selezionando la funzione Crea il file BLB, il file sgw_test_it.blb. Fine della compilazione.

§5.4 L'esecuzione

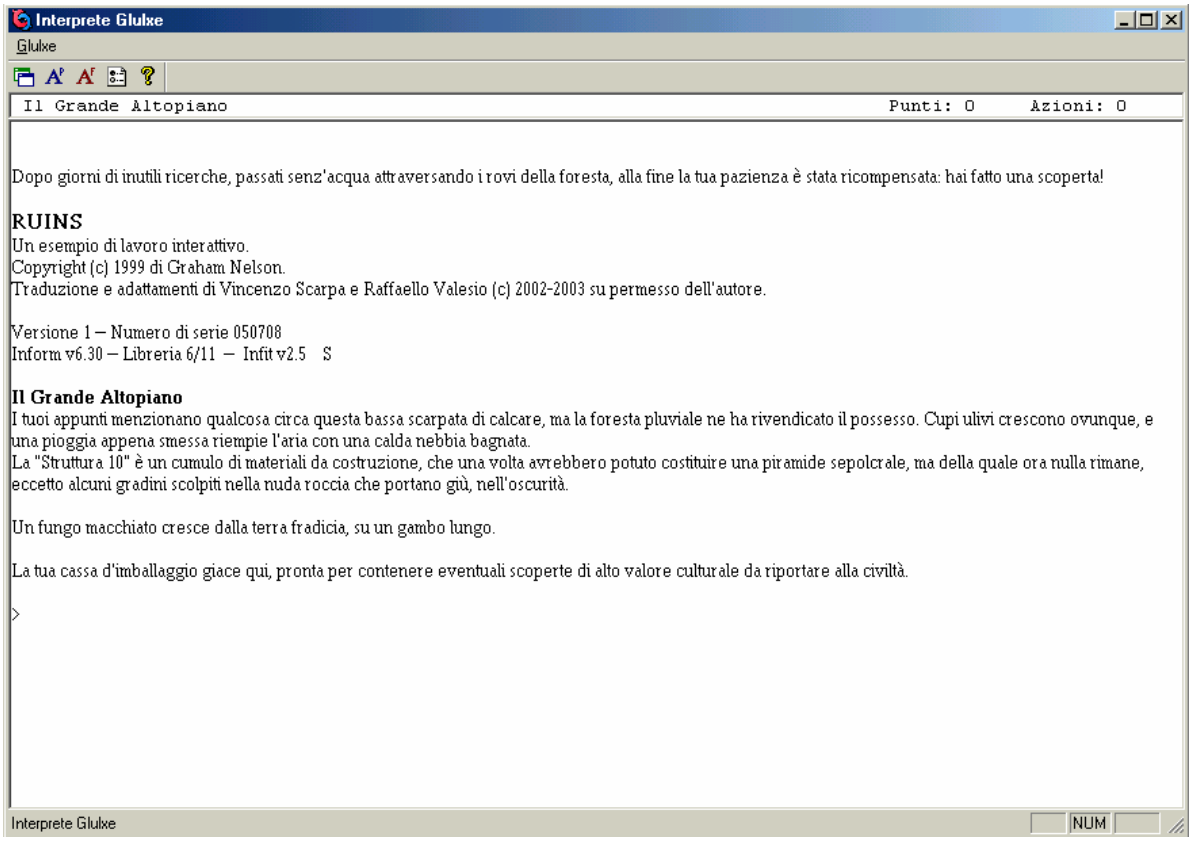
Il processo di compilazione che abbiamo visto nel paragrafo precedente si può riassumere nella creazione di due file: uno con l'estensione .ulx e l'altro con l'estensione .blb. Se proviamo ad aprirli e a eseguirli con WinGlulxe (o, eventualmente, usare le funzioni Esegui il file ULX e Esegui il file BLB del menu Glulx di Jif), abbiamo lo stesso ed identico risultato:



ovvero un esempio molto carino che ci illustra le potenzialità grafiche e sonore di questo bellissimo linguaggio[8]. In che cosa allora si differenziano questi due file?
È presto detto. UN FILE CON L'ESTENSIONE .BLB "ACCORPA" AL SUO INTERNO TUTTI GLI ELEMENTI MULTIMEDIALI UTILIZZATI (QUELLI GRAFICI E SONORI) E IL CODICE STESSO: questo fa sì che si abbia, come risultato, un unico file molto comodo da distribuire sul web.
UN FILE CON L'ESTENSIONE .ULX, INVECE, PER POTER FUNZIONARE DEVE ESSERE DISTRIBUITO INSIEME A TUTTI GLI ELEMENTI MULTIMEDIALI[9] SEPARATI DAL FILE PRINCIPALE; questo secondo tipo di file risulta essere molto utile (oltre che indispensabile per poter creare il file .blb) se vogliamo scrivere un'avventura testuale senza elementi multimediali (contenente, cioè, solo del testo) o per convertire un listato Inform in Glulx.

§5.5 Da Inform a Glulx

Se vogliamo arricchire Ruins (così come qualsiasi altra avventura testuale scritta in Inform) con un po' di grafica e di sonoro, non abbiamo alcuna speranza se la lasciamo nel formato Z-code. Compilandola invece in Glulx, otteniamo la stessa avventura di prima con in più la possibilità di arricchirla con elementi multimediali come più ci aggrada (possiamo ad esempio, nel caso specifico di Ruins, attivare un suono al crollo dell'entrata della piramide, inserire un'interfaccia grafica per la rosa dei venti, creare dei giochini interattivi che il sacerdote mummificato può proporre al giocatore come indovinelli e così via).
In questo caso poi, la compilazione risulta essere assai più semplice rispetto a quella vista nel paragrafo 5.3: non dovendo infatti inserire alcun elemento multimediale, basta creare il solo file .ulx. Avviate allora Jif, aprite il file RuinsDM4it.inf, assicuratevi che Jif sia in Glulx Mode e cliccate, con il tasto sinistro del mouse, il pulsante della barra degli strumenti e, per eseguirlo, il pulsante:



[8] Per vedere all'opera Glulx, vi consiglio vivamente di giocare alle avventure Little Falls e Beyond.
[9] Che devono peraltro essere numerati in un modo particolare. In caso contrario, vengono presi da un file .blb presente nella stessa directory o non vengono semplicemente visualizzati.

In seguito poi, una volta creati e inseriti tutti gli elementi multimediali del caso, si può allora compilare il tutto nel formato .blb (a partire naturalmente dal file .ulx) e ridistribuire l'avventura in una versione multimediale. Inoltre, il funzionamento di WinGlulxe è molto simile a quello di Windows Frotz 2002 (entrambi sono stati infatti programmati dalla stessa persona) e non si dovrebbe quindi avere nessuna (o quasi) difficoltà nel passare da un interprete all'altro.

§5.6 La grafica e il sonoro

Ora che sappiamo come compilare ed eseguire un listato in Glulx, possiamo finalmente esaminare il primo file di esempio (sgw_test_it.inf):

```
Constant Story "Test per sgw";
Constant Headline
            "^Simple Glulxe Wrapper (sgw)
            ^di Alessandro Schillaci^^";

Include "Parser";
Include "sgw.h"; ! Inclusione della libreria SGW
Include "Infglk";

Object LibraryMessages
  with before [;
    LMODE1 : lookmode = 2; Message_lmode(1); rtrue;
    LMODE2 : lookmode = 2; Message_lmode(2); rtrue;
    LMODE3 : lookmode = 2; Message_lmode(3); rtrue;
  ];

Include "VerbLib";
Include "Replace";
Include "sgw_test_it.bli";

! ---------------------------------------------------------------------
Object room_1 "room 1"
  with description [;
          print "Una semplice stanza di esempio. Stanza 1.^";
          viewImageLeft(room1);
          rtrue;
        ],
        cant_go "Puoi muoverti solo a nord, verso la stanza 2.",
        n_to room_2,
  has  light;

Object room_2 "room 2"
  with description [;
          print "Una semplice stanza di esempio. Stanza 2.^";
          viewImageLeft(room2);
          rtrue;
        ],
        cant_go "Puoi muoverti a nord verso la stanza 3 oppure a sud,
                 verso la stanza 1.",
        n_to room_3,
        s_to room_1,
  has  light;

Object room_3 "room 3"
```

```
  with description [;
          print "Una semplice stanza di esempio. Stanza 3.^";
          viewImageLeft(room3);
          rtrue;
        ],
        cant_go "Puoi muoverti solo a sud, verso la stanza 2.",
        s_to room_2,
  has  light;

! ---------------------------------------------------------------------
[ Initialise;
     initializeSGW(240);  ! Imposta la finestra grafica alta 240 pixel
     location = room_1;
     lookmode = 2;
     viewImageLeft(room1);
     playSound(chan1,rumble,1,VOLUME_HIGH);
     playSound(chan2,heart,-1,VOLUME_NORMAL);
];

[Message_lmode x;
     if (x == 1 or 3) print " supporta solo la modalit@`a ~completa~,
          che d@`a descrizioni lunghe per tutti i luoghi (anche se gi@`a
          visitati).^";
     else print " @`e gi@`a in modalit@`a completa.^";
];

! ---------------------------------------------------------------------
Include "ItalianG";
```

Il risultato è quello visto nella figura del paragrafo 5.4. Nella funzione Initialise, l'istruzione initializeSGW inizializza gli oggetti della libreria e crea una finestra grafica per le immagini con altezza uguale a n pixel. Ovviamente, È POSSIBILE UTILIZZARE UN ALTRO VALORE PER CAMBIARE LE DIMENSIONI DELLE IMMAGINI, A CONDIZIONE PERÒ CHE QUESTE SIANO NEL FORMATO JPG O PNG E CHE ABBIANO UN'ALTEZZA UGUALE AL VALORE SPECIFICATO (le immagini usate in questo esempio hanno una dimensione di 620x240 pixel – da qui il valore 240).
L'istruzione playSound è invece dedicata AI SUONI, CHE DEVONO ESSERE NEL FORMATO AIF; per funzionare, ha bisogno di ben quattro parametri nell'ordine: il tipo di canale (music per la musica oppure chan1 e chan2 per gli effetti), il nome del suono che deve essere eseguito, il numero di volte che il suono deve essere riprodotto (-1 = all'infinito) e il volume che può essere a sua volta alto (VOLUME_HIGH), normale (VOLUME_NORMAL) o basso (VOLUME_LOW).
L'istruzione viewImageLeft, infine, è quella che visualizza l'immagine alla sinistra dello schermo[10].
Attenzione però: POICHÉ GLULX IN MODALITÀ STANDARD (QUELLA CIOÈ NORMALE), CONSIDERA LA DESCRIPTION DI UNA STANZA UNA SOLA VOLTA, PER LA VISUALIZZAZIONE DELLE IMMAGINI OCCORRE NECESSARIAMENTE IMPOSTARE LA MODALITÀ COMPLETA NELLA FUNZIONE INITIALISE (ASSEGNANDO, COME ABBIAMO GIÀ VISTO NEL CAPITOLO 3, IL VALORE 2 ALLA VARIABILE DI LIBRERIA LOOKMODE) E DISABILITARE IN MANIERA PERMANENTE LE ALTRE DUE MODALITÀ (LA BREVE E LA NORMALE)[11].
Solo così si ha la certezza che le immagini vengano visualizzate sempre (e non solo una volta come accade nella modalità normale o addirittura mai nella modalità breve).

[10] In realtà, tutte queste istruzioni non esistono di default in Glulx, perché fanno parte della libreria realizzata da Alessandro Schillaci (la sgw.h, per l'appunto) ma in compenso semplificano di parecchio la gestione della grafica e del sonoro (e vi consiglio quindi di usarle).
[11] In realtà è anche possibile inserire l'istruzione viewImageLeft nelle funzioni di direzione di una stanza anziché nella sua description. Purtroppo però, l'immagine non viene più visualizzata se il giocatore usa il comando GUARDA O G.

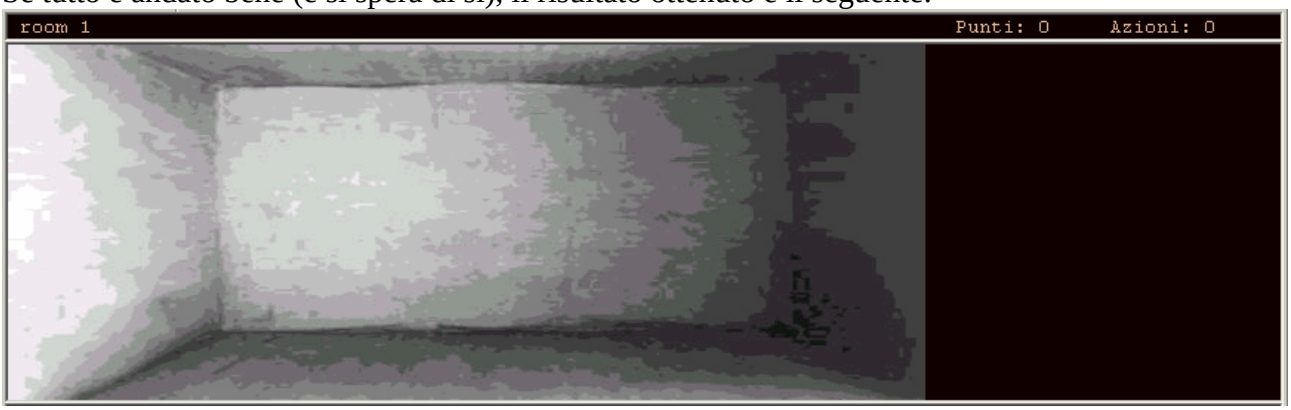
§5.7 L'allineamento delle immagini

Passiamo ora al secondo file di esempio (sgw_test_it_ALIGNMENT):

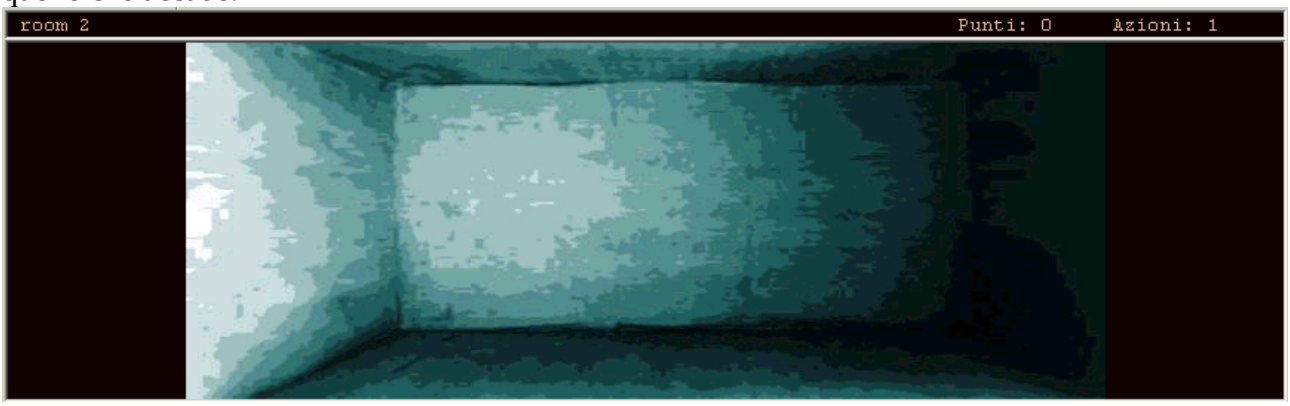
```
Constant Story "Test per sgw";
Constant Headline
            "^Simple Glulxe Wrapper (sgw)
            ^di Alessandro Schillaci^^";

Include "Parser";
Include "sgw.h"; ! Inclusione della libreria SGW
Include "Infglk";

Object LibraryMessages
  with before [;
    LMODE1 : lookmode = 2; Message_lmode(1); rtrue;
    LMODE2 : lookmode = 2; Message_lmode(2); rtrue;
    LMODE3 : lookmode = 2; Message_lmode(3); rtrue;
  ];
Include "VerbLib";
Include "Replace";
Include "sgw_test_it_ALIGNMENT.bli";

! ---------------------------------------------------------------------
Object room_1 "room 1"
  with description [;
          print "Una semplice stanza di esempio. Stanza 1.^";
          viewImageLeft(room1);
          rtrue;
       ],
       cant_go "Puoi muoverti solo a nord, verso la stanza 2.",
       n_to room_2,
  has  light;

Object room_2 "room 2"
  with description [;
          print "Una semplice stanza di esempio. Stanza 2.^";
          viewImageCenter(room2, 620);
          rtrue;
       ],
       cant_go "Puoi muoverti a nord verso la stanza 3 oppure a sud,
verso la stanza 1.",
       n_to room_3,
       s_to room_1,
  has  light;

Object room_3 "room 3"
  with description [;
          print "Una semplice stanza di esempio. Stanza 3.^";
          viewImageRight(room3, 620);
          rtrue;
       ],
       cant_go "Puoi muoverti solo a sud, verso la stanza 2.",
       s_to room_2,
  has  light;

! ---------------------------------------------------------------------
[ Initialise;
     initializeSGW(240);  ! Imposta la finestra grafica alta 240 pixel
     location = room_1;
     lookmode = 2;
```

```
     viewImageLeft(room1); ! Quando entro in una stanza ne visualizzo
l'immagine relativa
     playSound(chan1,rumble,1,VOLUME_HIGH);
     playSound(chan2,heart,-1,VOLUME_NORMAL);
];
[Message_lmode x;
     if (x == 1 or 3) print " supporta solo la modalit@`a ~completa~,
          che d@`a descrizioni lunghe per tutti i luoghi (anche se gi@`a
          visitati).^";
     else print " @`e gi@`a in modalit@`a completa.^";
];
! ----------------------------------------------------------------------
Include "ItalianG";
```

Se tutto è andato bene (e si spera di sì), il risultato ottenuto è il seguente:



Quando siamo nella prima stanza, viene chiamata la funzione viewImageLeft e di conseguenza la prima immagine viene visualizzata alla sinistra dello schermo. Se ci dirigiamo a nord, ecco invece quello che accade:



Quando siamo nella seconda stanza, viene chiamata la funzione viewImageCenter (alla quale si deve dare anche il valore della larghezza dell'immagine ⟶ in questo caso 620) e di conseguenza la seconda immagine viene visualizzata al centro dello schermo. Dirigendoci infine ancora a nord:

ci ritroviamo nella terza e ultima stanza, dove viene chiamata la funzione viewImageRight (alla quale occorre dare, come per la precedente, il valore 620 relativo alla larghezza dell'immagine) che visualizza la terza immagine alla destra dello schermo.

§5.8 Avventure senza grafica

Passiamo ora al terzo file di esempio (sgw_test_it_NOGRAPH):

```
Constant Story "Test per sgw";
Constant Headline
            "^Simple Glulxe Wrapper (sgw)
            ^di Alessandro Schillaci^^";

Constant NOGRAPHICS;

Include "Parser";
Include "sgw.h"; ! Inclusione della libreria SGW
Include "Infglk";
Include "VerbLib";
Include "Replace";
Include "sgw_test_it_NOGRAPH.bli";

! ---------------------------------------------------------------------
Object room_1 "room 1"
  with description "Una semplice stanza di esempio. Stanza 1.",
       cant_go "Puoi muoverti solo a nord, verso la stanza 2.",
       n_to room_2,
  has  light;

Object room_2 "room 2"
  with description "Una semplice stanza di esempio. Stanza 2.",
       cant_go "Puoi muoverti a nord verso la stanza 3 oppure a sud,
                verso la stanza 1.",
       n_to room_3,
       s_to room_1,
  has  light;

Object room_3 "room 3"
  with description "Una semplice stanza di esempio. Stanza 3.",
       cant_go "Puoi muoverti solo a sud, verso la stanza 2.",
       s_to room_2,
  has  light;

! ---------------------------------------------------------------------
[ Initialise;
     initializeSGW(0);
     location = room_1;
     playSound(chan1,rumble,1,VOLUME_HIGH);
     playSound(chan2,heart,-1,VOLUME_NORMAL);
];

! ---------------------------------------------------------------------
Include "ItalianG";
```

Il risultato ottenuto è quello mostrato in figura:

ovvero lo stesso esercizio di prima privato però della grafica. Per farlo, basta semplicemente definire all'inizio del programma la costante NOGRAPHICS.

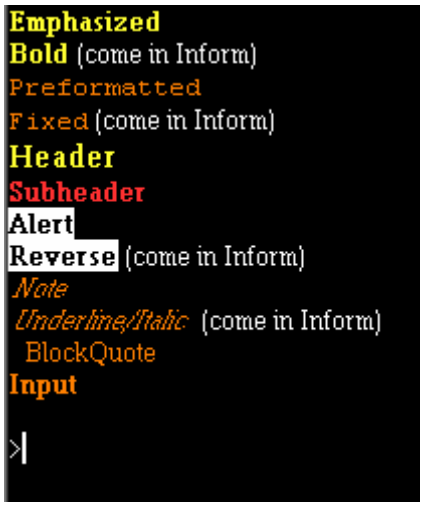
§5.9 I colori e gli stili del testo

Esaminiamo ora il quarto esempio (sgw_test_it_STYLE):

```
Constant Story "Test per sgw";
Constant Headline
           "^Simple Glulxe Wrapper (sgw)
           ^di Alessandro Schillaci^^";

Constant NOGRAPHICS;
Constant NOSOUND;

Constant SCBACK      CLR_GG_BLACK;
Constant SCTEXT      CLR_GG_WHITE;
Constant SCEMPH      CLR_GG_YELLOW;
Constant SCHEAD      CLR_GG_RED;
Constant SCINPU      CLR_GG_ORANGE;

Include "Parser";
Include "sgw.h"; ! Inclusione della libreria SGW
Include "Infglk";
Include "VerbLib";
Include "Replace";
Include "sgw_test_it_STYLE.bli";
! ----------------------------------------------------------------------
Object room_1 "room 1"
  with description [;
            print (s_emph) "Emphasized", "^";
            print (s_bold) "Bold", " (come in Inform)^";
            print (s_pref) "Preformatted", "^";
            print (s_fixed) "Fixed", " (come in Inform)^";
            print (s_head) "Header", "^";
            print (s_subhead) "Subheader", "^";
            print (s_alert) "Alert", "^";
            print (s_reverse) "Reverse", " (come in Inform)^";
            print (s_note) "Note", "^";
            print (s_underline) "Underline/Italic", " (come in
                                Inform)^";
            print (s_block) "BlockQuote", "^";
            print (s_input) "Input", "^";
       ],
```

```
  has  light;
! ----------------------------------------------------------------------
[ Initialise;
      initializeSGW(0);
      location = room_1;
];
! ----------------------------------------------------------------------
Include "ItalianG";
```

Il risultato che si ottiene è il seguente:



ovvero un esercizio che illustra alcuni colori e tutti gli stili del testo che si possono ottenere con Glulx. Per quanto riguarda invece i primi, questa estensione ne prevede di default ben dodici:

ARANCIONE	CLR_GG_ORANGE
AZZURRO	CLR_GG_AZURE
BIANCO	CLR_GG_WHITE
BLU	CLR_GG_BLUE
CIANO	CLR_GG_CYAN
GIALLO	CLR_GG_YELLOW
GRIGIO	CLR_GG_GREY
MAGENTA	CLR_GG_MAGENTA
MARRONE	CLR_GG_BROWN
NERO	CLR_GG_BLACK
ROSA	CLR_GG_PINK
ROSSO	CLR_GG_RED
VIOLA	CLR_GG_PURPLE
VERDE	CLR_GG_GREEN

Anche se i colori mostrati in questa tabella sono quattordici, in realtà il magenta e il ciano sono gli equivalenti rispettivamente del viola e dell'azzurro. A ogni modo, le costanti alle quali è possibile assegnarli, sono sei[12]:

```
Constant SCBACK      CLR_GG_BLACK;
Constant SCTEXT      CLR_GG_WHITE;
Constant SCEMPH      CLR_GG_YELLOW;
Constant SCHEAD      CLR_GG_RED;
Constant SCINPU      CLR_GG_ORANGE;
Constant SCSOFT      CLR_GG_BLUE;
```

[12] La gestione dei colori in Glulx è diversa da Inform; qui infatti, la variabile clr_on (usata da quest'ultimo per stabilire se attivare o meno i colori come spiegato nel paragrafo 4.5) è totalmente ignorata.

La prima (SCBACK), controlla il colore dello sfondo; la seconda (SCTEXT) controlla i colori del testo normale e nello stile alert/reverse; la terza (SCEMPH) controlla i colori del testo negli stili emphasized/bold (il grassetto) e header (il titolo del gioco); la quarta (SCHEAD) controlla i colori del testo nello stile subheader (il titolo della stanza); la quinta (SCINPU) controlla i colori del testo negli stili preformatted/fixed, note/underline (il corsivo), blockquote e input (il testo digitato dal giocatore); la sesta (SCSOFT) controlla i colori degli stili personalizzati (che sono usati molto di rado).
Oltre ai colori standard, e possibile definirne molti altri. Glulx, infatti, prevede che questi siano dichiarati nel formato esadecimale come nel seguente esempio:

```
Constant SCBACK      $110101;
Constant SCTEXT      $DDBB99;
Constant SCSOFT      $665544;
Constant SCEMPH      $FFFFDD;
Constant SCHEAD      $EEDDAA;
Constant SCINPU      $DDEEAA;
```

che ripristina i colori originali dei primi tre esempi. Non preoccupatevi comunque: ogni buon programma di grafica infatti[13], riporta sulla sua tavolozza dei colori anche il valore esadecimale di ogni colore da voi selezionato o creato con essa.

E per quanto riguarda gli stili del testo? Diamo un'occhiata più da vicino all'oggetto Room1:

```
Object room_1 "room 1"
  with description [;
            print (s_emph) "Emphasized", "^";
            print (s_bold) "Bold", " (come in Inform)^";
            print (s_pref) "Preformatted", "^";
            print (s_fixed) "Fixed", " (come in Inform)^";
            print (s_head) "Header", "^";
            print (s_subhead) "Subheader", "^";
            print (s_alert) "Alert", "^";
            print (s_reverse) "Reverse", " (come in Inform)^";
            print (s_note) "Note", "^";
            print (s_underline) "Underline/Italic", " (come in
                                Inform)^";
            print (s_block) "BlockQuote", "^";
            print (s_input) "Input", "^";
       ],
  has  light;
```

Lo stile Emphasized è stampato a video dalla funzione di stampa (s_emph); lo stile Bold (equivalente a Emphasized) da (s_bold); lo stile Preformatted da (s_pref); lo stile Fixed (equivalente a Preformatted) da (s_fixed); lo stile Header da (s_head); lo stile Subheader da (s_subhead); lo stile Alert da (s_alert); lo stile Reverse (equivalente ad Alert) da (s_reverse); lo stile Note da (s_note); lo stile Underline (equivalente a Note) da (s_underline); lo stile BlockQuote da (s_block); lo stile Input, infine, da (s_input).

[13] Uno di questi è sicuramente Paint Shop Pro della Jasc Software, che vi consiglio assolutamente di provare.

§5.10 La personalizzazione della status line

Così come per Inform, anche in Glulx è possibile personalizzare la status line. Diamo quindi un'occhiata al quinto e ultimo esempio (sgw_test_it_SLINE):

```
Constant Story "Test per sgw";
Constant Headline
           "^Simple Glulxe Wrapper (sgw)
           ^di Alessandro Schillaci^^";

Constant NOGRAPHICS;
Constant NOSOUND;

! Colori di default all'apertura di WinGlulxe
Constant SCBACK      $ffffff;
Constant SCTEXT      $000000;
Constant SCEMPH      $000000;
Constant SCHEAD      $000000;
Constant SCINPU      $000000;
Constant SCSOFT      $000000;

Replace DrawStatusLine;

Include "Parser";
Include "sgw.h"; ! Inclusione della libreria SGW
Include "Infglk";
Include "VerbLib";
Include "Replace";
Include "sgw_test_it_SLINE.bli";

! ----------------------------------------------------------------------
Object room_1 "room 1"
  with description "Una semplice stanza di esempio. Stanza 1.",
       cant_go "Puoi muoverti solo a nord, verso la stanza 2.",
       n_to room_2,
  has  light;

Object room_2 "room 2"
  with description "Una semplice stanza di esempio. Stanza 2.",
       cant_go "Puoi muoverti a nord verso la stanza 3 oppure a sud,
                verso la stanza 1.",
       n_to room_3,
       s_to room_1,
  has  light;

Object room_3 "room 3"
  with description "Una semplice stanza di esempio. Stanza 3.",
       cant_go "Puoi muoverti solo a sud, verso la stanza 2.",
       s_to room_2,
  has  light;

! ----------------------------------------------------------------------
[ Initialise;
    initializeSGW(0);
    location = room_1;
];

[ DrawStatusLine width height;
```

```
   ! Se non abbiamo una riga di stato, non dobbiamo cercare di
   ! ridisegnarla.
   if (gg_statuswin == 0)
      return;

   ! Non dobbiamo farlo neanche se non esiste una locazione per il
   ! giocatore
   if (location == nothing || parent(player) == nothing)
      return;

   StatusLineHeight(1); ! numero di righe della status line
   glk_set_window(gg_statuswin);
   glk_window_clear(gg_statuswin);
   glk_window_get_size(gg_statuswin, gg_arguments, gg_arguments+4);
   width = gg_arguments-->0; if (width == 0) width = 100;
   height = gg_arguments-->1; if (height == 0) height = 1;

   glk_window_move_cursor(gg_statuswin, 1, 0);
   if (location == thedark) {
         print (name) location;
   }
   else {
         FindVisibilityLevels();
         if (visibility_ceiling == location)
             print (name) location;
         else
             print (The) visibility_ceiling;
   }

   glk_window_move_cursor(gg_statuswin, 74, 0);
   print "Punteggio: ", score;
   glk_window_move_cursor(gg_statuswin, 93, 0);
   print "Turni: ", turns;
   glk_set_window(gg_mainwin);
];

! ----------------------------------------------------------------------
Include "ItalianG";
```

Così come per Inform, anche in Glulx occorre rifarsi all'istruzione Replace DrawStatusLine; seguita poi dalla definizione della funzione DrawStatusLine. L'istruzione:

```
glk_window_move_cursor(gg_statuswin, 1, 0);
```

posiziona il testo che segue (in questo caso il nome della locazione) alla posizione 1 della prima riga:

Il resto, a questo punto del manuale, dovrebbe ormai esservi abbastanza chiaro; comunque, per chi non l'avesse ancora letto, il paragrafo 4.1 contiene ulteriori informazioni sull'argomento.

§5.11 L'input-output dei dati

Tutte le funzioni per l'input-output dei dati che ho creato e descritto nel capitolo 2, funzionano anche sotto Glulx. Esse sono tutte raggruppate nella libreria io.h e sono nell'ordine:

- CmpStr, che confronta due stringhe restituendo 1 se sono uguali, viceversa 0 (ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 2.15);
- GetNumber, che legge in input un valore numerico digitato dall'utente (max 5 cifre) con un range che va - sotto Glulx - da 0 a 99999 (ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 2.7);
- Presskey, che legge il carattere di un tasto qualsiasi premuto dall'utente senza la pressione del tasto di Invio (ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 2.14);
- PrintNumericArray, che stampa a video il contenuto di un vettore numerico (ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 2.15);
- PrintStringArray, che stampa a video un vettore di tipo stringa. Prima di chiamarla, caricate il vettore con l'istruzione PrintToBuffer(array, lunghezza, stringa);. Le dimensioni del vettore devono essere calcolate in base alla lunghezza massima del testo che può contenere (+5 per Glulx) → es: se l'array deve contenere un testo lungo al massimo 20 caratteri, dovrà essere definito nel modo seguente: Array nome_array->25; (ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 2.15);
- ReadArray, che legge una stringa inserita dall'utente. Occorre passarle, come parametro, il vettore nel quale memorizzare la stringa e la sua lunghezza massima, tenendo sempre conto del +5. (ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 2.15);
- Readchar, che legge un carattere digitato dalla tastiera con la pressione del tasto di Invio (ulteriori informazioni sul suo utilizzo potete trovarle al paragrafo 2.15).

Naturalmente, oltre alle funzioni qui descritte, potete anche usare tutti gli esercizi del capitolo 2 (a condizione, ovviamente, che impostiate Jif in Glulx Mode).

§5.12 Varie

Ci sono ancora alcune cose da considerare su questo linguaggio di programmazione, nell'ordine:

- la sgw.h prevede delle istruzioni che in questi esempi non sono state usate. Esse sono nell'ordine: clearMainWindow (che ripulisce la finestra principale), closeAllWindows (che chiude tutte le finestre grafiche), silenceAll (che azzera tutti i canali audio), silenceChannel (che azzera il canale audio specificatole come parametro) e setVolume[val, channel] (che imposta il valore "val" per il volume da impostare sul canale specificato);
- la gestione dei menu può essere tranquillamente effettuata tramite l'estensione Dmenus.h vista nel paragrafo 4.9;
- tutte le estensioni viste nel paragrafo 4.9 (fatta eccezione per la style.h e la boxclever.h) funzionano correttamente anche sotto Glulx;
- Reform, il decompilatore per i file nel formato Z-code, è anche in grado di decompilare i file blorb (quelli cioè contrassegnati dall'estensione .blb);
- anche qui, come per Inform, sono state previste delle estensioni per arricchire ulteriormente le capacità di questo linguaggio di programmazione. Potete trovarle tutte all'indirizzo http://www.inform-fiction.org/extensions/glulx.html;
- Glulx è in grado di gestire anche l'input del mouse, i collegamenti ipertestuali, le pause e il real time. Ulteriori informazioni sul loro utilizzo potete trovarle su Gull, come spiegato nel prossimo paragrafo.

§5.13 Gull, il manuale ufficiale

Le potenzialità di Glulx non si fermano di certo qui. Chi è interessato ad approfondire questo linguaggio deve necessariamente fare riferimento a Gull, il manuale ufficiale scritto da Adam Cadre e tradotto in italiano dal bravissimo Paolo Vece. Per leggerlo, dovete decomprimere il file gull.zip in una directory qualsiasi e aprire, con un browser per Internet (non necessariamente Internet Explorer), il file index.html[14]. Altre informazioni molto utili su questo linguaggio potete trovarle all'indirizzo http://www.eblong.com/zarf/glulx.

Se volete invece provare a giocare a qualche avventura italiana scritta con questo linguaggio, potete scaricare Schizo (di Tommaso Caldarola), Little Falls (di Alessandro Schillaci e Roberto Grassi) e la stratosferica Beyond. Se conoscete invece l'inglese (e perché no, anche lo spagnolo) andate sul sito di Wurb alla sezione Glulx.

Buon divertimento…

[14] Altri due manuali molto utili sono la Guida a Inform-Glulx di Marco Falcinelli e la Guida a Inform-Glulx di Andrew Plotkin.

APPENDICI

APPENDICE A – LE AZIONI PRICIPALI[1]

Inform, a differenza d'altri linguaggi di programmazione, prevede di default molte azioni che sono qui elencate. Abbiamo visto nel capitolo 3 come usarle e come crearne delle nuove; a ogni modo, riassumiamo qui alcune regole principali:

1. Inform suddivide le azioni in tre gruppi (1 per i comandi di sistema, 2 per le azioni controllate dalla proprietà after e 3 per quelle controllate dalla proprietà before);
2. non è necessariamente detto che un'azione del gruppo 2 non possa appartenere anche al gruppo 3 e viceversa. Solo la pratica può darci la risposta giusta; in poche parole, quando si definisce un'azione per un dato oggetto, prima la si prova sotto la relativa proprietà di appartenenza. Se funziona, tutto ok, altrimenti si prova con l'altra;
3. alcune azioni (come ad es. chiudi o prendi) hanno diverse modalità di utilizzo.

Prendiamo ora la definizione di un oggetto qualsiasi e cerchiamo di capire meglio quanto sto dicendo:

```
Object  -> mushroom "fungo macchiato"
  with  name 'macchiato' 'macchiettato' 'fungo' 'velenoso',
        initial
            "Un fungo macchiato cresce dalla terra fradicia, su un gambo
             lungo.",
        description
            "Il fungo @`e ricoperto da delle macchie e non sei del tutto
             sicuro che non sia velenoso.",
        after [;
            Take:
                if (self.mushroom_picked)
                    "Hai raccolto il fungo mezzo marcio.";
                self.mushroom_picked = true;
                "Hai raccolto abilmente il fungo, senza staccarlo dal suo
                 gambo sottile.";
         .
         .
         .
```

A cosa corrisponde quindi l'azione Take in italiano?
Andando a verificare nella tabella, vediamo che le azioni corrispondenti sono Afferra, Prendi e Raccogli (si potrà quindi dire PRENDI [OGGETTO], AFFERRA [OGGETTO] e RACCOGLI [OGGETTO]). Non lasciatevi però ingannare dalle apparenze, perché NON È SEMPRE DETTO CHE A UN'AZIONE INGLESE CORRISPONDA NECESSARIAMENTE QUELLA TRADOTTA IN ITALIANO. Prendiamo ad esempio look under: in italiano dovrebbe esserci un'azione chiamata guarda sotto mentre c'è solo guarda. Potrò quindi dare il comando GUARDA SOTTO solo se testerò in Inform l'azione inglese LookUnder piuttosto che Look (che corrisponde anch'essa all'azione italiana guarda ma con un risultato completamente diverso).
Quindi, quando cercate un'azione in italiano, non guardate solo la prima colonna a sinistra ma anche l'ultima a destra (quella degli esempi, tanto per intenderci), soprattutto per quei comandi che prevedono più modalità di utilizzo.

N.B.: Infit 2.5 supporta di default il verbo Siediti ma non il verbo Alzati. Per capire come implementarlo, potete studiare l'esercizio del trono descritto al paragrafo 3.15 del manuale.

Adesso vi lascio finalmente all'ELENCO DELLE AZIONI PRINCIPALI, definite nel file ItalianG.h[2].

[1] L'elenco delle azioni che segue è quello relativo a Infit 2.5 e può variare a seconda della versione di Infit utilizzata.
[2] A differenza del file ItalianG.h (all'interno del quale sono definite tutte le azioni possibili in Inform), quest'elenco riporta solo quelle maggiormente utilizzate.

ITALIANO	INGLESE	GRUPPO	ESEMPIO
Abbraccia	Kiss	3	“abbraccia la ragazza”
Accarezza	Touch	3	“accarezza il cavallo”
Accendi	SwitchOn	2	“accendi la lampada”
Afferra	Take	2	“afferra la maniglia”
Affetta	Cut	3	“affetta il salame”
Agita	Wave	3	“agita la bacchetta”
Ammazza (1)	Attack	3	“ammazza il drago”
Ammazza (2)	Attack	3	“ammazza il drago con la spada”
Annoda	Tie	3	“annoda la corda al gancio”
Annusa	Smell	3	“annusa l’aria”
Apri (1)	Open	2	“apri la porta”
Apri (2)	Unlock	2	“apri la porta con la chiave”
Arrampicati	Climb	3	“arrampicati sull’albero”
Ascolta	Listen	3	“ascolta”
Aspetta	Wait	2	“aspetta” o “z”
Assaggia	Taste	3	“assaggia la minestra”
Attacca (1)	Attack	3	“attacca il drago”
Attacca (2)	Attack	3	“attacca il drago con la spada”
Attacca (3)	Tie	3	“attacca il quadro alla parete”
Attiva	SwitchOn	2	“attiva l’allarme”
Bacia	Kiss	3	“bacia la ragazza”
Bevi	Drink	3	“bevi l’acqua”
Blocca	Lock	2	“blocca la porta con il paletto”
Brucia	Burn	3	“brucia il giornale”
Cammina	Go	2	“cammina verso sud”
Canta	Sing	3	“canta”
Carica	Restore	1	“carica” o “caricare”
Cerca (1)	Search	2	“cerca nella cassa”
Cerca (2)	Consult	3	“cerca freccia nel dizionario”
Chiedi (1)	Ask	3	“chiedi al sacerdote delle rovine”
Chiedi (2)	AskFor	3	“chiedi al sacerdote il dizionario”
Chiudi (1)	Close	2	“chiudi la porta”
Chiudi (2)	Lock	2	“chiudi la porta a chiave”
Colpisci (1)	Attack	3	“colpisci il nano”
Colpisci (2)	Attack	3	“colpisci il nano con l’ascia”
Consulta	Consult	3	“consulta il dizionario sul simbolo”
Combatti (1)	Attack	3	“combatti il drago”
Combatti (2)	Attack	3	“combatti il drago con la spada”
Compra	Buy	3	“compra le mele”
Corri (1)	Enter	2	“corri nella casa”
Corri (2)	Climb	3	“corri sulla collina”
Corri (3)	Exit	2	“corri fuori”
Corri (4)	GoIn	2	“corri dentro”
Corri (5)	Go	2	“corri verso/a nord”
Dai	Give	3	“dai una moneta al mendicante”
Disattiva	SwitchOff	2	“disattiva l’allarme”
Dormi	Sleep	3	“dormi”
Entra	Enter	2	“entra dentro/nella caverna”
Esamina	Examine	2	“esamina la statua” o “x statua”
Esci	Exit	2	“esci dalla stanza”
Fai (1)	Inv	2	“fai l’inventario”

ITALIANO	INGLESE	GRUPPO	ESEMPIO
Fai (2)	Show	3	“fai vedere alla mummia il giornale”
Fine	Quit	1	“fine” o “q”
Fissa	Tie	3	“fissa la corda al gancio”
Gira	Turn	3	“gira la manopola”
Guarda (1)	Look	2	“guarda” o “g”
Guarda (2)	Search	2	“guarda dentro/nella cassa”
Guarda (3)	LookUnder	3	“guarda sotto il letto”
Inserisci	Insert	2	“inserisci la chiave nella serratura”
Incendia	Burn	3	“incendia la casa”
Indossa	Wear	2	“indossa la maschera”
Inventario	Inv	2	“inventario” o “i” o “inv”
Inventario esteso	InvTall	2	“inventario esteso”
Inventario completo	InvWide	2	“inventario completo”
Lancia	ThrowAt	3	“lancia l’osso al cane”
Lascia	Drop	2	“lascia il giornale”
Lega	Tie	3	“lega la corda al gancio”
Leggi	Examine	2	“leggi il giornale”
Lucida	Rub	3	“lucida le monete”
Lustra	Rub	3	“lustra le monete”
Mangia	Eat	2	“mangia il fungo”
Metti (1)	Insert	2	“metti la chiave nella serratura”
Metti (2)	PutOn	2	“metti la maschera sull’altare”
Modalità breve	LMode3	1	“modalità breve”
Modalità normale	LMode1	1	“modalità normale”
Modalità completa	LMode2	1	“modalità completa”
Mostra	Show	3	“mostra il dizionario al sacerdote”
Muovi (1)	Push	3	“muovi il tavolo”
Muovi (2)	PushDir	3	“muovi il tavolo a/verso sud”
Notify on	NotifyOn	1	“notifica attivata”
Notify off	NotifyOff	1	“notifica disattivata”
Nuota	Swim	3	“nuota”
Offri	Give	3	“offri una moneta al mendicante”
Oggetti	Objects	1	“oggetti”
Ordina	Order	3	“sacerdote, vai a sud”
Parla (1)	Tell	3	“parla con il sacerdote”
Parla (2)	Tell	3	“parla con il sacerdote delle rovine”
Pela	Take	2	“pela le patate”
Pensa	Think	3	“pensa”
Picchia (1)	Attack	3	“picchia il nano”
Picchia (2)	Attack	3	“picchia il nano con la mazza”
Posa (1)	Drop	2	“posa il giornale”
Posa (2)	Insert	2	“posa il giornale nella cassa”
Posa (3)	PutOn	2	“posa il giornale sul tavolo”
Posti	Places	1	“posti” o “luoghi”
Prega	Pray	3	“prega”
Premi	Push	3	“premi il pulsante”
Prendi (1)	Take	2	“prendi la maschera”
Prendi (2)	Remove	2	“prendi il biscotto dalla scatola”
Pronomi	Pronouns	1	“pronomi”
Pulisci	Rub	3	“pulisci il tavolo”
Punteggio	Score	1	“punteggio”
Punteggio completo	FullScore	1	“punteggio completo”

Punteggio pieno	FullScore	1	“punteggio pieno”
Raccogli	Take	2	“raccogli la moneta”
Recording on	CommandsOn	1	“recording on”
Recording off	CommandsOff	1	“recording off”
Replay	CommandsRead	1	“replay”
Ricomincia	Restart	1	“ricomincia” o “ricominciare”
Riempi	Fill	3	“riempi la bottiglia”
Rifletti	Think	3	“rifletti”
Rimuovi (1)	Disrobe	2	“rimuovi il giornale”
Rimuovi (2)	Remove	2	“rimuovi il giornale dalla cassa”
Rispondi	Answer	3	“rispondi sì al nano”
Rompi (1)	Attack	3	“rompi il vetro”
Rompi (2)	Attack	3	“rompi il vetro con il martello”
Rovescia (1)	Empty	2	“rovescia il vino”
Rovescia (2)	EmptyT	2	“rovescia il vino sulla tovaglia”
Ruota	Turn	3	“ruota la manopola”
Sali	Climb	3	“sali sulla scala”
Salta (1)	Jump	3	“salta”
Salta (2)	JumpOver	3	“salta il baratro”
Saluta	WaveHands	3	“saluta il nano”
Salva	Save	1	“salva” o “salvare”
Sblocca	Unlock	2	“sblocca la serratura con il grimaldello”
Sbuccia	Take	2	“sbuccia la mela”
Scassina	Unlock	2	“scassina la serratura con il grimaldello”
Scava	Dig	3	“scava la fossa con la pala”
Scendi	Exit	2	“scendi dal tavolo”
Schiaccia	Squeeze	3	“schiaccia il pulsante”
Script on	ScriptOn	1	“script on”
Script off	ScriptOff	1	“script off”
Scuoti	Swing	3	“scuoti l’albero”
Sdraiati	Enter	2	“sdraiati sul divano”
Serra	Lock	2	“serra la porta con il paletto”
Siediti	Enter	2	“siediti sulla sedia”
Soffia	Blow	3	“soffia nel fischietto”
Sorseggia	Drink	3	“sorseggia l’aperitivo”
Spacca	Cut	3	“spacca il vetro”
Spegni	SwitchOff	2	“spegni la lampada”
Spingi (1)	Push	3	“spingi il tavolo”
Spingi (2)	PushDir	3	“spingi il tavolo a/verso sud”
Spolvera	Rub	3	“spolvera la scrivania”
Sposta (1)	Push	3	“sposta il tavolo”
Sposta (2)	Transfer	2	“sposta le uova nella cesta”
Spremi	Squeeze	3	“spremi il limone”
Stai	Enter	2	“stai nell’edificio”
Strappa	Cut	3	“strappa la carta”
Strofina	Rub	3	“strofina la lampada”
Sveglia (1)	Wake	3	“svegliati”
Sveglia (2)	WakeOther	3	“sveglia il nano”
Svuota (1)	Empty	2	“svuota il sacco”
Svuota (2)	EmptyT	2	“svuota il sacco nella cassa”
Taglia	Cut	3	“taglia la carta con le forbici”
Tira	Pull	3	“tira la corda”
Tocca	Touch	3	“tocca il dipinto”

Togli	Disrobe	2	“togli la maschera”
Tortura (1)	Attack	3	“tortura il sacerdote”
Tortura (2)	Attack	3	“tortura il sacerdote con la frusta”
Trascina	Pull	3	“trascina il sacco”
Transcript on	ScriptOn	1	“trascrizione attivata”
Transcript off	Script Off	1	“trascrizione disattivata”
Trasferisci	Transfer	2	“trasferisci le uova nella cesta”
Trova	Search	2	“trova nella cassa”
Uccidi (1)	Attack	3	“uccidi il drago”
Uccidi (2)	Attack	3	“uccidi il drago con la spada”
Unisci	Tie	3	“unisci la corda al gancio”
Vai (1)	Exit	2	“vai fuori”
Vai (2)	GoIn	2	“vai dentro”
Vai (3)	Climb	3	“vai sopra il tavolo”
Vai (4)	Go	2	“vai verso/a nord”
Verifica	Verify	1	“verifica” o “verificare”
Versa	EmptyT	2	“versa il vino nel bicchiere”
Versione	Version	1	“versione”

APPENDICE B – I MESSAGGI DELLA LIBRERIA[1]

Inform dà sempre, per ogni azione effettuata dal giocatore, una risposta di default che può però essere personalizzata come spiegato alla fine del paragrafo 3.6 del manuale. In questo modo è possibile evitare di andare a modificare direttamente le librerie, azione che, oltre che pericolosa per Inform in sé, non è legale (non è possibile cioè modificare e distribuire le librerie senza l'autorizzazione dei rispettivi autori – Graham Nelson per Inform e Giovanni Riccardi per Infit).
Spesso però, quando si devono personalizzare dei messaggi di sistema, occorre conoscere le funzioni interne di Infit, che sono quelle elencate nella guida di Infit presente all'interno del file infit25.zip.
Una precisazione: nella guida, l'elenco di queste funzioni è rappresentato in questo modo:
(itorthem) obj: ...
(thatorthose) obj: ...
(cthatorthose) obj: ...
(isorare) obj: ...
(cisorare) obj: ...
.
.
.

dove con obj s'intende l'oggetto al quale si riferiscono le funzioni di stampa. Quello che forse non è ancora chiaro ad alcuni di voi, è come usarle in un listato Inform. Il seguente esempio dovrebbe chiarire meglio quello che sto cercando di dire:

```
Drop: print_ret (The)noun," scivola attraverso uno dei cunicoli
                 e ", (itorthem) noun, " perdi rapidamente di vista.";
```

la funzione itorthem stampa lo, la, li, le a seconda del genere (maschile o femminile) e il numero (singolare o plurale). Questo fa sì che Inform "capisca" il tipo di oggetto con cui ha a che fare, e stampi così a video il relativo messaggio corretto dopo che il giocatore lo posa (drop) per terra. Ecco alcuni esempi:

DIZIONARIO → posa il dizionario → Il dizionario scivola attraverso uno dei cunicoli e lo perdi…
MAPPA → posa la mappa → la mappa scivola attraverso uno dei cunicoli e la perdi…
LINGOTTI → posa i lingotti → i lingotti scivolano attraverso uno dei cunicoli e li perdi…
PERGAMENE → le pergamene scivolano attraverso uno dei cunicoli e le perdi…

e così via; obj va inteso quindi come un termine generico, che è stato qui sostituito con noun. Altra cosa da ricordare durante la definizione di un oggetto, è che va assolutamente specificato, a livello di attributo, se esso è femminile (has female) o plurale (has pluralname); se invece è maschile (has male), si può anche non specificare nulla, perché questo valore è già impostato di default.
Occorre anche specificare il relativo articolo: se ad esempio definisco un oggetto che si chiama gradini, dovrò specificare che il suo articolo sarà dei, come accade nel seguente esempio:

```
Object  -> steps "gradini scolpiti nella roccia"
  with  article "dei",
   .
   .
   .
```

[1] Relativi alla versione 6.11.

Tuttavia, non sempre è necessario specificare l'articolo ogni volta che si definisce un nuovo oggetto. Se date un'occhiata al listato di Ruins, vi renderete ben presto conto che la maggior parte di essi saranno privi dell'articolo in questione. L'unica eccezione è data da un oggetto con l'attributo proper (in genere il giocatore stesso, definito in Inform come player's object, oppure un oggetto nel quale il giocatore dovrà trasformarsi, come ad esempio il facocero – warthog – definito in Ruins), perché il suo nome è visto come proprio di persona e non ha mai, di conseguenza, bisogno dell'articolo.

Segue ora l'elenco completo di tutti i messaggi della libreria italiana, definiti all'interno del file Italian.h.

N. B.: i messaggi contrassegnati dal simbolo [G] sono relativi a Glulx.

Answer, Ask	"Nessuna risposta."
Attack	"La violenza non è la giusta risposta a questo."
Blow	"Non c'è niente di utile nel soffiare sul/sullo/sulla/sugli/sull'/sulle [oggetto]."
Burn	"Con questo atto pericoloso non concluderai niente."
Buy	"Non c'è niente in vendita."
Climb	"Non penso si possa raggiungere qualcosa da qui."
Close	1: "Non puoi chiudere il/lo/la/i/gli/le/l' [oggetto]." 2: "È/Sono già chiuso/a/i/e." 3: "Hai chiuso il/lo/la/i/gli/le/l' [oggetto]."
CommandsOff	1: "[Registrazione dei comandi disattivata.]" 2: "[Registrazione dei comandi già disattivata.]" [G]
CommandsOn	1: "[Registrazione dei comandi attivata.]" 2: "[La riproduzione dei comandi è attiva proprio in questo momento.]" [G] 3: "[Registrazione dei comandi già attiva.]" [G] 4: "[Errore nella registrazione dei comandi.]" [G]
CommandsRead	1: "[Riproduzione dei comandi registrati.]" 2: "[I comandi registrati sono già stati riprodotti.]" [G] 3: "[Errore nella riproduzione dei comandi registrati. La registrazione dei comandi è attiva.]" [G] 4: "[Errore nella riproduzione dei comandi registrati.]" [G] 5: "[La riproduzione dei comandi registrati è stata completata.]" [G]
Consult	"Non hai scoperto niente di interessante, consultando il/lo/la/i/gli/le/l' [oggetto]."
Cut	"Tagliarlo/la/li/le a pezzi servirà a poco."
Dig	"Scavare non serve a niente qui."
Disrobe	1: "Non lo/la/li/le stai indossando." 2: "Ti sei tolto il/lo/la/i/gli/le/l' [oggetto]."
Drink	"Non c'è niente che puoi bere qui."
Drop	1: "Il/lo/la/i/gli/le/l' [oggetto] è/sono già posato/a/i/e qui." 2: "Non possiedi quello/quella/quelli/quelle." 3: "(Prima ti togli il/lo/la/i/gli/le/l' [oggetto])" 4: "Posato/a/i/e."
Eat	1: "È/Sono praticamente immangiabile/i." 2: "Mangi il/lo/la/i/gli/le/l' [oggetto]. Niente male."
EmptyT	1: "Il/lo/la/i/gli/le/l' [oggetto] non è un contenitore/non sono contenitori." 2: "Il/lo/la/i/gli/le/l' [oggetto] è/sono chiuso/a/i/e." 3: "Il/lo/la/i/gli/le/l' [oggetto] è/sono già vuoto/a/i/e." 4: "Probabilmente in questo modo non svuoteresti nulla."
Enter	1: "Ma già sei sopra/dentro il/lo/la/i/gli/le/l' [oggetto]." 2: "Non è/Non sono qualcosa su cui poter stare/su cui ti puoi sdraiare/su cui ti puoi sedere/in cui puoi entrare." 3: "Non puoi entrare dentro il/lo/la/i/gli/le/l' [oggetto]. È/Sono chiuso/a/i/e." 4: "Puoi entrare solo in qualcosa di libero." 5: "Ti trovi sopra/dentro il/lo/la/i/gli/le/l' [oggetto]." 6: "(ti togli da sopra/dentro il/lo/la/i/gli/le/l' [oggetto])" 7: "(sali sopra/entri dentro il/lo/la/i/gli/le/l' [oggetto])^"
Examine	1: "Oscurità sost. femminile. Assenza della luce necessaria per vedere. Buio. Tenebre." 2: "Esamini il/lo/la/i/gli/le/l' [oggetto] ma non noti niente di speciale." 3: "Il/lo/la/i/gli/le/l' [oggetto] è/sono acces/spento/a/e/i."
Exit	1: "E da dove? Dovresti spiegarti meglio." 2: "Non puoi uscire fuori, il/lo/la/i/gli/le/l' [oggetto] è/sono chiuso/a/e/i." 3: "Sei sceso/uscito dallo/dalla/dagli/dalle/dall' [oggetto]."

	4: “Cosa? Non sei sopra/dentro il/lo/la/i/gli/le/l’ [oggetto].”
Fill	“Ma non c’è acqua qui da trasportare.”
Fullscore	1: “Il punteggio è/era così composto:^”
	2: “in vari oggetti trovati”
	3: “visitando vari luoghi”
	4: “in totale (su [punteggio massimo] possibili)”
GenericVerb	“Cosa vuoi fare? Potresti essere più preciso?”
GetOff	“Ma non sei sopra il/lo/la/i/gli/le/l’ [oggetto] in questo momento.”
Give	1: “Non possiedi il/lo/la/i/gli/le/l’ [oggetto].”
	2: “Vuoi dare il/lo/la/i/gli/le/l’ [oggetto] a te stesso? Non ti seguo.”
	3: “Il/lo/la/i/gli/le/l’ [oggetto animato] non sembra/non sembrano interessato/a/e/i.”
Go	1: “Dovresti toglierti da sopra/dentro il/lo/la/i/gli/le/l’ [oggetto] prima.”
	2: “Non puoi andare in quella direzione.”
	3: “Non riesci a salire sopra il/lo/la/i/gli/le/l’ [oggetto].”
	4: “Non riesci a scendere da sopra il/lo/la/i/gli/le/l’ [oggetto].”
	5: “Non puoi, il/lo/la/i/gli/le/l’ [oggetto] è/sono in quella direzione.”
	6: “Non puoi, il/lo/la/i/gli/le/l’ [oggetto] non conduce/non conducono da nessuna parte.”
Insert	1: “È necessario che tu possieda il/lo/la/i/gli/le/l’ [primo oggetto] prima di poterlo/a/e/i mettere dentro il/lo/la/i/gli/le/l’ [secondo oggetto].”
	2: “Quello/Quella/Quelli/Quelle non è/non sono contenitori.”
	3: “Il/lo/la/i/gli/le/l’ [oggetto] è/sono chiuso/a/i/e.”
	4: “Sarà necessario che tu lo/la/li/le lasci prima.”
	5: “Non puoi mettere una cosa dentro se stessa.”
	6: “(prima lo/la/li/le lasci)^”
	7: “Non c'è più spazio dentro il/lo/la/i/gli/le/l’ [oggetto].”
	8: “Fatto.”
	9: “Hai messo il/lo/la/i/gli/le/l’ [primo oggetto] dentro il/lo/la/i/gli/le/l’ [secondo oggetto].”
Inv	1: “Non stai portando niente.”
	2: “Stai portando”
	3: “:^”
	4: “.^”
Jump	“Salti sul posto, senza risultato.”
JumpOver	“Non raggiungerai niente con questo.”
Kiss	“Concentrati sul gioco.”
Listen	“Non si sente niente di particolare.”
ListMiscellany	1: “ (acceso/a/e/i)”
	2: “ (che è/sono chiuso/a/e/i)”
	3: “ (chiuso/a/e/i e acceso/a/e/i)”
	4: “ (che è/sono vuoto/a/e/i)”
	5: “ (vuoto/a/e/i e acceso/a/e/i)”
	6: “ (che è/sono chiuso/a/e/i e vuoto/a/e/i)”
	7: “ (chiuso/a/e/i, vuoto/a/e/i e acceso/a/e/i)”
	8: “ (acceso/a/e/i e indossato/a/e/i.)”
	9: “ (acceso/a/e/i”
	10: “ (indossato/a/e/i”
	11: “ (che è/sono ”
	12: “aperto/a/e/i”
	13: “aperto/a/e/i ma vuoto/a/e/i”
	14: “chiuso/a/e/i”
	15: “chiuso/a/e/i a chiave”

	16: " e vuoto/a/e/i" 17: " (che è/sono vuoto/a/e/i)" 18: " che contiene/che contengono " 19: " (sul quale/sulla quale/sui quali/sulle quali " 20: ", sopra il quale/la quale/i quali/le quali " 21: " (nel quale/nella quale/nei quali/nelle quali " 22: ", dentro il quale/la quale/i quali/le quali "
LMode1	"è ora in modalità "normale", che dà descrizioni lunghe per i luoghi mai visitati prima e brevi se già visitati."
LMode2	"è ora in modalità "completa", che dà descrizioni lunghe per tutti i luoghi (anche se già visitati)."
LMode3	"è ora in modalità "breve", che dà descrizioni brevi per tutti i luoghi (anche se mai visitati)."
Lock	1: "Non sembra/Non sembrano essere qualcosa che possa essere chiusa a chiave." 2: "È/Sono chiuso/a/e/i a chiave in questo momento." 3: "Prima dovresti chiudere il/lo/la/i/gli/le/l' [oggetto]." 4: "Non sembra/Non sembrano entrare nella serratura." 5: "Ora hai chiuso a chiave il/lo/la/i/gli/le/l' [oggetto]."
Look	1: " (sopra il/lo/la/i/gli/le/l' [oggetto])" 2: " (dentro il/lo/la/i/gli/le/l' [oggetto])" 3: " (come il/lo/la/i/gli/le/l' [oggetto])" 4: "^Sopra il/lo/la/i/gli/le/l' [oggetto] vedi [lista oggetti]." 5, 6: "|^Puoi| |Sopra il/lo/la/i/gli/le/l' [oggetto] puoi| |Dentro il/lo/la/i/gli/le/l' [oggetto] puoi| |anche| vedere [lista oggetti] |.| |qui.|" 7: "Non vedi niente di strano in quella direzione."
LookUnder	1: "Ma è buio." 2: "Non trovi niente di interessante."
Mild	"Proprio così."
Miscellany	1: "(considero solo i primi sedici oggetti)^" 2: "Niente da fare!" 3: " Sei morto " 4: " Hai vinto " 5: "^Vuoi RICOMINCIARE, CARICARE una partita salvata/, ANNULLARE il tuo ultimo comando/, avere il punteggio COMPLETO della partita/, avere suggerimenti per cose DIVERTENTI da fare/ o USCIRE ?" 6: "[L'interprete non può gestire il comando "cancella".]" 7: ""Cancella" non ha funzionato. [Non tutti gli interpreti lo supportano.]" 7: "[Non puoi più cancellare i comandi.]" [G] 8: "Per favore, dai una delle risposte elencate." 9: "^È completamente buio qui!" 10: "Scusa!!??" 11: "[Non puoi cancellare ciò che non hai fatto!]" 12: "[Non puoi cancellare due volte di seguito. Spiacente!]" 13: "[Azione precedente cancellata.]" 14: "Spiacente, questo non può essere corretto." 15: "Non c'è di che." 16: "Il comando "oops" può correggere solo una parola." 17: "È completamente buio, e non riesci a vedere niente." 18: "te stesso" 19: "Hai sempre lo stesso bell'aspetto." 20: "Per ripetere un comando come "rana, salta", basta solamente "ancora", non "rana, ancora"."

	21: “Difficilmente potrai ripetere quello che hai appena fatto.” 22: “Non puoi iniziare un comando con una virgola.” 23: “Sembra che tu voglia parlare con qualcuno, ma non capisco con chi.” 24: “Non puoi parlare con il/lo/la/i/gli/le/l’ [oggetto].” 25: “Per parlare a qualcuno prova "qualcuno, ciao" o qualcosa di simile.” 26: “(prima prendi il/lo/la/i/gli/le/l’ [oggetto])” 27: “Cosa vuoi dire? Non riesco a capire.” 28: “Ho capito la frase solo fino a: ” 29: “Non riesco a capire il numero.” 30: “Non vedi nulla del genere.” 31: “Mi sembra che tu abbia detto troppo poco!” 32: “Non possiedi questa cosa!” 33: “Non puoi usare oggetti multipli con quel verbo.” 34: “Puoi usare oggetti multipli solo una volta per comando.” 35: “Non riesco a capire a cosa "[pronome]" si riferisca.” 36: “Non puoi escludere qualcosa che non hai incluso!” 37: “Hey! Puoi farlo solo con esseri viventi.” 38: “Questo è un verbo che non conosco.” 39: “Non è importante ai fini del gioco.” 40: “(il/lo/la/i/gli/le/l’ [oggetto]) ^Non lo/la/li/le vedi.” 41: “Non ho capito la fine del comando.” 42: “Nessuno/Solamente [numero oggetti] ne è disponibile/sono disponibili.^” 43: “Niente da fare!” 44: “Non ce ne sono di disponibili!” 45: “Chi intendi, ” 46: “Cosa intendi, ” 47: “Spiacente, ne puoi avere solo uno. Quale esattamente?” 48: “|Chi vuoi [verbo infinito]?|/|Chi dovrebbe/dovrebbero [verbo infinito] il/lo/la/i/gli/le/l’ [oggetto animato]|?^” 49: “|Cosa vuoi [verbo infinito]?|/|Cosa dovrebbe/dovrebbero [verbo infinito] il/lo/la/i/gli/le/l’ [oggetto animato]|?^” 50: “Il tuo punteggio è appena aumentato/diminuito di un punto/[n] punti.” 51: “(È accaduto qualcosa di drammatico, i comandi disponibili sono stati ridotti.)” 52: “^Inserisci un numero da 1 a [n], 0 per rivedere o premi INVIO.” 53: “^[Per favore premi SPAZIO.]” 54: “[Il tuo commento è stato annotato.]” 55: “[Attenzione: la TRASCRIZIONE non è attiva.]” 56: “.^” 57: “?^”
No	“Era una domanda retorica.”
NotifyOff	“Notifica del punteggio disattivata.”
NotifyOn	“Notifica del punteggio attivata.”
Objects	1: “Oggetti posseduti:^” 2: “Nessuno.” 3: “ (indossato/a/e/i)” 4: “ (in tuo possesso)” 5: “ (dato/a/e/i via)” 6: “ (si trova [nome oggetto o locazione])” 7: “ (nel quale/nella quale/nei quali/nelle quali [nome oggetto o locazione].)” 8: “ (dentro il/lo/la/i/gli/le/l’ [oggetto])” 9: “ (sopra il/lo/la/i/gli/le/l’ [oggetto])” 10: “ (perso/a/e/i)”

Open	1: “Non puoi aprire il/lo/la/i/gli/le/l’ [oggetto].”
	2: “Sembra/Sembrano essere chiuso/a/e/i a chiave.”
	3: “È/Sono già aperto/a/e/i.”
	4: “Hai aperto il/lo/la/i/gli/le/l’ [oggetto], trovando [lista oggetti]/un bel nulla.”
	5: “Ora hai aperto il/lo/la/i/gli/le/l’ [oggetto].”
Order	“Il/lo/la/i/gli/le/l’ [oggetto animato] ha/hanno altre cose da fare.”
Places	“Hai già visitato: ”
Pray	“Sembra che le tue preghiere non siano state esaudite.”
Prompt	1: “^Adesso?”
	2: “^E adesso?”
	3: “^Allora?”
	4: “^Ora?”
	5: “^E ora?”
	6: “^Quindi?”
	7: “^Che si fa?”
	8: “^Che suggerisci di fare?”
	9: “^Che dici di fare?”
	10: “^E ora che si fa?”
	11: “^Cosa devo fare ora?”
	12: “^Sì?”
	13: “^Poi?”
	14: “^Dimmi:”
	15: “^Aspetto un tuo comando:”
Pronouns	1: “Al momento, ”
	2: “significa ”
	3: “è disattivato”
	4: “nessun pronome è conosciuto dal gioco.”
Pull, Push	1: “È/Sono fisso/a/e/i al suo/loro posto.”
	2: “Non sei capace di farlo.”
	3: “Non succede niente di particolare.”
	4: “Questo sarebbe meno che cortese.”
PushDir	1: “Questo è il meglio che ti viene in mente?”
	2: “Non è una direzione.”
	3: “Non puoi in quella direzione.”
PutOn	1: “È necessario che tu possieda il/lo/la/i/gli/le/l’ [primo oggetto] prima di poterlo/a/e/i mettere sopra il/lo/la/i/gli/le/l’ [secondo oggetto].”
	2: “Non puoi mettere qualcosa su se stessa.”
	3: “Mettere cose sopra il/lo/la/i/gli/le/l’ [oggetto] non porterà a niente.”
	4: “Manchi di destrezza.”
	5: “(prima lo/la/li/le lasci)^”
	6: “Non c’è più spazio sopra il/lo/la/i/gli/le/l’ [oggetto].”
	7: “Fatto.”
	8: “Hai messo il/lo/la/i/gli/le/l’ [primo oggetto] sopra il/lo/la/i/gli/le/l’ [secondo oggetto].”
Quit	1: “Per favore, rispondi SI o NO.”
	2: “Sei sicuro di voler uscire? ”
Remove	1: “È/Sono chiuso/a/e/i.”
	2: “Ma non è/Ma non sono lì ora.”
	3: “Rimosso/a/e/i.”
Restart	1: “Sei sicuro di voler ricominciare? ”
	2: “Tentativo fallito.”
Restore	1: “Caricamento fallito.”
	2: “Ok.”

Rub	"Non otterrai niente con questo."
Save	1: "Salvataggio della partita fallito." 2: "Ok."
Score	"In questa partita hai totalizzato/Finora hai totalizzato [punteggio] punto/punti su [punteggio massimo] possibili, in [numero turni] turno/turni"
ScriptOn	1: "La trascrizione è già attivata." 2: "Inizio della trascrizione di " 3: "È fallito il tentativo di iniziare la trascrizione."
ScriptOff	1: "La trascrizione è già disattivata." 2: "^Fine della trascrizione." 3: "È fallito il tentativo di terminare la trascrizione"
Search	1: "Ma è buio." 2: "Non c'è niente sopra il/lo/la/i/gli/le/l' [oggetto]." 3: "Sopra il/lo/la/i/gli/le/l' [oggetto] vedi [lista oggetti]." 4: "Non hai trovato niente di interessante." 5: "Non puoi guardare dentro, perché è/sono chiuso/a/e/i." 6: "Il/lo/la/i/gli/le/l' [oggetto] è/sono vuoto/a/e/i." 7: "Dentro il/lo/la/i/gli/le/l' [oggetto] vedi [lista oggetti]."
Set	"No, non puoi metterlo/la/li/le."
SetTo	"No, non puoi impostarlo/la/li/le a nulla."
Show	1: "Non possiedi il/lo/la/i/gli/le/l' [oggetto]." 2: "Il/lo/la/i/gli/le/l' [oggetto animato] non sembra/non sembrano interessato/a/e/i."
Sing	"Sei stonatissimo."
Sleep	"Non sei per niente assonnato."
Smell	"Non senti alcun odore particolare."
Sorry	"Oh, non scusarti."
Squeeze	1: "Tieni a posto le mani." 2: "Non hai ottenuto niente con questo."
Strong	"I veri avventurieri non usano un simile linguaggio."
Swim	"Non c'è abbastanza acqua in cui poter nuotare."
Swing	"Non c'è niente da poter scuotere qui."
SwitchOn	1: "Non puoi accendere il/lo/la/i/gli/le/l' [oggetto]." 2: "È/Sono già acceso/a/e/i." 3: "Hai acceso il/lo/la/i/gli/le/l' [oggetto]."
SwitchOff	1: "Non puoi spegnere il/lo/la/i/gli/le/l' [oggetto]." 2: "È/Sono già spento/a/e/i." 3: "Hai spento il/lo/la/i/gli/le/l' [oggetto]."
Take	1: "Preso/a/e/i." 2: "Sei sempre il possessore di te stesso." 3: "Non credo che il/lo/la/i/gli/le/l' [oggetto animato] abbia/abbiano intenzione di farsi prendere in braccio." 4: "Dovresti toglierti da sopra/dentro il/lo/la/i/gli/le/l' [oggetto] prima." 5: "Già lo/la/le/li possiedi." 6: "Sembra che appartenga/Sembrano appartenere allo/alla/agli/alle/all' [oggetto]." 7: "Sembra/Sembrano essere parte dello/della/delle/dell' [oggetto]." 8: "Quello/Quella/Quelli/Quelle non è disponibile/non sono disponibili." 9: "Il/lo/la/i/gli/le/l' [oggetto] non è/sono aperto/a/e/i." 10: "Il/lo/la/i/gli/le/l' [oggetto] è/sono difficilmente trasportabile/i." 11: "È/Sono fisso/a/e/i al suo/al loro posto." 12: "Stai trasportando già troppe cose." 13. "(metti il/lo/la/i/gli/le/l' [primo oggetto] dentro il/lo/la/i/gli/le/l' [secondo

	oggetto] per fare spazio)”
Taste	“Nessun sapore particolare.”
Tell	1: “Stai parlando a te stesso.”
	2: “Proprio nessuna reazione.”
Think	“Questa sì che è una buona idea.”
ThrowAt	1: “Senza successo.”
	2: “Nel momento cruciale ti manca il coraggio.”
Tie	vedi **JumpOver**.
Touch	1: “Tieni a posto le mani!”
	2: “Non succede niente di particolare.”
	3: “Se pensi che ciò sia utile.”
Turn	vedi **Pull**.
Unlock	1: “Cerca un'altra soluzione.”
	2: “ Non è/sono chiuso/a/e/i a chiave in questo momento.”
	3: “Non sembra/Non sembrano entrare nella serratura.”
	4: “Ora hai aperto il/lo/la/i/gli/le/l’ [oggetto].”
VagueGo	“Dovresti indicare l'esatta direzione in cui vuoi andare.”
Verify	1: “Il file di gioco è intatto.”
	2: “Il file di gioco non è stato verificato come intatto, e potrebbe essere danneggiato.”
Wait	“Il tempo passa.”
Wake	“Questo non è un sogno. È la spaventosa verità.”
WakeOther	“Non sembra necessario.”
Wave	1: “Ma non lo/la/li/le possiedi.”
	2: “Sembri ridicolo, agitando il/lo/la/i/gli/le/l’ [oggetto].”
WaveHands	“Ti agiti, sentendoti stupido.”
Wear	1. “Non lo/la/le/li puoi indossare!”
	2. “Non lo/la/le/li possiedi!”
	3. “Lo/La/Le/Li stai già indossando!”
	4. “Hai indossato il/lo/la/i/gli/le/l’ [oggetto].”
Yes	vedi **No**.

APPENDICE C – GLI ERRORI DELLA COMPILAZIONE

La compilazione è quel processo che trasforma un file sorgente (nel caso di Inform il file contraddistinto dall'estensione .inf) in un file eseguibile (il file nel formato Z-code, ovvero l'avventura testuale stessa[1]). Nonostante JIF e WIDE mettano a disposizione degli appositi tasti per effettuare quest'importantissima operazione, non sempre tutto va per il verso giusto (possono cioè verificarsi degli errori che impediscono la creazione del file eseguibile). Vediamo allora di elencare alcune regole per impedire che accada quest'evento (o perlomeno per evitare che accada spesso):

1. AD OGNI PARENTESI GRAFFA APERTA ({), NE DEVE SEMPRE CORRISPONDERE UNA CHIUSA (}); LA STESSA COSA VALE PER LE PARENTESI TONDE E QUADRE ([ ]). Questo è l'errore più tipico che si possa fare, e purtroppo è anche uno dei più difficili da trovare, perché il compilatore produce in genere una sfilza di errori che non hanno nulla a che vedere con quello che a noi interessa. Spesso poi, non bisogna fidarsi troppo dei numeri di riga, perché non è detto che l'errore sia sempre nelle righe corrispondenti, ma magari è nelle "vicinanze" (qualche riga più sopra o più sotto) o addirittura in qualche altra parte del programma. JIF, comunque, ci viene in aiuto con un'apposita funzione di controllo delle parentesi;
2. OGNI ISTRUZIONE DEVE TERMINARE (SALVO ALCUNE ECCEZIONI) CON UN PUNTO E VIRGOLA (;). Se questo simbolo viene omesso il compilatore genera, come nel punto 1, una sfilza di errori che non hanno nulla a che vedere con quello che a noi interessa;
3. TUTTO QUELLO CHE VIENE DICHIARATO DEVE SEMPRE ESSERE USATO. Non si può cioè definire, ad esempio, una costante MAX e non usarla all'interno del programma. Il compilatore in questo caso genera sempre un messaggio di errore:

   #C:\Jif\games\Ruins.inf(59): Warning: Defined constant "MAX" declared but not used

   che qui è proprio alla riga 59 ma, lo ripeto per l'ennesima volta, non è sempre così. Lo stesso discorso vale anche per le variabili, le funzioni, gli oggetti, ecc.;
4. TUTTO QUELLO CHE NON VIENE DICHIARATO NON PUÒ ESSERE USATO. Non è possibile, cioè, cercare di utilizzare una variabile, una funzione, un oggetto o altro se non è stato prima dichiarato (o, per dirla in parole più semplici, creato);
5. A PARTE RARE ECCEZIONI (RIGUARDANTI SOPRATTUTTO ALCUNE ESTENSIONI), IL FILE ESEGUIBILE DEVE ESSERE SEMPRE CREATO SENZA LE SEGNALAZIONI DI WARNING. Esistono infatti due tipi di errori: quello più grave (indicato dal compilatore come Error → impedisce la creazione del file eseguibile) e quello più "lieve" (indicato dal compilatore come Warning → non impedisce la creazione del file eseguibile, ma può dare dei problemi durante lo svolgimento del gioco);
6. SE VIENE GENERATA UNA LISTA DI ERRORI, OCCORRE RISOLVERE IL PRIMO E, DOPO, TUTTI QUELLI CHE LO SEGUONO. Nella maggior parte dei casi, infatti, alcuni errori nascono come conseguenza di quelli fatti prima. Risolti questi ultimi, anche gli altri scompaiono;
7. NON SI PUÒ DEFINIRE UN NUOVO VERBO CHE È GIÀ ESISTENTE. Potete però ampliarne il significato tramite le istruzioni Extend e Extend only, come spiegato nel paragrafo 3.9. Un esempio di messaggio per questo tipo di errore è:

   #C:\Jif\games\Ruins.inf(1059): Error: Two different verb definitions refer to "prendi"

   dove si è cercato di definire, mediante l'istruzione Verb, il verbo prendi che esiste già in Infit;

[1] In realtà, i file codificati in questo formato non sono dei veri e propri file eseguibili (che sono invece contraddistinti dall'estensione .exe); questo perché i file nel formato Z-code hanno bisogno, per essere eseguiti, di un programma chiamato interprete (mentre un file eseguibile si avvia da solo, senza nessun programma aggiuntivo). Gli interpreti variano a seconda del sistema operativo usato e su Windows il più famoso è sicuramente Windows Frotz 2002 di David Kinder. Per ulteriori informazioni, consultate il capitolo 1.

8. IN INFORM SI POSSONO USARE, DI DEFAULT, SOLO I VALORI NUMERICI COMPRESI TRA –32768 E 32767. Una variabile, cioè, può contenere solo un valore numerico compreso nel range sopra indicato. In caso contrario, il compilatore non genera un errore, ma i valori memorizzati in essa sono completamente diversi e casuali, con dei risultati totalmente imprevedibili;
9. LE LIBRERIE DI SISTEMA NON SI DEVONO MAI MODIFICARE. Vanno invece usate le apposite istruzioni per la modifica dei messaggi di default, delle azioni e della status line[2]. In questo modo si evitano dei possibili errori e (soprattutto) non si violano i diritti di copyright (le librerie di sistema – quelle di Inform e quelle italiane – sono di proprietà dei rispettivi autori, e nessuno – salvo una loro precisa autorizzazione – può modificarle né tantomeno distribuirle);
10. SEBBENE ESISTANO DIVERSE VERSIONI DELLA Z-MACHINE, È VIVAMENTE CONSIGLIATO USARE LA VERSIONE 5 (.z5) PER LE AVVENTURE TESTUALI A LUNGHEZZA STANDARD, E LA VERSIONE 8 (.z8) PER QUELLE PIÙ GROSSE. A volte (anche se raramente) è possibile trovare qualche gioco nella versione 6 (.z6) usata per supportare la grafica e il suono. Ma in questo caso è molto meglio usare Glulx, una virtual machine a 32 bit il cui compilatore supporta appieno le caratteristiche della (vecchia) Z-machine. In questo modo è possibile, per un programmatore Inform, compilare il suo programma anche per Glulx e accedere così alle istruzioni extra di quest'ultimo per ottenere una maggiore interattività dal punto di vista grafico e sonoro[3];
11. NON POTETE USARE L'ISTRUZIONE INCLUDE SU UN FILE CHE NON ESISTE O CHE NON SI TROVA NELLA PATH LIBRARY SPECIFICATA DI DEFAULT NELL'EDITOR DI TESTO. Nel caso ad esempio di WIDE, supponendo che il mio listato Inform contenga l'istruzione Include "Parser"; e che la path library di default sia "C:\Inform\Libraries", se al momento della compilazione il file Parser.h non si trova nella directory sopra specificata, il compilatore Inform genererà un errore che bloccherà l'operazione in corso;
12. TUTTO QUELLO CHE VIENE DICHIARATO NEL FILE DI INCLUDE, DEVE SEMPRE ESSERE UTILIZZATO. Non posso, cioè, definire un'ipotetica variabile count dichiarata ad esempio nel file Variabili.h e non utilizzarla all'interno del listato. Questo perché quando inizia la fase di compilazione, il compilatore Inform riunisce tutti questi file "esterni" in un unico file che poi utilizza per creare (se non ci sono ovviamente degli errori) l'eseguibile;
13. LA STATUS LINE, SE USATA, VA SEMPRE INIZIALIZZATA. Per sapere come fare, consultate il paragrafo 4.1.

Occorre ora spendere qualche parola su un altro tipico errore "informiano":

#C:\Jif\games\Ruins.inf(154): Warning: This statement can never be reached

generato, per esempio, dal seguente codice:

```
description [;
   "Potresti farci dell'ottimo olio d'oliva.";
   print "Extra-vergine naturalmente.";
],
```

dove l'istruzione print non può essere presa in considerazione dal compilatore. Per far sì che questo non avvenga occorre aggiungere quest'istruzione anche alla riga sopra:

```
description [;
   print "Potresti farci dell'ottimo olio d'oliva.";
   print "Extra-vergine naturalmente.";
```

[2] Con questo termine s'intende la barra in alto che compare nella finestra di Windows Frotz 2002 (o in qualsiasi altro interprete Z-code) quando iniziate a giocare un'avventura testuale. Per ulteriori informazioni, consultate il capitolo 4.
[3] Occorre ricordare che Infit (così come Jif) supporta appieno anche Glulx, e che i giochi in questo formato devono essere eseguiti dal programma WinGlulxe. Ulteriori informazioni potete trovarle nel capitolo 5.

```
],
```

e il problema è risolto. Un altro caso in cui si può verificare quest'errore, è quando "sposto" di posizione una specifica istruzione:

```
[ Initialise;
        TitlePage();
        move map to player;
        move sodium_lamp to player;
        move dictionary to player;
        StartDaemon(sodium_lamp);
        location = Forest;
        thedark.description =
            "L'oscurit@`a intorno a te @`e opprimente e ti senti
             quasi soffocare.";
        "^^^Dopo giorni di inutili ricerche, passati senz'acqua
         attraversando i rovi della foresta, alla fine la tua pazienza
         @`e stata ricompensata: hai fatto una scoperta!^";
        ];
```

Se l'istruzione location non si trova all'inizio della funzione Initialise (o, in questo caso, subito sotto la chiamata alla funzione TitlePage), il compilatore non riuscirà a "trovarla". Attenzione quindi all'ordine che date alle istruzioni delle vostre avventure!!!

Vi ricordo poi, che in fase di esecuzione potrebbero verificarsi degli errori che per il compilatore non esistono. Questi sono i tipici casi in cui il programmatore scambia, ad esempio, il nome di una stanza per un altro, oppure assegna un valore sbagliato (ad esempio 3 al posto di 2) ad una variabile locale. Per il compilatore va tutto bene (non sono cioè presenti errori di sintassi), ma i risultati non sono ovviamente quelli voluti; in questi casi si può ricorrere alla forzatura dell'esecuzione del gioco, come accade nel seguente esempio:

```
[ Initialise;
        TitlePage();

        !#################################################################
        location = Square_Chamber;
        move camera to player;
        move newspaper to player;
        steps.rubble_filled = false;
        remove mushroom;
        ! #################################################################

        move map to player;
        move sodium_lamp to player;
        move dictionary to player;
        StartDaemon(sodium_lamp);
        thedark.description =
            "L'oscurit@`a intorno a te @`e opprimente e ti senti
             quasi soffocare.";
        "^^^Dopo giorni di inutili ricerche, passati senz'acqua
         attraversando i rovi della foresta, alla fine la tua pazienza
         @`e stata ricompensata: hai fatto una scoperta!^";
        ];
```

Se voglio testare direttamente la Sala Quadrata con tutti gli oggetti della cassa da imballaggio nell'inventario del giocatore, stabilisco Square_Chamber come nuova locazione di partenza, sposto

la macchina fotografica e il giornale nell'inventario dell'archeologo, libero il passaggio dalle macerie e rimuovo il fungo. Ecco il risultato:

```
RUINS
Un esempio di lavoro interattivo.
Copyright (c) 1999 di Graham Nelson.
Traduzione e adattamenti di Vincenzo Scarpa e Raffaello Valesio (c) 2002-2003 su permesso dell'autore.

Versione 1 -- Numero di serie 041109
Inform v6.30 -- Libreria 6/11  --  Infit v2.5    S

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>i
Stai portando:
  un dizionario maya di Waldeck
  una lampada al sodio
  la mappa di Quintana Roo
  un giornale di un mese fa
  una macchina fotografica a lastre

>su

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata.
La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>esamina la cassa
La cassa d'imballaggio è vuota.

>giù

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>
```

In questo modo, è possibile testare ogni singola parte del gioco, riuscendo così a capire dove si verificano gli errori più "profondi".
Per i programmatori più esperti, Inform mette a disposizione delle specifiche istruzioni che permettono di controllare il processo di compilazione (le direttive, inizianti tutte con il carattere # e descritte nel paragrafo 2.17). Dei loro esempi di utilizzo potete trovarli nei listati avventura.inf e negozio.inf , oppure, si può sempre dare un'occhiata al listato di "La pietra della luna", la bellissima avventura testuale scritta da Paolo Lucchesi.

Non mi resta che augurare una buona compilazione a tutti…

APPENDICE D – LA SOLUZIONE DI RUINS

Quella che segue, è la soluzione di Ruins sotto forma di TRANSCRIPT OF PLAY (l'esecuzione dell'avventura passo dopo passo): ho volutamente inserito qualche mossa sbagliata per mettere in evidenza alcune caratteristiche di questo linguaggio di programmazione, senza comunque pregiudicare più di tanto l'andamento del gioco[1]. Buon divertimento…

Dopo giorni di inutili ricerche, passati senz'acqua attraversando i rovi della foresta, alla fine la tua pazienza è stata ricompensata: hai fatto una scoperta!

RUINS
Un esempio di lavoro interattivo.
Copyright (c) 1999 di Graham Nelson.
Traduzione e adattamenti di Vincenzo Scarpa e Raffaello Valesio (c) 2002-2003 su permesso dell'autore.

Versione 1 -- Numero di serie 041109
Inform v6.30 -- Libreria 6/11 -- Infit v2.5 S

Il Grande Altopiano
I tuoi appunti menzionano qualcosa circa questa bassa scarpata di calcare, ma la foresta pluviale ne ha rivendicato il possesso. Cupi ulivi crescono ovunque, e una pioggia appena smessa riempie l'aria con una calda nebbia bagnata. La "Struttura 10" è un cumulo di materiali da costruzione, che una volta avrebbero potuto costituire una piramide sepolcrale, ma della quale ora nulla rimane, eccetto alcuni gradini scolpiti nella nuda roccia che portano giù, nell'oscurità.

Un fungo macchiato cresce dalla terra fradicia, su un gambo lungo.

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>E
La foresta pluviale è fitta, e non l'hai attraversata per giorni e giorni per rinunciare alla tua scoperta proprio ora. Hai bisogno di raccogliere un po' di manufatti da riportare alla civiltà, prima di abbandonare la spedizione.

>SU
Gli alberi sono coperti di spine e ti ridurresti le mani a brandelli.

>ESAMINA LA CASSA
Dentro la cassa d'imballaggio vedi una macchina fotografica a lastre e un giornale di un mese fa.

>ESAMINA LA MACCHINA
È un ingombrante, robusto e tenace modello di macchina fotografica a lastre con la struttura di legno: come tutti gli archeologi, le sei particolarmente affezionato.

>PRENDI LA MACCHINA
Presa.

>PRENDI IL GIORNALE
Preso.

>LEGGI IL GIORNALE
Il "Times" del 26 febbraio 1938, che si è subito bagnato e stropicciato dopo essere stato esposto per un mese a questo clima, più o meno come ti senti tu ora. Forse c'è la nebbia a Londra. Forse ci sono le bombe.

>ESAMINA LA STRUTTURA
Le macerie ostruiscono il passaggio dopo appena pochi passi.

>ESAMINA IL FUNGO

[1] Se volete, potete anche scaricare il listato e l'eseguibile del gioco in questione.

Il fungo è ricoperto da delle macchie e non sei del tutto sicuro che non sia velenoso.

>PRENDI IL FUNGO
Hai raccolto abilmente il fungo, senza staccarlo dal suo gambo sottile.

>MANGIA IL FUNGO
Lo sgranocchi ad un angolo, incapace di capire l'origine di un gusto così acre, distratto dal volo di un macao sopra la tua testa che sembra quasi un'esplosione nel sole. Il battito delle sue ali è quasi assordante e delle pietre crollano una sull'altra.

>ESAMINA LA STRUTTURA
Dei gradini rotti e logori conducono verso una sala poco illuminata. Potresti essere la prima persona a mettervi piede dopo cinquecento anni. Sul primo gradino è inscritto il simbolo Q1.

>I
Stai portando:
  un giornale di un mese fa
  una macchina fotografica a lastre
  il dizionario maya di Waldeck
  una lampada al sodio
  la mappa di Quintana Roo

>ESAMINA IL DIZIONARIO
Compilata dall'inaffidabile litografia del leggendario narratore ed esploratore "Conte" Jean Frederic Maximilien Waldeck (1766??-1875), questa guida contiene quel poco che si conosce sui simboli usati nell'antico dialetto locale.

>CERCA Q1 NEL DIZIONARIO
(Questo è uno dei simboli che hai annotato!)
Q1: "luogo sacro".

>ESAMINA LA MAPPA
Questa mappa evidenzia meglio il ruscello che ti ha portato qui, al di là del confine meridionale del Messico, nella più profonda foresta pluviale, interrotta solo da questo altopiano.

>GIÙ

La Sala Quadrata
Sei in una sala di pietra oscura e profonda, larga circa dieci metri. Un raggio di sole, proveniente dalla cima della scalinata, la illumina diffusamente, ma le ombre del livello più basso rivelano dei passaggi verso est e sud, che conducono verso la più profonda oscurità del Tempio.

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>ESAMINA LE ISCRIZIONI
Ogni volta che le osservi attentamente sembrano essere ferme. Ma hai la spiacevole sensazione che, quando distogli lo sguardo, ti si muovano velocemente intorno. Ci sono due simboli predominanti: una freccia e un cerchio.

>CERCA FRECCIA NEL DIZIONARIO
Freccia: "viaggio; divenire".

>CERCA CERCHIO NEL DIZIONARIO
Cerchio: "il Sole; anche la vita, l'arco della vita".

>E

Tana Del Verme
Un groviglio di cunicoli disordinati come una ragnatela si dirige verso le fessure tra le pietre. I soli abbastanza larghi da poterci strisciare dentro sono quelli che si dirigono verso l'alto, a nordest e a sud.

Un bozzolo bianco e scintillante, grande come un pallone da spiaggia, è appiccicato alla fessura di una parete.

>PRENDI IL BOZZOLO
Bleah!

>O

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>POSA IL BOZZOLO NEL RAGGIO
Lasci cadere il bozzolo nel bagliore solare. Ribolle oscenamente, si dilata e poi scoppia. Centinaia di piccoli insetti corrono in tutte le direzioni nell'oscurità; gli spruzzi di melma e una curiosa chiave di pietra gialla sono tutto ciò che rimane sul pavimento.

>PRENDI LA CHIAVE
Presa.

>S

Oscurità
L'oscurità intorno a te è opprimente e ti senti quasi soffocare.

Da qualche parte, dei piccoli artigli si stanno muovendo velocemente.

>N

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>ACCENDI LA LAMPADA
La lampada deve essere ben posizionata prima di essere accesa.

>POSA LA LAMPADA
Posata.

>ACCENDI LA LAMPADA
Hai acceso la lampada al sodio.

>SPINGI LA LAMPADA A SUD

Corridoio in pendenza
Un corridoio basso e squadrato va da nord verso sud, inclinandosi verso la fine.

Il passaggio è bloccato da una massiccia porta di pietra gialla.

C'è una preziosa statuetta maya qui!

>APRI LA PORTA
Sembra essere chiusa a chiave.

>APRI LA PORTA CON LA CHIAVE
Ora la porta non è più chiusa a chiave.

>APRI LA PORTA
Ora hai aperto la porta di pietra.

>ESAMINA LA STATUETTA
È una statuetta minacciosa di uno spirito pigmeo di aspetto quasi grottesco. Ha un serpente intorno al collo.

>PRENDI LA STATUETTA

Questi sono gli anni '30 e non i tempi andati. Prendere un manufatto senza prima registrarlo equivale a un saccheggio.

>POSA TUTTO ECCETTO LA MACCHINA
chiave di pietra: Posata.
giornale di un mese fa: Posato.
dizionario maya di Waldeck: Posato.
mappa di Quintana Roo: Posata.

>FOTOGRAFA LA STATUETTA
Prepari l'elefantiaca macchina fotografica a lastre, sistemi la lampada al sodio e metti pazientemente in posa la statuetta pigmea.

>PRENDI LA STATUETTA, IL GIORNALE E IL DIZIONARIO
statuetta pigmea: Presa.
giornale di un mese fa: Preso.
dizionario maya di Waldeck: Preso.

>N

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>SU

Il Grande Altopiano

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>POSA LA STATUETTA NELLA CASSA
Depositata al sicuro!
[Il tuo punteggio è appena aumentato di cinque punti.]

>GIÙ

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>S

Corridoio in pendenza

La grande porta di pietra gialla è aperta.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Puoi anche vedere la mappa di Quintana Roo e una chiave di pietra qui.

>SPINGI LA LAMPADA A SUD

Il Tempio
Questo magnifico tempio mostra segni di scavi da preesistenti miniere di calcare, specialmente verso il lato occidentale, dove due lunghi cornicioni si dirigono verso sud.

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

Sopra il ripiano dell'altare vedi una maschera facciale in mosaico di giada.

>ESAMINA LE PITTURE
La carne sui corpi è di colore rosso-sangue. Gli indici del conteggio a lungo termine datano l'evento al 10 baktun 4 katun 0 tun 0 uinal 0 kin, quel tipo di anniversario in cui un sovrano decapita un rivale imprigionato che è stato ritualmente torturato lungo un periodo di alcuni anni, secondo la pazzia balcanizzata delle città stato Maya.

>ESAMINA LA MASCHERA
Sarebbe stupendo se si potesse esporre nel museo.

>POSA TUTTO ECCETTO LA MACCHINA
dizionario maya di Waldeck: Posato.
giornale di un mese fa: Posato.

>FOTOGRAFA LA MASCHERA
Prepari l'elefantiaca macchina fotografica a lastre, sistemi la lampada al sodio e metti pazientemente in posa la maschera facciale in mosaico di giada.

>PRENDI LA MASCHERA, IL GIORNALE E IL DIZIONARIO
maschera facciale in mosaico di giada: Presa.
giornale di un mese fa: Preso.
dizionario maya di Waldeck: Preso.

>INDOSSA LA MASCHERA
Guardando attraverso gli occhi di ossidiana della maschera in mosaico, si rivela una presenza spettrale: un sacerdote mummificato aspetta che tu parli.

>ESAMINA IL SACERDOTE
Il suo corpo è disidratato, ed è tenuto insieme solo grazie alla sua forza di volontà. Sebbene la sua lingua sia il Maya locale, hai la curiosa impressione che comprenderà le tue parole.

>CHIEDI AL SACERDOTE DELLE ROVINE
"Le rovine sono sempre state difese dai ladri. Nell'Oltretomba, i saccheggiatori sono stati torturati per l'eternità." Una pausa. "Così come gli archeologi."

>CHIEDI AL SACERDOTE DELLE PITTURE
Il sacerdote si acciglia: "10 baktun, 4 katun, che sono 1,468,800 giorni dall'inizio del tempo: nel tuo calendario il 19 gennaio 909."

>MOSTRA IL DIZIONARIO AL SACERDOTE
Il sacerdote legge un pezzo del libro, sogghignando sinistramente. Incapace di nascondere il suo divertimento, scarabocchia qualche correzione prima di restituirtelo.

>MOSTRA IL GIORNALE AL SACERDOTE
Guarda la data. "12 baktun 16 katun 4 tun 1 uinal 12 kin", dichiara, prima di dare un'occhiata alla prima pagina. "Ah, vedo che continua."

>SO
I cornicioni si vanno restringendo in una crepa che proseguirebbe se non fosse ostruita dai ghiaccioli. Il ghiaccio non riesce a nascondere completamente il simbolo di una mezzaluna.

>CERCA MEZZALUNA NEL DIZIONARIO
Mezzaluna: credo che si pronunci "xibalba", sebbene il suo significato sia sconosciuto.

>CHIEDI AL SACERDOTE DI XIBALBA
Il sacerdote indica con le dita ossute i ghiaccioli, che si sciolgono come neve al sole quando parla. "Xibalbá, l'Oltretomba."

>SPINGI LA LAMPADA A SO

La cosa migliore che puoi fare è spingere la lampada proprio verso il margine del Tempio, dove il pavimento della caverna si ritira.

>SO

Xibalbá
Sei a cinquanta metri sotto la foresta pluviale e lo scroscio dell'acqua si sente ovunque: queste profonde ed erose miniere di calcare si estendono come radici. Verso nordest un percorso scivoloso, formato da un'ampia colonna di roccia ricoperta dal ghiaccio, ti riporta al Tempio, mentre un canyon prosegue verso l'alto a nord e verso il basso a sud. Alla luce diffusa della lampada al sodio proveniente da sopra, sembra essere bianco come i denti di uno squalo.

Una piccola stele, o forse una pietra di confine, giace su una sporgenza ad altezza d'uomo.

>S

Estremità inferiore del canyon
All'estremità sud, più bassa e stretta, il canyon si arresta ad un baratro oscuro e vertiginoso. Niente può essere visto o sentito da lì sotto.

>N

Xibalbá

Una piccola stele, o forse una pietra di confine, giace su una sporgenza ad altezza d'uomo.

>N

Estremità superiore del canyon
La più alta e vasta estremità a nord del canyon sale soltanto verso una parete irregolare di roccia calcarea vulcanica.

C'è un'enorme sfera di pietra pomice larga circa due metri e mezzo qui.

>ESAMINA LA SFERA
È larga circa due metri e mezzo, sebbene sembri piuttosto leggera.

>SPINGI LA SFERA A SUD
La sfera è difficile da fermare una volta che è stata mossa.

Xibalbá

Una piccola stele, o forse una pietra di confine, giace su una sporgenza ad altezza d'uomo.

>G

Xibalbá
Sei a cinquanta metri sotto la foresta pluviale e lo scroscio dell'acqua si sente ovunque: queste profonde ed erose miniere di calcare si estendono come radici. Verso nordest un percorso scivoloso, formato da un'ampia colonna di roccia ricoperta dal ghiaccio, ti riporta al Tempio, mentre un canyon prosegue verso l'alto a nord e verso il basso a sud. Alla luce diffusa della lampada al sodio proveniente da sopra, sembra essere bianco come i denti di uno squalo.

C'è un'enorme sfera di pietra pomice larga circa due metri e mezzo qui.

Una piccola stele, o forse una pietra di confine, giace su una sporgenza ad altezza d'uomo.

>SPINGI LA SFERA A SUD
La sfera è difficile da fermare una volta che è stata mossa.

Estremità inferiore del canyon

La sfera di pietra pomice rotola giù nel canyon senza controllo per alcuni metri, prima di sussultare nelle fauci del baratro. Con un piccolo rimbalzo, ti colpisce di striscio sulla fronte: cadi in avanti sanguinante e... la sfera di pietra pomice diventa più piccola oppure è la tua mano che cresce, perché ora ti sembra di tenerla, mentre fissi l'Alligatore, figlio di sette Macao, le teste dei suoi ultimi avversari infilzati sulle punte, una congregazione che scalpita per il tuo sangue, e comunque non c'è nulla che tu possa fare e... ma tutto questo non ha senso e hai un mal di testa martellante.

>G

Estremità inferiore del canyon
L'estremità sud del canyon continua ora sopra la sfera di pietra pomice incastrata nel baratro.

>S

Sporgenza di pietra pomice
Una sporgenza improvvisa formata dalla sfera di pietra pomice è conficcata in un punto del baratro. Il canyon tuttavia termina qui.

Di tutte le offerte sacrificali gettate nel baratro, nulla forse sarà recuperato: fatta eccezione per un osso inciso, più luminoso di quello che sembra, che sporge da una polla di limo bagnato nella parete del canyon.

>ESAMINA L'OSSO
Una mano che tiene una penna di piuma compare dalle mascelle di Itzamná, l'inventore della scrittura, nella sua forma di serpente.

>TOGLI LA MASCHERA
Ti sei tolto la maschera facciale in mosaico di giada.

>POSA TUTTO ECCETTO LA MACCHINA
dizionario maya di Waldeck: Posato.
giornale di un mese fa: Posato.
maschera facciale in mosaico di giada: Posata.

>FOTOGRAFA L'OSSO
Prepari l'elefantiaca macchina fotografica a lastre, sistemi la lampada al sodio e metti pazientemente in posa l'osso inciso.

>PRENDI TUTTO
maschera facciale in mosaico di giada: Presa.
giornale di un mese fa: Preso.
dizionario maya di Waldeck: Preso.
osso inciso: Preso.

>N

Estremità inferiore del canyon

>N

Xibalbá

Una piccola stele, o forse una pietra di confine, giace su una sporgenza ad altezza d'uomo.

>ESAMINA LA STELE
Le iscrizioni sembrano avvertirti che il confine di Xibalbá, il Luogo Della Paura, è vicino. Il simbolo di un uccello è predominante.

>CERCA UCCELLO NEL DIZIONARIO
Uccello: "morto stecchito".

>POSA TUTTO ECCETTO LA MACCHINA
osso inciso: Posato.
dizionario maya di Waldeck: Posato.

giornale di un mese fa: Posato.
maschera facciale in mosaico di giada: Posata.

>FOTOGRAFA LA STELE
Prepari l'elefantiaca macchina fotografica a lastre, sistemi la lampada al sodio e metti pazientemente in posa la stele.

>PRENDI TUTTO
maschera facciale in mosaico di giada: Presa.
giornale di un mese fa: Preso.
dizionario maya di Waldeck: Preso.
osso inciso: Preso.
stele: Presa.

>NE

Il Tempio

La grande porta di pietra gialla è aperta.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

>N

Oscurità

Da qualche parte, dei piccoli artigli si stanno muovendo velocemente.

>N

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>SU

Il Grande Altopiano

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>POSA LA STELE NELLA CASSA
Depositata al sicuro!
[Il tuo punteggio è appena aumentato di cinque punti.]

>POSA L'OSSO NELLA CASSA
Depositato al sicuro!
[Il tuo punteggio è appena aumentato di cinque punti.]

>POSA LA MASCHERA NELLA CASSA
Depositata al sicuro!
[Il tuo punteggio è appena aumentato di dieci punti.]

>GIÙ

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>S

Oscurità

Da qualche parte, dei piccoli artigli si stanno muovendo velocemente.

>S

Il Tempio

La grande porta di pietra gialla è aperta.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

>SPINGI LA LAMPADA A SE

Anticamera
I cornicioni formano, a sudest del Tempio, una curiosa anticamera.

Una gabbia di ferro, di aspetto alquanto sinistro, ha la porta aperta. È abbastanza larga da poterci entrare dall'alto e ci sono alcuni simboli sulla struttura.

>ENTRA NELLA GABBIA
Gli scheletri che popolano la gabbia ritornano in vita, bloccandoti con le loro mani ossute, schiacciandoti e prendendoti a pugni. Perdi conoscenza e quando ti risvegli, ti rendi conto che è accaduto qualcosa di impossibile e grottesco...

Anticamera (come facocero)
I cornicioni formano, a sudest del Tempio, una curiosa anticamera.

La lampada al sodio, posata a terra, si sta lentamente consumando.

La gabbia di ferro è chiusa.

>APRI LA GABBIA
Un facocero non può fare una cosa simile. Se non fosse per il peso e per la capacità di vedere di notte, non sarebbe poi tanto peggio di molte persone.

>NO

Il Tempio (come facocero)

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

>N

Corridoio in pendenza (come facocero)

La grande porta di pietra gialla è aperta.

Puoi anche vedere la mappa di Quintana Roo e una chiave di pietra qui.

>N

La Sala Quadrata (come facocero)

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>E

Tana Del Verme (come facocero)

>GIÙ
La tana diventa scivolosa intorno al tuo corpo di facocero e strilli senza volerlo come annaspi nell'oscurità, precipitando infine a sud verso...

La Cripta (come facocero)
Sui tuoi appunti di lavoro puoi leggere: "Una cripta arredata di mensole, otturata con un sigillo di terracotta e contenente alcune figure dipinte che potrebbero rappresentare i Nove Signori Della Notte. Delle ossa sparpagliate sembrerebbero appartenere a un uomo anziano e a diversi bambini sacrificati, mentre altri resti funerari comprendono anche le zampe di un giaguaro." (Negli appunti è importante non far notare che sei spaventato.)

C'è un favo splendidamente conservato qui.

>SU
Con un balzo poderoso, cozzi sopra il sigillo di terracotta posto al di sopra della sala, facendoti crollare addosso cenere e terra. Qualcosa privo di vita e terribilmente pesante cade su di te. Perdi conoscenza e quando ti risvegli ti rendi conto che è accaduto qualcosa di impossibile e grottesco...

La Cripta
Sui tuoi appunti di lavoro puoi leggere: "Una cripta arredata di mensole, otturata con un sigillo di terracotta e contenente alcune figure dipinte che potrebbero rappresentare i Nove Signori Della Notte. Delle ossa sparpagliate sembrerebbero appartenere a un uomo anziano e a diversi bambini sacrificati, mentre altri resti funerari comprendono anche le zampe di un giaguaro." (Negli appunti è importante non far notare che sei spaventato.)

Un facocero sbuffa e grugnisce nelle ceneri.

C'è un favo splendidamente conservato qui.

>ESAMINA IL FAVO
Forse qualche tipo di offerta funeraria.

>POSA TUTTO ECCETTO LA MACCHINA
dizionario maya di Waldeck: Posato.
giornale di un mese fa: Posato.

>FOTOGRAFA IL FAVO
Prepari l'elefantiaca macchina fotografica a lastre, sistemi la lampada al sodio e metti pazientemente in posa il favo.

>PRENDI IL FAVO E IL DIZIONARIO
favo: Preso.
dizionario maya di Waldeck: Preso.

>SU

Anticamera (dentro la gabbia di ferro)
I cornicioni formano, a sudest del Tempio, una curiosa anticamera.

La lampada al sodio, posata a terra, si sta lentamente consumando.

Dal pavimento della gabbia, un pozzo di terracotta scende nella cripta.

>ESCI DALLA GABBIA
Non puoi uscire fuori, la gabbia di ferro è chiusa.

>APRI LA GABBIA
Ora hai aperto la gabbia di ferro.

>ESCI DALLA GABBIA
Sei uscito dalla gabbia di ferro.

Anticamera

La lampada al sodio, posata a terra, si sta lentamente consumando.

Una gabbia di ferro, di aspetto alquanto sinistro, ha la porta aperta. È abbastanza larga da poterci entrare dall'alto e ci sono alcuni simboli sulla struttura.

>SPINGI LA LAMPADA A NO

Il Tempio

La grande porta di pietra gialla è aperta.

Ci sono delle pitture impegnative e vivide qui, troppo lucide per essere guardate e ottenute mediante incisione da parte di un popolo altamente organizzato. Mostrano un sovrano corazzato che calpesta un prigioniero.

Una grande lastra di pietra che funge da tavolo o altare, domina il Tempio.

>SPINGI LA LAMPADA A N

Corridoio in pendenza

La grande porta di pietra gialla è aperta.

Puoi anche vedere la mappa di Quintana Roo e una chiave di pietra qui.

>SPINGI LA LAMPADA A N

La Sala Quadrata

Delle iscrizioni scolpite riempiono le pareti, il pavimento e il soffitto.

>SPEGNI LA LAMPADA
Hai spento la lampada al sodio.

>PRENDI LA LAMPADA
Presa.

>SU

Il Grande Altopiano

La tua cassa d'imballaggio giace qui, pronta per contenere eventuali scoperte di alto valore culturale da riportare alla civiltà.

>POSA IL FAVO NELLA CASSA
Come depositi con attenzione il favo un macao dalla coda rossa fluttua giù dalle cime degli alberi, con le piume appesantite dalla pioggia recente. Il battito delle sue ali è quasi assordante e delle pietre crollano una sull'altra.

Non appena il cielo si schiarisce, una luna crescente sorge sopra una tranquilla giungla. È la fine di marzo del 1938, ed è l'ora di tornare a casa.

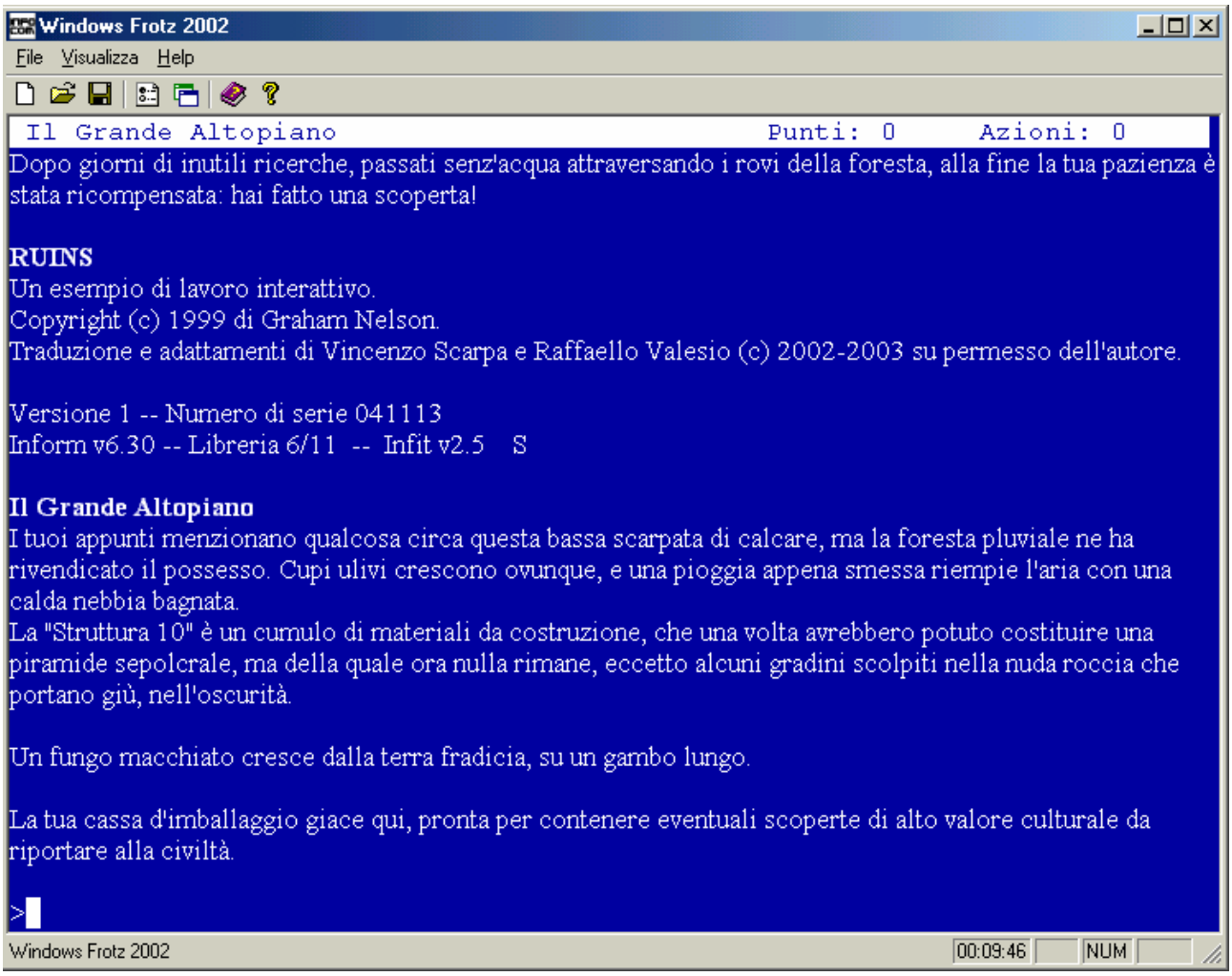
*** Hai vinto ***

In questa partita hai totalizzato 30 punti su 30 possibili, in 117 turni, guadagnando il rango di Direttore della Fondazione Carneige.

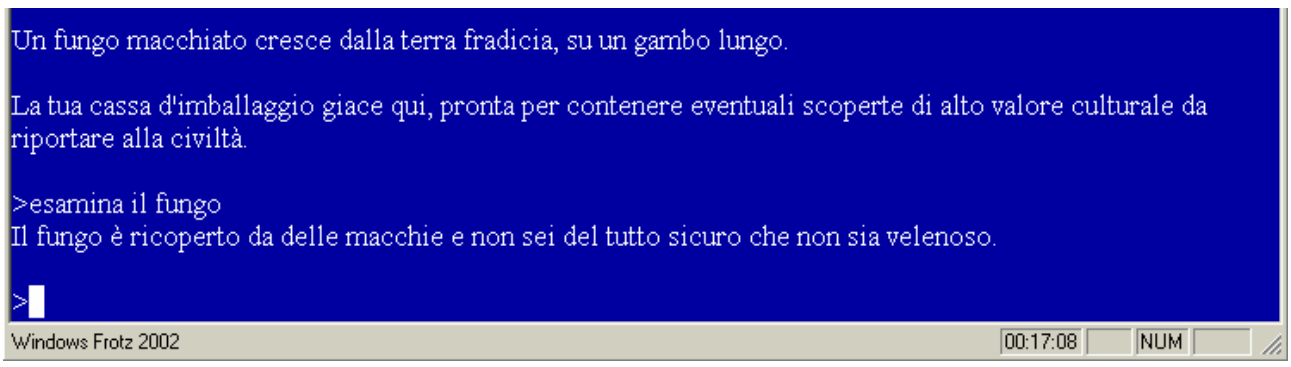
Vuoi RICOMINCIARE, CARICARE una partita salvata o USCIRE ?
> USCIRE

APPENDICE E – COME USARE WINDOWS FROTZ 2002

Come abbiamo ripetuto più volte all'interno di questo manuale, Windows Frotz 2002 è un interprete che esegue tutte le avventure testuali compilate nel formato Z-code (.Z5 o .Z8)[1]. Vediamo allora come usarlo:



Questa è la tipica schermata iniziale che contraddistingue l'inizio di ogni avventura che viene giocata. I comandi vengono digitati per mezzo della tastiera, seguiti dalla pressione del tasto di INVIO:



[1] Oltre alle avventure testuali nel formato Z-code, ne esistono anche delle altre scritte in formati diversi. I principali di questi sono: ADRIFT (contraddistinto dall'estensione .taf), AGT, ALAN, GLULX (contraddistinto dalle estensioni .ulx e .blb), HUGO, MS-DOS e TADS (contraddistinto dall'estensione .gam). Ulteriori informazioni potete reperirle alle rispettive home page di appartenenza.

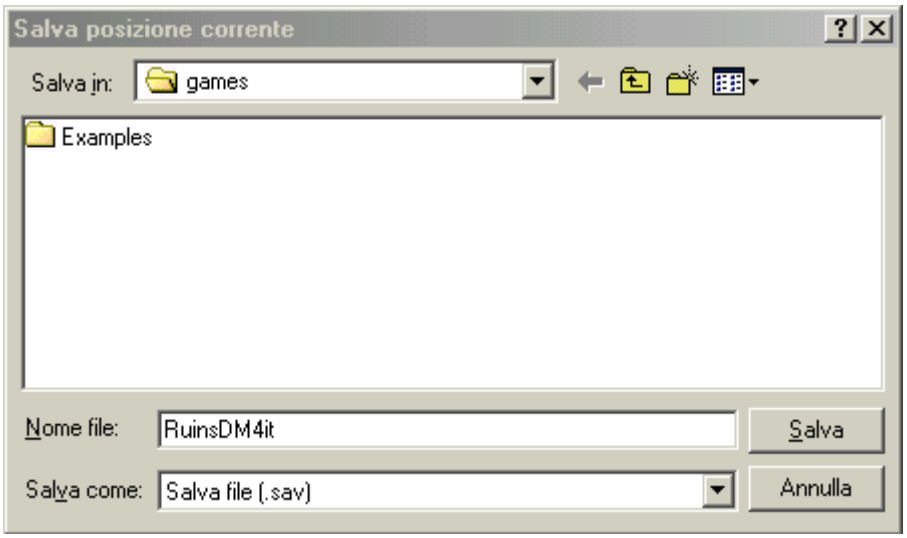
Alcuni di questi (descritti nel paragrafo 3.10) permettono di agire sul sistema, mentre tutti gli altri sono descritti nell'appendice A (o, eventualmente, nell'estensione command_it.h).

Dalla BARRA DEGLI STRUMENTI (contraddistinta da una serie di sette icone) è possibile effettuare le seguenti operazioni:

- per iniziare una nuova avventura (equivalente alla pressione contemporanea dei tasti CTRL + N);
- per caricare una posizione salvata (equivalente al comando CARICA o alla pressione contemporanea dei tasti CTRL + O);
- per salvare la posizione corrente (equivalente al comanda SALVA o alla pressione contemporanea dei tasti CTRL + S);
- per le opzioni (equivalente alla pressione contemporanea dei tasti CTRL + P);
- per il buffer di scorrimento (equivalente alla pressione contemporanea dei tasti CTRL + L);
- per la guida in linea;
- per le informazioni sul programma.

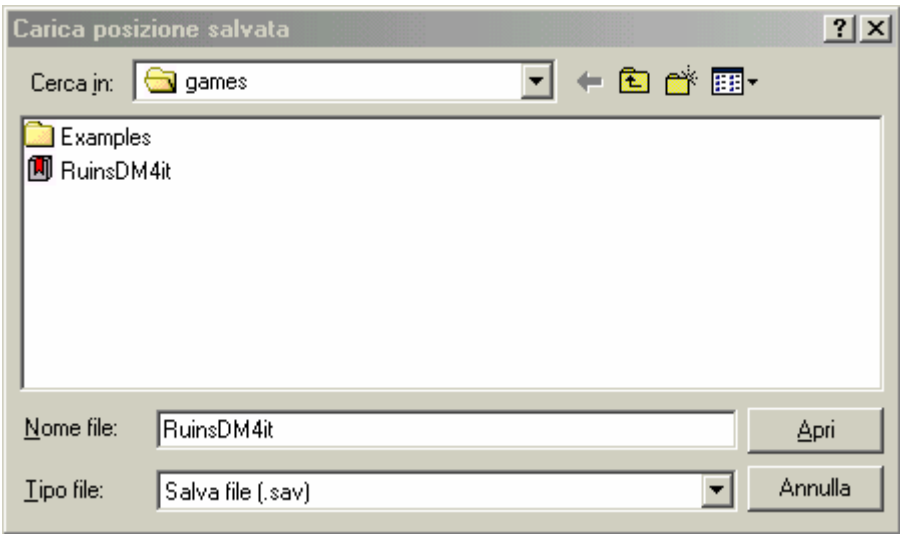
Il BUFFER DI SCORRIMENTO è una modalità di funzionamento nella quale il programma dà, al giocatore, la possibilità di copiare tutto o parte del testo visualizzato in quel momento (senza però scriverlo in un file come avviene nel caso del comando SCRIPT ON).

Per quanto riguarda il SALVATAGGIO DELLA POSIZIONE CORRENTE, alla pressione dell'apposito pulsante con il tasto sinistro del mouse appare la seguente finestra:



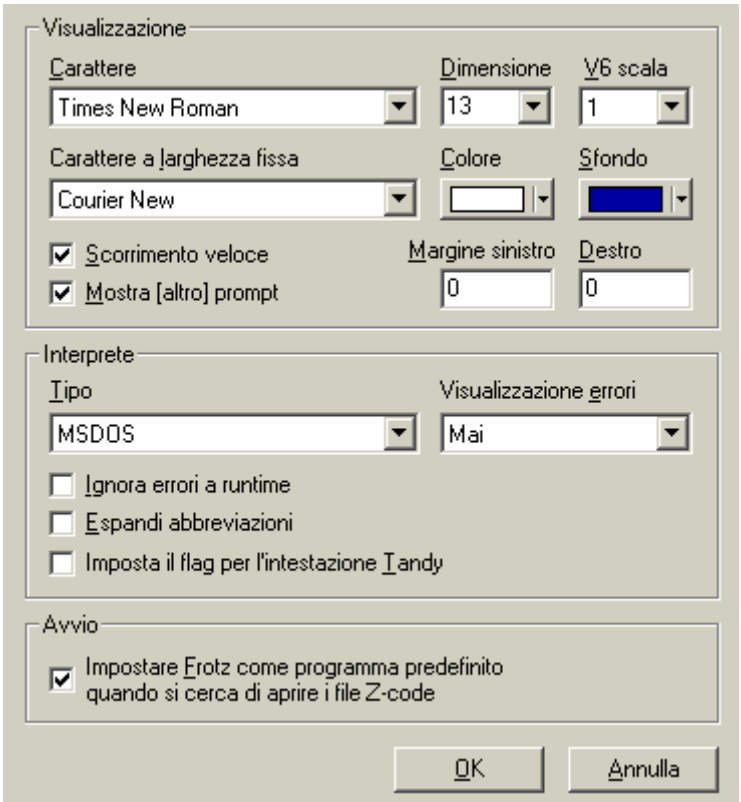
all'interno della quale occorre specificare il nome del file (che avrà l'estensione .sav) e cliccare poi sul pulsante SALVA.

Per quanto riguarda invece il CARICAMENTO DELLA POSIZIONE PRECEDENTEMENTE SALVATA, alla pressione dell'apposito pulsante con il tasto sinistro del mouse, appare invece la seguente finestra:



all'interno della quale occorre selezionare il file interessato e cliccare poi sul pulsante APRI. Una volta caricato il file, è sempre poi utile dare il comando GUARDA (o G) per stampare a video la descrizione della locazione corrente.

Le OPZIONI del programma sono racchiuse nella seguente finestra:



Nella scheda VISUALIZZAZIONE è possibile stabilire il tipo di carattere col quale visualizzare il testo dell'avventura giocata (Times New Roman di default), la dimensione di quest'ultimo e la scala V6 (usata per scalare i file grafici di alcuni giochi della Infocom scritti nel formato V6 della Z-Machine); segue poi il tipo di carattere a larghezza fissa (Courier New di default) insieme rispettivamente al colore del testo (bianco di default) e dello sfondo (blu di default). Segue infine la possibilità di attivare o meno lo scorrimento veloce del testo (attivato di default), di mostrare o meno il prompt [Altro] durante lo scorrimento di un testo molto lungo (attivato di default) e di definire i valori rispettivamente del margine sinistro e destro dello schermo.

Nella scheda INTERPRETE è possibile invece stabilire il tipo (MS-DOS di default) e se visualizzare o meno gli errori ('Mai' di default); segue poi la possibilità di ignorare o meno gli errori a runtime (se questa opzione è attiva, Windows Frotz 2002 continua ad essere eseguito anche se si è in presenza di questo tipo di errore – utile soltanto nel caso in cui si stia provando un gioco di cui non sono disponibili né il listato né il nome dell'autore), di espandere o meno le abbreviazioni (se questa opzione è attiva, Windows Frotz 2002 espande i comandi abbreviati come completi - ad esempio x → esamina - anche se il gioco non li supporta; utile per giocare ad alcune vecchie avventure della Infocom) e di impostare o meno il flag per l'intestazione Tandy (disattivato di default).
Nella scheda AVVIO è possibile impostare o meno il programma come predefinito quando si cerca di aprire un file nel formato Z-code.

Occorre inoltre ricordare che:

- con la PRESSIONE CONTEMPORANEA DEI TASTI ALT+D si accede alle opzioni di debug;
- con la PRESSIONE CONTEMPORANEA DEI TASTI ALT+N si incomincia una nuova partita (equivalente al comando RICOMINCIA/RICOMINCIARE);
- con la PRESSIONE CONTEMPORANEA DEI TASTI ALT+H si accede agli aiuti di Windows Frotz 2002;
- con la PRESSIONE CONTEMPORANEA DEI TASTI ALT+X si termina il gioco.

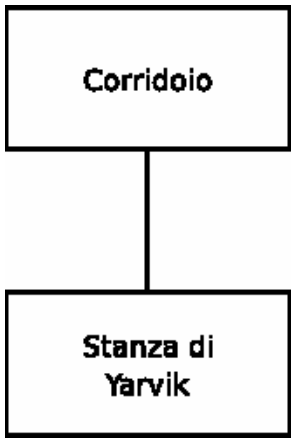
Occorre infine notare che questo programma è in grado di eseguire i file nel formato .blb, ma per questi ultimi è molto meglio utilizzare allo scopo WinGlulxe[2], l'interprete che esegue le avventure testuali scritte in Glulx[3].

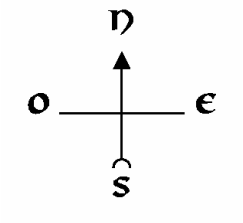
[2] In realtà è possibile utilizzare anche Gargoyle, uno straordinario interprete che è in grado di eseguire avventure testuali in diversi formati (Z-code compreso). Ulteriori informazioni potete trovarle al paragrafo 1.4.
[3] Glulx (pienamente supportato da Infit e da Jif) è un linguaggio di programmazione (perfettamente compatibile con Inform) che offre la possibilità di creare delle avventure testuali grafiche e sonore. Ulteriori informazioni potete trovarle al capitolo 5.

APPENDICE F – LE MAPPE

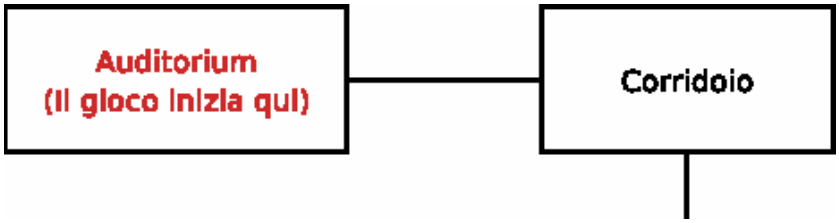
Spesso, quando ci s'imbatte in un'avventura piuttosto lunga e difficile, è molto utile (se non addirittura indispensabile) creare una mappa. Ogni locazione esplorata infatti, può essere rappresentata graficamente da un rettangolo contenente al suo interno il nome di quest'ultima:

Se, come di solito accade, la locazione in questione ne ha d'adiacenti, occorre allora collegarle tra loro:

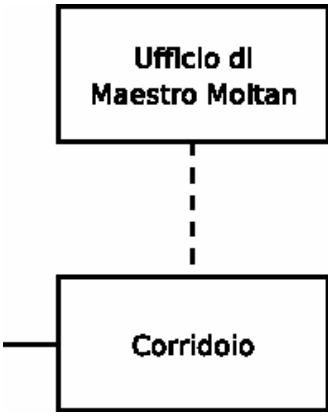




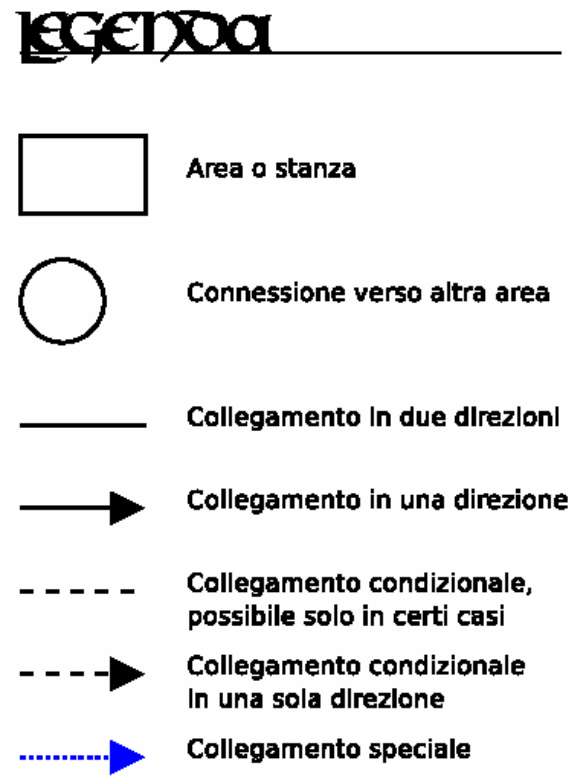
Così facendo, si capisce molto bene che a sud del Corridoio si trova la Stanza di Yarvik. La rosa dei venti, posizionata in alto a destra e indicante le quattro principali direzioni cardinali (oltre a nord, sud, est e ovest ci sono anche nordovest, nordest, sudovest e sudest), deve essere quindi sempre presente. Naturalmente, fra tutte queste locazioni ce ne deve pur essere una di partenza:



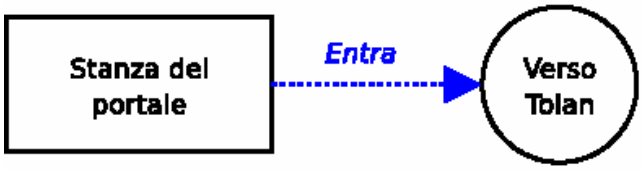
che in questo caso è l'Auditorium, che si trova proprio ad ovest del corridoio (o meglio, è quest'ultimo che si trova a est dell'Auditorium – la locazione iniziale ha sempre la priorità maggiore su tutte le altre).
Stando così le cose, tutto sembra essere semplice, vero? Beh… proviamo allora a dirigerci a nord del corridoio:



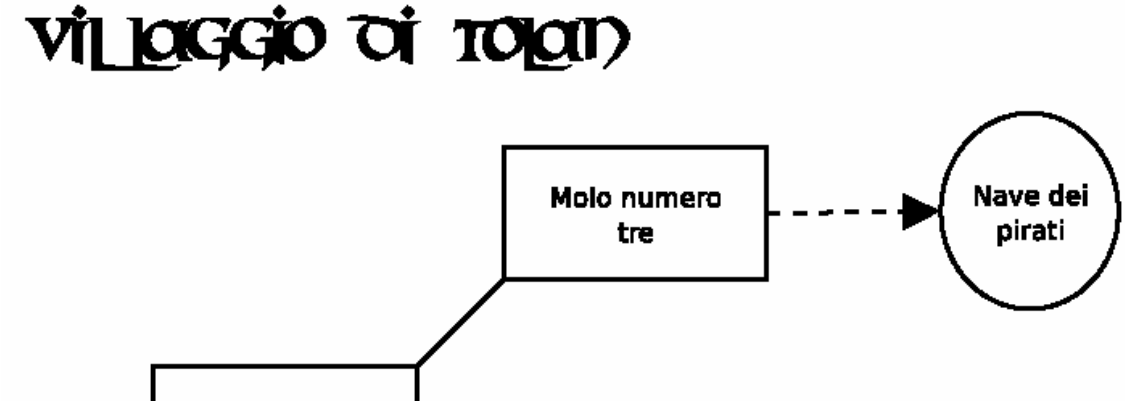
Notiamo subito che in questa direzione c'è l'Ufficio del Maestro Moltan; perché però la linea di collegamento è tratteggiata? Se diamo un'occhiata alla legenda:



capiamo subito che si tratta di un collegamento condizionale, possibile solo in certi casi. Per quanto riguarda invece il collegamento speciale:



si può capire abbastanza facilmente dalla figura a cosa si riferisce. È un collegamento a una locazione particolare, che in questo caso prevede addirittura un vero e proprio "teletrasporto" verso Tolan, un'altra area del gioco[1].
L'ultima cosa che rimane da vedere è il collegamento in una direzione:



che, tradotto in parole povere, indica la possibilità di andare verso una certa direzione senza però poter tornare più indietro (una via a senso unico insomma). Nel caso della figura, il giocatore può dirigersi (a patto che si verifichino una o più condizioni) dal Molo numero Tre verso la Nave dei

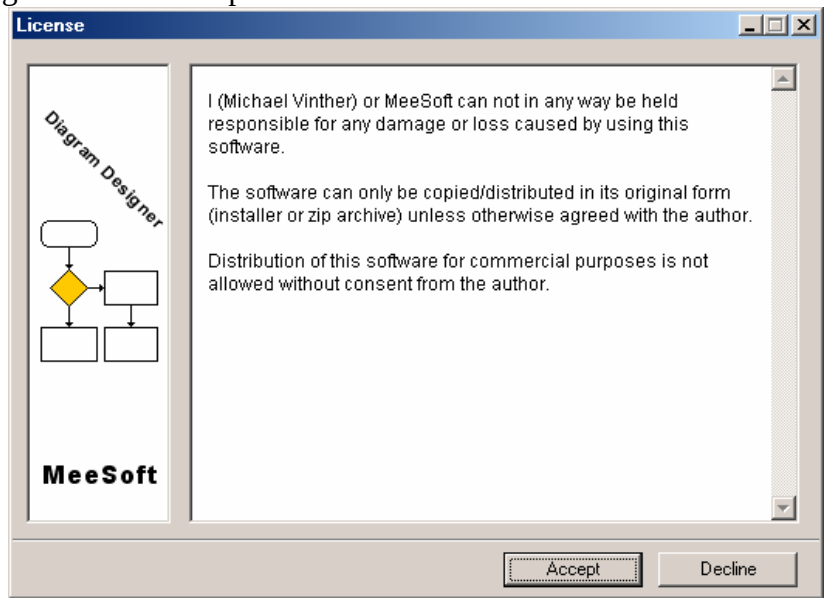
[1] Per chi non l'avesse ancora capito, il gioco in questione è WarMage dello stratosferico Giancarlo Niccolai.

pirati ma non viceversa. La linea di collegamento obliqua, inoltre, indica che la locazione cui fa riferimento è collegata a un'altra verso sudovest (il Porticciolo, anche se non si vede in figura)[2].

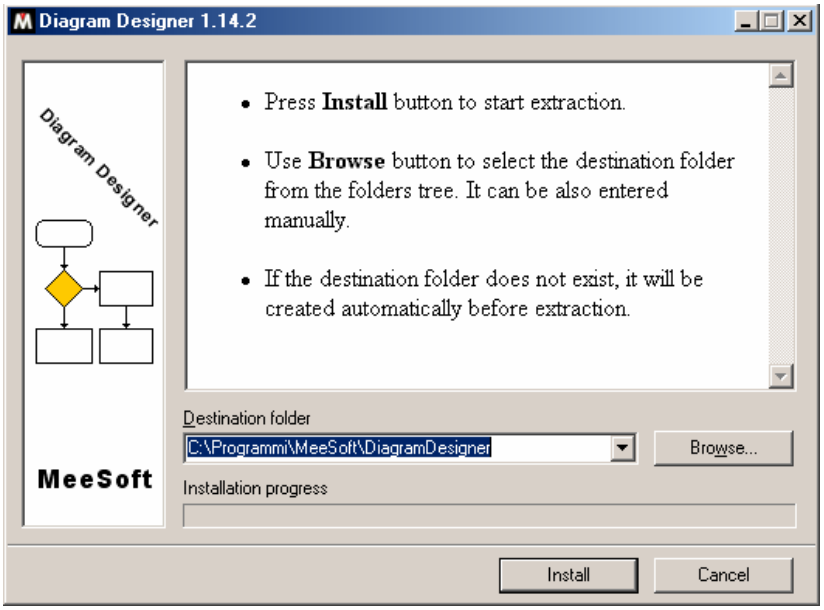
Bene. Ora che abbiamo visto da vicino una mappa ben fatta, passiamo ad occuparci del lato "pratico", ovvero: come possiamo crearne una da soli?

Negli anni '80, quando cioè le limitate capacità grafiche dei computer dell'epoca non permettevano d'avere la grafica di oggi, non era insolito trovare qualcuno che se le disegnava a mano. Oggi, invece, è possibile rifarsi a numerosi programmi adatti allo scopo. Tra questi, penso che occorra citare almeno il Dia di Hans Breuer e Steffen Macke e l'IFMapper di Gonzalo Garramuno. Io però, ho preferito rifarmi al Diagram Designer di Michael Vinther per due ragioni: la ridotta dimensione del file (appena 800 k) e l'estrema facilità d'uso che lo caratterizza[3].

Vediamo allora come installarlo. Per prima cosa, scaricate il file DiagramDesigner.zip e decomprimetelo in una directory qualsiasi. Eseguite poi il file DiagramDesignerSetup.exe per ottenere così la seguente videata di presentazione:



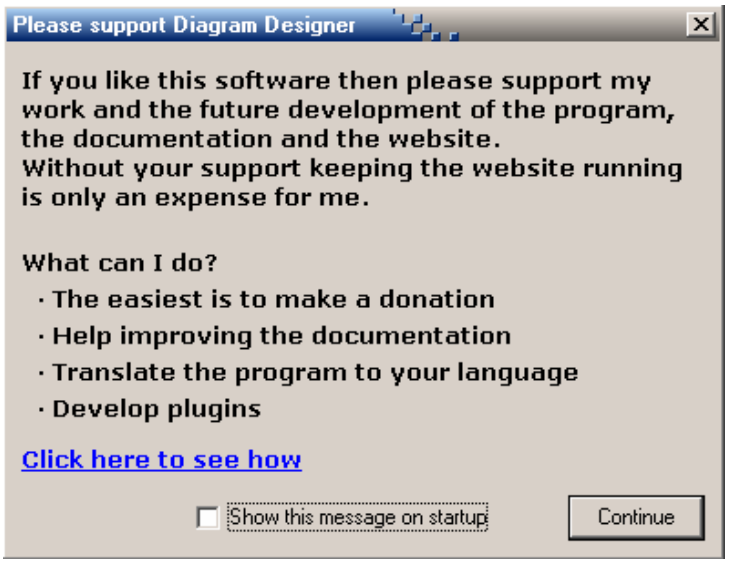
Posizionatevi quindi con il puntatore del mouse sul tasto Accept e cliccate il tasto sinistro di quest'ultimo. La videata successiva:



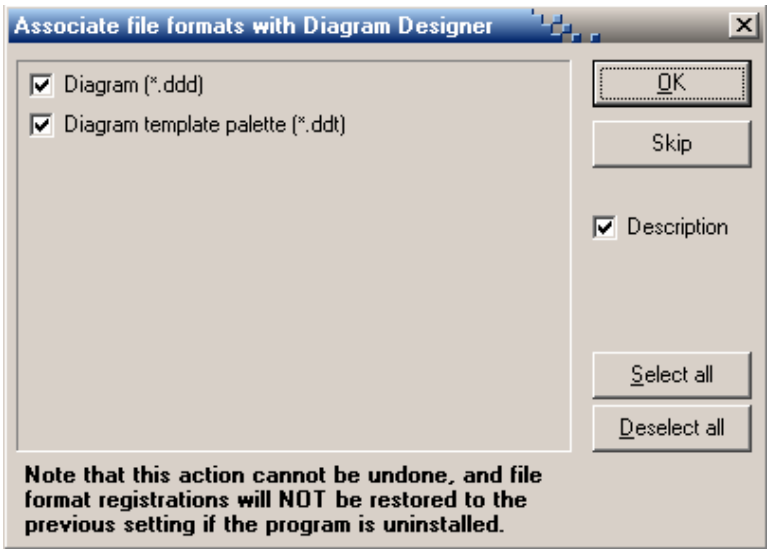
[2] La mappa completa dell'avventura potete scaricarla dal sito di WarMage. Ne potete comunque reperire molte altre all'indirizzo http://solutionarchive.com/.

[3] Tutti e tre i programmi sono completamente freeware (o gratuiti). Tra questi, comunque, il Dia è sicuramente il più completo (ed è quello tra l'altro che ha usato Giancarlo Niccolai per creare la sua splendida mappa).

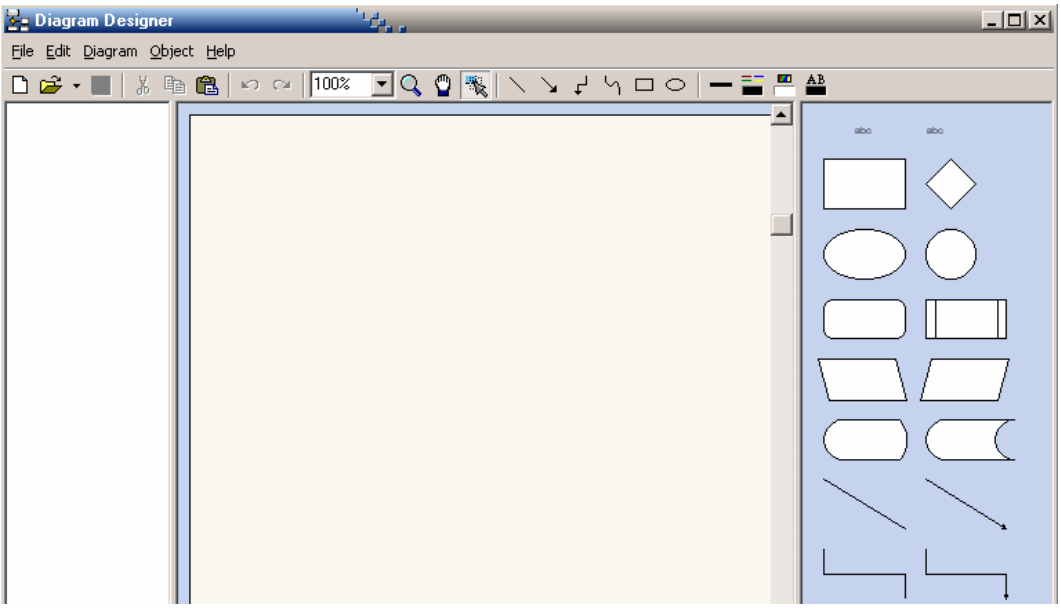
ci chiede semplicemente dove vogliamo installare il programma. Lasciamo tutto così com'è e proseguiamo cliccando sul tasto Install:



Assicuriamoci che la casella di spunta non sia selezionata e clicchiamo sul tasto Continue:

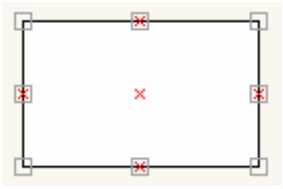


In questa videata vengono associati, se selezionati, i formati dei file proprietari del programma. Lasciamo tutto così com'è e proseguiamo cliccando, con il pulsante sinistro del mouse, il tasto Ok:

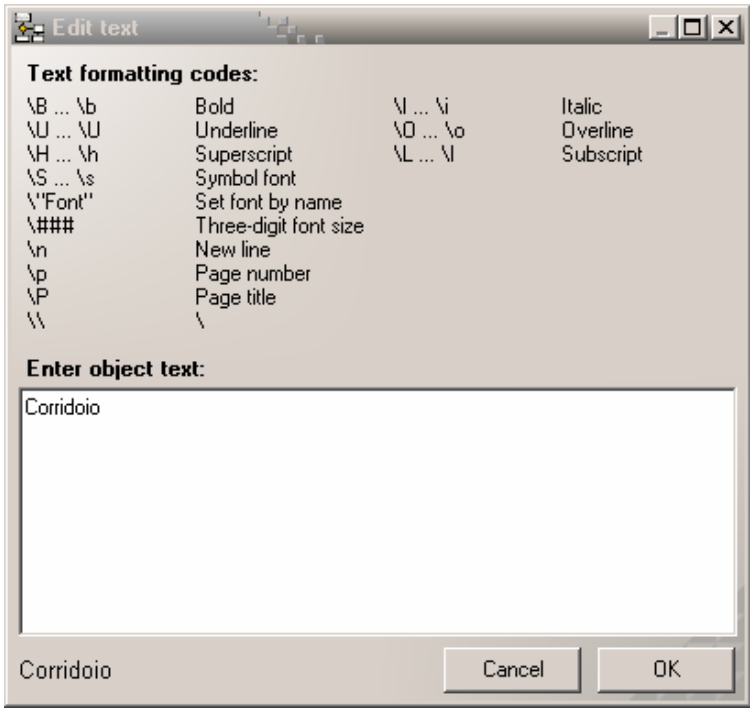


Ed ecco finalmente il nostro programma. Carino, vero?

La terza e ultima parte di quest'appendice riguarda appunto la creazione della mappa vera e propria. Come potete vedere, Diagram Designer permette di creare diverse figure geometriche anche se a noi interessano principalmente il rettangolo e il cerchio[4].
Iniziamo allora a creare un rettangolo. Posizioniamoci con il puntatore del mouse sulla forma del rettangolo e, tenendo premuto il tasto sinistro del mouse, trasciniamolo nell'area grigia a sinistra. Il risultato ottenuto è il seguente:



ovvero un rettangolo che può, tramite le apposite maniglie, essere ridotto o ampliato di dimensioni. Dando per scontato che così com'è vada bene, vediamo allora come si può inserire al suo interno del testo:

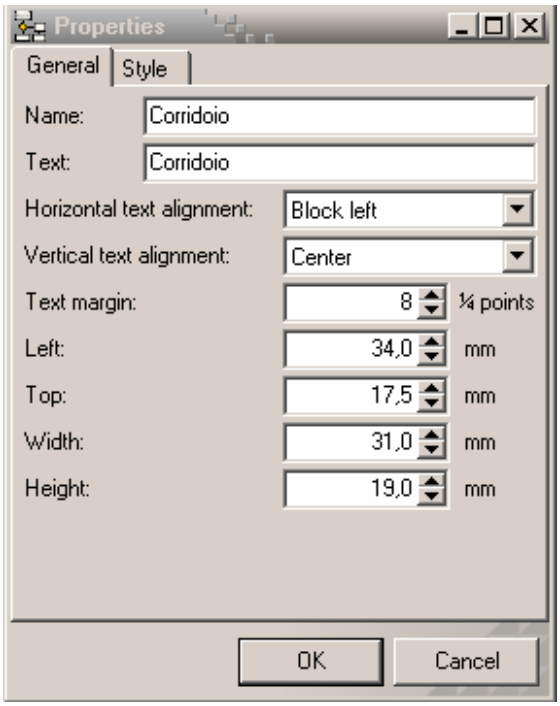


Premendo il tasto F2 appare la finestra di editazione del testo. Come potete facilmente notare, se vogliamo utilizzare uno stile diverso da quello standard, dobbiamo racchiudere il testo da noi voluto dai simboli \B e \b per il grassetto, \I e \i per il corsivo, \U e \u per il sottolineato e così via. Clicchiamo comunque sul tasto OK nell'angolo in basso a destra:

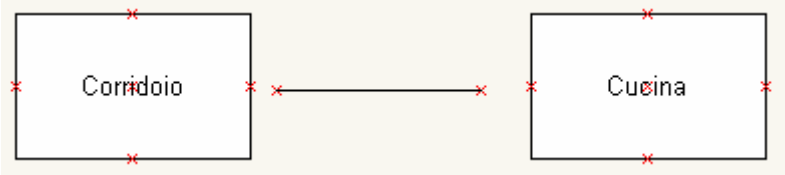
Ecco dunque rappresentata la nostra prima locazione. Se non ci piace il font di default (ovvero l'Arial con corpo 10) possiamo tranquillamente cambiarlo selezionando la funzione Default Font dal menu Diagram. Se desideriamo invece cambiare le caratteristiche del rettangolo quali ad

[4] Naturalmente, nulla vieta di usare anche le altre forme. Il rettangolo e il cerchio tuttavia, sono quelle maggiormente utilizzate.

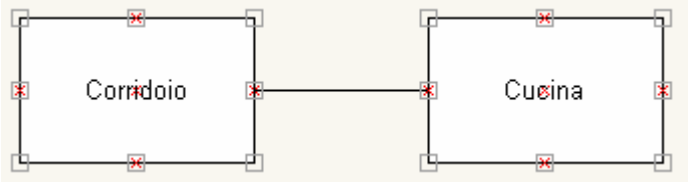
esempio il colore delle linee e il loro spessore, basta premere contemporaneamente i tasti Alt+Enter (a figura selezionata):



Tutto quanto fin qui detto vale naturalmente anche per le altre figure.
Vediamo adesso come collegarle:



Tutto quello che dobbiamo fare è selezionare la linea con il tasto sinistro del mouse e trascinare le sue estremità sulle crocette rosse delle figure poste ai rispettivi lati[5]:



Dopo essersi poi assicurati di aver selezionato tutto (premendo contemporaneamente i tasti CTRL + A, oppure selezionando con il tasto sinistro del mouse e il tasto SHIFT premuto le figure che a mano a mano c'interessano) creiamo un gruppo premendo contemporaneamente i tasti CTRL + G. In questo modo, è possibile spostare come blocco unico tutte le figure anziché una sola per volta[6].
Per ottenere poi una preview di stampa basta premere il tasto F4 (seguito poi dalla pressione del tasto ESC per uscire). Per salvare infine il tutto (ovvero la nostra mappa) occorre selezionare la funzione Export Page dal menu File e salvare nel formato png[7] (selezionabile dall'apposito menu a tendina che appare nella finestra di salvataggio). Buon lavoro a tutti…

[5] Durante quest'operazione molti ricorrono all'uso della lente d'ingrandimento (attivabile premendo il tasto F6).
[6] Naturalmente è anche possibile separarlo per riottenere i singoli oggetti: basta selezionarlo con il tasto sinistro del mouse e premere contemporaneamente i tasti CTRL + U.
[7] Acronimo di Portable Network Graphics. È un formato grafico che riesce a comprimere un'immagine senza perdere troppo in termini di qualità (a differenza di quello che spesso avviene con il formato jpeg).

APPENDICE G - SCRIVERE, DISTRIBUIRE E REPERIRE DELLE AVVENTURE TESTUALI

Spesso si sente parlare di un'avventura testuale in termini di qualcosa che si può paragonare, in qualche modo, ad una sorta di "racconto interattivo", ma le cose non stanno proprio così. Il racconto (così come il romanzo) è un testo che il lettore legge passivamente, limitandosi cioè a fungere da "spettatore": per far sì che la sua attenzione sia stimolata il più possibile occorre l'impegno, da parte dell'autore, di narrare ogni evento nei minimi particolari. Ecco un esempio della descrizione di un ipotetico tavolo:

"Quel tavolo in noce color marrone destava non poco la mia curiosità. Era appartenuto a una mia vecchia zia di Cuneo, e ne aveva viste di tutti i colori. Ricordo ancora quando io e mio fratello ci salivamo sopra con i nostri piedini nudi e ci divertivamo a giocare alla "boxe". Il tavolo restava fermo, impossibilitato a muoversi, sopportando le torture infantili di due demonietti scatenati che non facevano altro che urlare per tutto il giorno. Ora era lì, davanti a me, nella stanza semivuota di una casa tetra e polverosa. Su di esso era poggiato un libro con la copertina di pelle rossa. Lo raccolsi e cominciai a sfogliare le prime pagine. Era scritto in una strana lingua che non riuscivo proprio a comprendere…"

Ecco adesso uno dei tanti modi possibili di tradurre questo breve testo in un'avventura testuale:

Sei in una stanza tetra e polverosa, come tutto il resto di questa casa abbandonata ormai da anni. Davanti a te puoi vedere un tavolo, sul quale è poggiato un libro.

>esamina il tavolo
È un vecchio tavolo in noce, tutto sporco e impolverato.

>esamina il libro
È rilegato con una copertina di pelle rossa.

>prendi il libro
Preso.

>leggi il libro
Provi a sfogliare le prime pagine. È scritto in una strana lingua che non riesci proprio a comprendere.

Come potete ben vedere, il risultato è decisamente diverso. Stiamo descrivendo lo stesso tavolo, ma sono gli obiettivi che cambiano. In un'avventura testuale il lettore diventa a sua volta un giocatore, perché deve intervenire attivamente all'interno della storia per poter proseguire: la scrittura è più scarna, i contorni meno definiti, i dettagli nascosti… tutto questo perché è il giocatore stesso che li deve scoprire, altrimenti che gusto ci sarebbe?
Un'avventura testuale è quindi un gioco testuale; e se in un racconto l'attenzione del lettore la si deve catturare con delle opportune descrizioni, in un'avventura testuale occorre invece rifarsi al concetto basilare di interazione, che dipende da quello che l'avventura testuale stessa ci permette di fare.
Durante l'esecuzione dell'avventura, il giocatore effettua delle azioni, che devono essere espresse in termini di comandi digitati attraverso la tastiera del computer: PRENDI IL PACCO, SPOSTA IL TAVOLO, VAI A SUD, APRI LA PORTA sono solo alcuni dei possibili esempi, e il loro limite dipende dal tipo di parser implementato. Con quest'ultimo termine si intende, come afferma giustamente Stefano Gaburri nel suo articolo apparso sul numero 2 di Terra d'If[1], "la parte di software che si occupa di tradurre la stringa testuale di comando in un formato intelligibile al motore dell'avventura". In poche parole, è grazie ad esso se il computer è in grado di comprendere i comandi digitati dalla tastiera; durante il gioco infatti, c'è un continuo "dialogo" tra il parser e il giocatore, e sulla base

[1] Terra d'If è una fanzine, curata da Roberto Grassi, dedicata esclusivamente alle avventure testuali. È possibile scaricare i vari numeri all'indirizzo http://www.ifitalia.info/pmwiki/pmwiki.php?n=Riviste.TerraDIf.

delle risposte che quest'ultimo riceve, si riesce a capire se si sta proseguendo all'interno del gioco oppure no.
Ma avventura vuol dire anche programmazione, perché quando si vuole scriverne una bisogna prendere in considerazione la storia in sé ma anche il codice vero e proprio. Infatti, come ho già detto all'inizio del capitolo 2, la sequenza di testi, domande e risposte che il vostro computer vi mostra a mano a mano che proseguite nel gioco, è il risultato di una sequenza di istruzioni che nel loro insieme costituiscono il listato o codice sorgente. Le istruzioni da usare variano a seconda del linguaggio usato, ma mi preme far notare che programmazione vuol dire anche azione. Strumenti più evoluti come Tads o Inform permettono infatti di crearne delle nuove, che possono differire anche di parecchio rispetto a quelle previste di default. In Lurking Horror della Infocom, ad esempio, a un certo punto occorre agganciare l'estremità di una robusta catena a un gancio conficcato nel muro e l'altra estremità alla cabina di un ascensore. Il giocatore deve poi salire al piano superiore e chiamare quest'ultima con l'apposita pulsantiera: così facendo, l'ascensore comincia a salire insieme alla catena e nel muro si forma un bel buco che funge poi da passaggio…
Questo non significa che non si possono scrivere delle ottime avventure testuali utilizzando le sole azioni di default, ma più si è bravi a programmare, più si riesce a scriverne di "personalizzate" o, per meglio dire, di "creative". Naturalmente, occorre non esagerare con la creatività: le azioni che il giocatore deve effettuare possono avere una diversa difficoltà, ma devono sempre essere reali all'interno del contesto in cui sono inserite. In Ruins, per esempio, l'azione di mangiare il fungo per aprire il passaggio della piramide sepolcrale è davvero assurda e inusuale: neanche il più abile dei giocatori penserebbe a uno stratagemma simile e questo non va decisamente bene. Così come non va bene dare un nome univoco all'oggetto dichiarato: se io definisco un cupo ulivo all'interno di una foresta, non posso chiamarlo solo 'ulivo', altrimenti quando il giocatore agirà su di esso (per esempio esaminandolo) sarà obbligato a usare solo quella parola per far sì che il parser potrà riconoscerlo come oggetto in quanto tale (sarà quindi possibile scrivere ESAMINA L'ULIVO ma non ESAMINA L'ALBERO). Ricorrerò allora, oltre al nome originale, anche a dei sinonimi quali ALBERO, ALBERELLO, PIANTA, CUPO, ecc.
Molto importante è poi l'inserimento nel gioco degli aiuti. Oltre alla classica spiegazione dei comandi più usati (introdotta nel paragrafo 4.9 con l'esempio dell'estensione dmenus.h) è anche possibile inserire degli aiuti relativi all'utilizzo di tutti i comandi previsti di default (come spiegato nel paragrafo 4.9 con l'estensione command_it.h). Molto importante è anche la fase di betatesting; un'avventura infatti, si può definire completamente conclusa solo dopo che è passata in "rassegna" ai diversi betatester, a coloro cioè che la giocano in cerca di errori che possono riguardare sia il testo in sé (errori di sintassi, parti mancanti, ecc.) che il gioco vero e proprio (presenza di situazioni senza uscita, enigmi troppo difficili e incomprensibili, ecc)[2].

Esistono poi due diverse correnti di pensiero che distinguono le avventure testuali italiane in quelle "vecchio stile", che danno maggiore priorità al gioco (come quelle di Enrico Colombini e di Bonaventura Di Bello[3]), e quelle "moderne" che danno invece maggior risalto alla trama e allo sviluppo narrativo. A mio modesto parere vanno bene entrambe, sebbene io propenda più per queste ultime (ma è comunque solo una questione di gusti personali). È però anche vero che, come afferma giustamente Bonaventura Di Bello nella sua intervista apparsa sul numero 2 di Terra d'If, "il

[2] Sulla rivista Terra d'If sono apparsi diversi articoli inerenti agli argomenti fin qui trattati. Fra questi, mi preme citarne almeno quattro: la "Guida per principianti all'interactive Fiction" di Joe W. Aultman (apparso sul numero 1), i "10 suggerimenti per un ottimo 'Game Design'" di C. E. Forman (apparso sul numero 3), "Oh no! Betatest!" di C. E. Forman (apparso sul numero 5) e soprattutto la "Carta dei diritti del giocatore" di Graham Nelson (apparso sul numero 7).
[3] Enrico Colombini e Bonaventura Di Bello sono stati gli autori italiani di avventure testuali più famosi degli anni '80. In particolare, il secondo ha scritto tutte le avventure apparse sulle riviste Viking ed Explorer, che si possono reperire sul sito del Progetto Lazzaro (http://www.progettolazzaro.org/), su IfItalia o all'indirizzo http://ready64.org/; per poterle eseguire, avete bisogno di un emulatore del vecchio e glorioso Commodore 64 e personalmente vi consiglio l'ottimo e gratuito Vice. Le avventure di Enrico Colombini, invece, girano sotto MS-DOS e si possono scaricare dalla sua home page (http://www.erix.it/avventure.html).

giocatore deve concentrarsi sugli enigmi e sulla trama, con la possibilità di ottenere appunto una descrizione dettagliata e 'romanzata' di luoghi, oggetti ed eventi solo la prima volta che incorre in essi".

Per quanto riguarda i sistemi con i quali creare le avventure testuali, oltre a Inform sono disponibili il M.A.C. (Mystery Adventure Creator) di Paolo Lucchesi e il vecchio Modulo Base di Enrico Colombini, più semplici da usare rispetto al primo ma anche più limitati.

Parlando invece della distribuzione, occorre tirare nuovamente in ballo il parser, definito da qualcuno come "un joystick a sinonimi che simula solo una banalizzazione del linguaggio naturale". In poche parole, dal momento che in un'avventura testuale non è possibile utilizzare dei comandi del tipo ADESSO VAI A OVEST PER 10 METRI E POI DIRIGITI VERSO NORD (ma molto più semplicemente VAI A OVEST e VAI A NORD → banalizzazione del linguaggio naturale) occorre capire come darli (joystick a sinonimi). Infatti, come afferma Paolo Maroncelli, "chi si avvicina per la prima volta alle avventure testuali si sbizzarrisce di solito in frasi mirabolanti e incomprensibili". Occorre quindi capire quali sono i meccanismi e adeguarsi di conseguenza. Per far sì che questo avvenga, l'unico modo, secondo me, è quello di distribuire la propria avventura testuale insieme alla soluzione, perché includere la sola introduzione che spiega quali sono i tipici comandi standard non basta, soprattutto nelle avventure più complesse come quelle della Infocom. Ovviamente quando parlo di soluzione intendo quella passo-a-passo (walkthrough in inglese) che altro non è che il mero elenco dei comandi da digitare per portare a termine il gioco. Mi sembra però giusto ricordare che esiste anche la soluzione sotto forma di "consigli" (hints in inglese) che spiega, con una sequenza alternata di domande e risposte, cosa si deve fare in una determinata situazione prevista dal gioco stesso. Riassumendo, PER DISTRIBUIRE IN MANIERA OTTIMALE UN GIOCO DI QUESTO TIPO OCCORRE SECONDO ME CREARE UN FILE COMPRESSO (MEGLIO SE IN FORMATO ZIP CHE È QUELLO PIÙ USATO) AL CUI INTERNO INCLUDERE IL GIOCO, L'INTRODUZIONE, UN'EVENTUALE MAPPA E LE SOLUZIONI PASSO-A-PASSO E "A CONSIGLI". In questo modo si accontenteranno anche quei giocatori più esperti che, non riuscendo più ad andare avanti, preferiranno leggere gli hint e ragionarci un po' sopra piuttosto che ritrovarsi la "pappetta pronta". Ma esiste anche un'altra possibilità: quella cioè, di chiedere aiuto a qualcuno collegandosi direttamente al gruppo di discussione italiano dedicato alle avventure testuali (it.comp.giochi.avventure.testuali) tramite Outlook, Internet, o dei veri e propri newsreader (come ad esempio FreeAgent).

MA OCCORRE ANCHE SAPERE DOVE DISTRIBUIRE QUESTE BENEDETTE AVVENTURE. È PRESTO DETTO: BASTA FARE RIFERIMENTO A IFITALIA[4], il portale italiano dedicato alle avventure testuali italiane, dal quale potete anche reperire tutte (o quasi) quelle scritte dagli anni '80 fino ad oggi. La maggior parte di esse sono in formato MS-DOS e INFORM. Per le prime, occorre semplicemente eseguire il file contrassegnato dall'estensione .exe, assicurandosi però di avere estratto in una directory qualsiasi tutti i file dell'archivio[5] (e non solo l'eseguibile); per le seconde, invece, occorre usare un programma interprete come Windows Frotz 2002 di David Kinder. Per le avventure in MAC, vale la stessa regola delle avventure in formato MS-DOS.

Se siete invece interessati alle avventure testuali in inglese (o in altre lingue come il francese, il tedesco e lo spagnolo) vi consiglio allora di collegarvi all'indirizzo http://wurb.com/if/ dove troverete di tutto e di più. Vi ricordo anche che questo tipo di giochi ben si adatta all'apprendimento (gratuito) di altre lingue, per via della semplicità dei comandi da dare al parser e la descrizione dei luoghi che, per i motivi che abbiamo visto, sono spesso abbastanza semplici.

[4] Per distribuire le vostre avventure testuali su IfItalia dovete registrarvi: per sapere come farlo contattate direttamente il gestore del sito. N.B. → per chi non lo sapesse, il termine nickname indica un nome qualsiasi, così come download significa scaricare un file da; il termine upload, invece, significa mandare un file a.

[5] Su Internet, la maggior parte dei dati è distribuita sotto forma di file compressi. Per decomprimerli (o, se preferite, estrarli dal file compresso) occorrono degli appositi programmi di compressione/decompressione; dal momento che il formato attualmente più diffuso è lo zip, il programma che vi occorre è WinZip, reperibile praticamente dappertutto.

E per quanto riguarda le soluzioni e i listati delle avventure testuali scritte da altri?
Per le prime, se le cercate in italiano, fate sempre riferimento a IfItalia; se le cercate invece in inglese o in qualche altra lingua, andate all'indirizzo http://solutionarchive.com/ dove troverete oltre 700 soluzioni online (la maggior parte di esse in lingua inglese). Per i listati, invece, potete fare riferimento all'indirizzo http://www.ifitalia.info/pmwiki/pmwiki.php?n=Main.Sorgenti, oppure richiederli via e-mail ai rispettivi autori che (bontà loro) decideranno se accontentarvi o meno.

Buona avventura a tutti…

APPENDICE H – LINK VARI

Il mondo delle avventure testuali, per quanto forse strano possa sembrare ad alcuni voi, è molto più vasto di quello che a prima vista potrebbe sembrare. Non si possono quindi non citare almeno i principali siti che si occupano dell'argomento in questione.

Per quanto riguarda la programmazione in Inform e Glulx abbiamo:

- EMILY SHORT HOME PAGE (http://emshort.wordpress.com/), il sito ufficiale della straordinaria Emily Short. Di notevole interesse è la sezione dedicata ai PNG (un suo articolo - tradotto in italiano - sull'argomento potete scaricarlo dalla mia home page);
- GLULX (http://www.eblong.com/zarf/glulx/), il sito ufficiale di Glulx, il linguaggio di programmazione basato su Inform che permette di creare delle avventure testuali multimediali;
- GULL (http://adamcadre.ac/gull/index.html), il sito dedicato a Gull, il manuale ufficiale di Glulx scritto da Adam Cadre (la versione in italiano, a cura del bravissimo Paolo Vece, potete scaricarla dalla mia home page);
- INFORM HOME PAGE v6 (http://www.inform-fiction.org/inform6.html), il sito ufficiale di Inform 6 (sul quale è basato questo manuale), dove è possibile trovare tutto ciò che può servire per programmare con questo linguaggio (l'Inform Designer Manual, le estensioni e molto altro ancora);
- INFORM HOME PAGE v7 (http://inform7.com/), il sito ufficiale di Inform 7 che permette di creare un'avventura testuale in una modalità molto più vicina al modo di pensare "umano" (con un linguaggio, cioè, naturale).
- INFORM-ITALIA (http://www.inform-italia.org/), il sito italiano di Inform, nel quale è possibile trovare tutti gli strumenti per programmare con Inform in italiano (Infit, manuali vari e altro ancora);
- JIM FISHER HOME PAGE (http://www.onyxring.com/orlibrary.aspx), una pagina dedicata alle utilissime ORL Library;
- JIF HOME PAGE (http://www.slade.altervista.org/jif.html), il sito ufficiale di Jif, uno dei programmi più completi (e forse più usati) per programmare in Inform e Glulx;
- MILLE E UNA AVVENTURA (http://milleuna.sourceforge.net/), il sito italiano dedicato a Inform 7 (contenente anche le librerie per la lingua italiana), la nuova versione di questo linguaggio scritto da Graham Nelson;
- PAOLO LUCCHESI HOME PAGE (http://www.paololucchesi.it/at/index.html), il sito ufficiale di Paolo Lucchesi. Qui potete trovare tutte le sue librerie (una di queste è la mitica wtalk.h) oltre naturalmente a tutte le sue avventure e molto altro ancora;
- PARCHMENT (http://code.google.com/p/parchment/), la pagina ufficiale di questo utilissimo strumento, che consente di giocare online a tutte le avventure scritte in Inform;
- ROGER FIRTH'S IF PAGES (http://www.firthworks.com/roger/index.html), il sito ufficiale del grande Roger Firth. Di notevole interesse sono la sezione Informary (che raccoglie tutte le principali novità mondiali su Inform) e la Guida a Inform per principianti (scaricabile all'indirizzo http://www.inform-italia.org/docs/gip o eventualmente dalla mia home page);
- VINCENZO SCARPA HOME PAGE (http://www.vincenzoscarpa.it/inform/manuale/), la mia home page dalla quale potete scaricare questo manuale (in versione stampabile nel formato PDF) e molto altro ancora;
- WARMAGE HOME PAGE (http://pitermos.niccolai.cc/diven.php), il sito dedicato a Warmage e al mondo di Pitermos. Frutto della genialità di Giancarlo Niccolai, quest'avventura in puro stile fantasy è anche una dimostrazione di quello che sono in grado di fare le sue MFS library (un suo articolo sull'argomento potete scaricarlo dalla mia home page);

- WIDE HOME PAGE (http://wide.berlios.de/), la pagina ufficiale di questo nuovo e versatile editor per Inform, scritto in C++ da Alessandro Schillaci e Paolo Lucchesi.

Per quanto riguarda invece la programmazione con altri linguaggi, abbiamo:

- ADRIFT (http://www.adrift.org.uk/cgi/new/adrift.cgi), dedicato a Adrift, un linguaggio di programmazione che permette di creare delle avventure testuali in maniera più semplice (ma anche più limitata) di Inform. Le librerie in italiano (scritte da Roberto Grassi), potete scaricarle dalla mia home page;
- AGT (http://www.ifarchive.org/indexes/if-archiveXprogrammingXagt.html), un linguaggio interpretato scritto da Robert Masenten e disponibile per moltissime piattaforme;
- ALAN (http://www.alanif.se/), un altro linguaggio adatto allo scopo;
- HUGO (http://www.generalcoffee.com/hugo/gethugo.html), un linguaggio di programmazione molto interessante che consente di creare avventure testuali complesse (al momento solo in inglese);
- IDRA (http://www.erix.it/idra.html), il sito dedicato a Idra, uno strumento che, come dice lo stesso autore, consiste in un versatile meccanismo che consente di realizzare con facilità racconti-gioco o altre opere interattive, leggibili con un normale browser Web su ogni sistema operativo, sia in locale (cioè sul computer stesso) sia attraverso un collegamento Internet (un effetto simile potete ottenerlo in Inform tramite la cyoa.h scritta da Paolo Lucchesi);
- MAC (http://www.paololucchesi.it/mac/index.htm), il sito dedicato al MAC (acronimo di Mystery Adventure Creator), uno strumento per creare delle avventure testuali italiane;
- MODULO BASE (http://www.erix.it/), scritto dal grande Enrico Colombini che consente di creare delle avventure testuali sotto MS-DOS tramite il vecchio GW-BASIC. È anche possibile scaricare il libro ad esso dedicato ("Avventure per MS-DOS");
- TADS (http://www.tads.org/), un linguaggio di programmazione molto potente che consente di creare delle avventure testuali simili a quelle scritte in Inform (al momento, purtroppo, solo in inglese).

Per quanto riguarda i giochi e le soluzioni abbiamo:

- ADVENTURE SOLUTION ARCHIVE (http://solutionarchive.com/), il sito per eccellenza delle soluzioni e delle mappe (solo in inglese);
- BAF'S GUIDE TO THE IF ARCHIVE (http://wurb.com/if/), uno splendido sito dal quale potete scaricare una miriade di avventure, soluzioni e recensioni;
- DOROTHY MILLARD HOME PAGE (http://dorothyirene.fateback.com/), un altro bellissimo sito dedicato alle soluzioni (solo in inglese);
- IF-ITALIA (http://www.ifitalia.info/pmwiki/pmwiki.php), il portale delle avventure testuali italiane. Qui potete trovare di tutto: dalle prime avventure scritte nel vecchio MS-DOS fino alle più recenti;
- INTERACTIVE FICTION ARCHIVE (http://www.ifarchive.org/), il sito che contiene tutto (o quasi) quello che si può volere nell'ambito delle avventure testuali;
- PROGETTO LAZZARO (http://www.progettolazzaro.org/)[1], il sito che raccoglie tutte (o quasi) le avventure testuali scritte negli anni '80 da Bonaventura Di Bello e apparse sulle riviste Viking ed Explorer. Per eseguirle, occorre utilizzare un emulatore del vecchio e glorioso Commodore 64 (come ad esempio l'ottimo Vice);
- WORLD OF SPECTRUM (http://www.worldofspectrum.org/textadv/index.html), ovvero il sito che contiene tutte le avventure testuali inglesi (più di mille) uscite per questo fantastico computer a 8 bit. Per eseguirle, occorre utilizzare un apposito emulatore (come ad esempio l'incredibile Spectaculator).

[1] Nel momento in cui scrivo, il sito sembra non funzionare più da parecchio tempo. Le avventure in questione potete comunque ugualmente scaricarle agli indirizzi http://www.ifitalia.info/pmwiki/pmwiki.php e http://ready64.org/.

Per quanto riguarda le riviste abbiamo:

- SPAG (http://sparkynet.com/spag/), molto famosa per le sue recensioni (solo in inglese);
- TERRA D'IF (http://www.ifitalia.info/pmwiki/pmwiki.php?n=Riviste.TerraDIf), la fanzine italiana delle avventure testuali, curata dal mitico Roberto Grassi;
- XYZZY (http://www.xyzzynews.com/), il cui nome è tutto un programma (solo in inglese)[2].

Per quanto riguarda i gruppi di discussione (i cosiddetti newsgroup) abbiamo[3]:

- IT.COMP.GIOCHI.AVVENTURE.TESTUALI (news:it.comp.giochi.avventure.testuali), il gruppo italiano dedicato agli appassionati e agli sviluppatori italiani che vogliono discutere di avventure testuali;
- REC.ARTS.INT-FICTION (news:rec.arts.int-fiction), in cui si parla della creazione delle avventure (solo in inglese);
- REC.GAMES.INT-FICTION (news:rec.games.int-fiction), in cui si parla dei giochi in generale (solo in inglese).

Per quanto riguarda infine i siti di interesse generale, abbiamo:

- ADVENTURELAND (http://adventure.if-legends.org/) un sito che funge da "enciclopedia" delle avventure testuali rilasciate per i vari computer;
- BRASS LANTERN (http://www.brasslantern.org/) un altro bel sito inglese dedicato a questo genere di giochi. Di notevole importanza è la sua guida dedicata ai giocatori alle prime armi;
- COLOSSAL CAVE ADVENTURE HOME PAGE (http://www.rickadams.org/adventure/), il sito dedicato a Colossal Cave, la prima avventura testuale ad essere mai stata scritta per un computer;
- GARGOYLE (http://ccxvii.net/gargoyle/), il sito di un bellissimo programma che esegue avventure testuali di tutti i tipi (tra cui, naturalmente, Inform e Glulx). Assolutamente da provare;
- HOME OF THE UNDERDOGS (http://www.homeoftheunderdogs.net/), un immenso archivio di giochi "underdog" con una nutrita sezione dedicata all'Interactive Fiction;
- IF CORNER (http://www.vece.net/if/) l'home page di Paolo Vece con utili informazioni e link riguardanti l'IF;
- IF RATINGS (http://www.carouselchain.com/if/), un sito che raccoglie opinioni, commenti e giudizi sulle avventure testuali;
- IF-REVIEW (http://www.ministryofpeace.com/if-review/), il sito dedicato alle recensioni sulle avventure testuali;
- IL BLOG DI ALEXAIN (http://avventuriero.wordpress.com/) , un blog che riporta notizie e articoli sul mondo dell'Interactive Finction;
- INFOCOM (http://www.csd.uwo.ca/Infocom/), un bel sito inglese che riporta tutto quello che c'è da sapere sulla Infocom, la "casa madre" delle avventure testuali;
- INFORMARY (http://www.firthworks.com/roger/informary/index.html), la sezione dell'immenso sito di Roger Firth che raccoglie tutte le principali novità mondiali su Inform;
- IFWIKI (http://www.ifwiki.org/index.php/Main_Page), l'enciclopedia per eccellenza delle avventure testuali;
- L'AVVENTURA È L'AVVENTURA (http://www.avventuretestuali.com/), il bel sito italiano gestito da Francesco Cordella, contenente numerose informazioni su questo genere di giochi. Di notevole interesse sono le interviste rilasciate da alcuni autori del periodo d'oro, ovvero gli anni '80;
- LEVEL 9 MEMORIAL (http://l9memorial.if-legends.org/html/home.html), un sito dedicato a questa famosa software house, produttrice di capolavori come "Emerald Isle" e "The Archers";

[2] Per i più curiosi, Xyzzy è una parola magica usata in Colossal Cave Adventure, la prima avventura testuale ad essere mai stata scritta per un computer.

[3] I newsgroup possono essere consultati anche su Internet o con i newsreader (come ad esempio FreeAgent).

- MAGNETIC SCROLLS MEMORIAL'S (http://msmemorial.if-legends.org/memorial.php), il sito dedicato a questa prestigiosa software house che, grazie alle sue avventure testuali, fu in grado di rivaleggiare per un certo periodo con la Infocom;
- MARCO VALLARINO HOME PAGE (http://www.marcovallarino.it/), il sito ufficiale dell'autore di "Enigma" e "Il giardino Incantato";
- PAOLO MARONCELLI HOME PAGE (http://www.maroncelli.net/if/), la sezione del sito di Paolo dedicata al mondo dell'Interactive Fiction;
- SAGA - SCOTT ADAMS GRAND ADVENTURES (http://www.msadams.com/), il sito ufficiale di questo straordinario autore di avventure testuali;
- TOMMASO CALDAROLA HOME PAGE (http://www.caldarola.net/if.html), un sito dedicato ad un altro bravissimo autore italiano di avventure testuali.

È tutto. Buona navigazione a tutti…