L'essenziale per cominciare

Per programmare un'avventura testuale è sufficiente un codice sorgente come il seguente:

```
! Teniamo questa riga libera
Include "Parser";
! Teniamo questa riga libera
Include "Verblib";
Include "Replace";

! Da qui si dichiarano gli oggetti
Object stanza_1 "Stanza"
with description "Ti trovi in una stanza.",
has light;

! Da qui si dichiarano le routines
[ Initialise;                    ! Apriamo una quadra, scriviamo il nome della routine e ;
location = stanza_1;             ! Inseriamo le istruzioni della routine
];                               ! Chiudiamo la quadra e poi un altro ;

Include "ItalianG";  ! Sempre alla fine del listato
```

Se provate a copiarlo e incollarlo in WIDE potrete già eseguirlo e notare che funziona correttamente. [noterete anche altre cose di cui parleremo in seguito e di cui, per il momento, non dovete preoccuparvi]

Vediamo gli elementi caratteristici di questo listato:
Ogni volta che compare un punto esclamativo non incluso nelle virgolette, tutto ciò che sta alla sua destra (sulla stessa linea) non viene tenuto in considerazione dal compilatore: questo è un modo utile ai programmatori per inserire **commenti**, ossia annotazioni personali, direttamente nel codice sorgente. Vedrete che WIDE ci ricorda dei commenti colorandoli di verde. In questo esempio, ci vogliamo ricordare di tenere delle righe libere: vedremo nei prossimi articoli che in quegli spazi tenuti da parte sarà possibile inserire molte cose interessanti.

Le successive tre istruzioni, sempre necessarie, che iniziano con *Include*, detto molto brutalmente, servono per includere istruzioni già preformate che servono per far funzionare tutto il programma (e vanno scritte in questo esatto ordine, come quasi tutto quel che segue), tra queste, ad esempio, sono presenti i verbi già preimpostati nelle librerie: pur non avendoli definiti nel nostro codice sorgente, è possibile, ad esempio, cantare o pensare ed ottenere un'adeguata risposta (vedremo più avanti quali sono questi verbi).

Dopo i tre *Include* abbiamo definito un oggetto. Entriamo nel dettaglio.

Tutto ciò con cui il giocatore interagisce è un oggetto, sia esso una locazione, come ad esempio la stanza dell'esempio, un oggetto "vero e proprio" o un personaggio non giocabile, quindi si dichiarano tutti e tre questi tipi con la dicitura Object.

In seguito inseriamo il nome con cui *il programma* capisce che deve rifarsi a quell'oggetto, nel nostro caso stanza_1, e infine il nome che vogliamo compaia sullo schermo compreso tra doppie virgolette (anche in questo caso WIDE ci informa che tutto ciò che è fra virgolette è una cosa che verrà stampata a schermo colorandola di rosso!). Per capire meglio facciamo un salto avanti alla routine *Initialise*,

anche questa sempre necessaria. Lo scopo principale di questa routine è definire la locazione di partenza: se avessimo scritto

location = Stanza;

il compilatore ci avrebbe risposto con un messaggio di errore, non trovando nessun oggetto con quel nome. Lo stesso erroneo risultato si sarebbe ottenuto con

location = "Stanza";

infatti, come detto, il testo quotato serve solo per l'interazione del giocatore, che in effetti giocando potrà digitare *>esamina stanza* ma non di certo *> esamina stanza_1* (non sa nemmeno dell'esistenza del nomestanza_1... e se lo farà gli verrà risposto che non esiste nulla con quel nome!).

Chiaramente è necessario indicare sia il nome utilizzato dal programma che quello utilizzabile dal giocatore.

Pur essendo opzionale, nella riga immediatamente successiva a quella in cui abbiamo dichiarato l'oggetto, dobbiamo aggiungere **with** per creare alcune *proprietà* che caratterizzano l'oggetto, nel nostro esempio la sua descrizione. Quello proposto è il modo caratteristico per proporre la descrizione di un oggetto quando viene esaminato (nota: nell'esempio, trattandosi della descrizione di una stanza, il comando adeguato non è *> esamina*, ma *> guarda*), anche se ne esistono altri, che vedremo in seguito, quasi del tutto equivalenti.

Infine vediamo **has**, anche questo non obbligatorio, che serve per "attaccare" all'oggetto i cosiddetti *attributi*. In questo caso il solo attributo conferito è *light*, che sta ad indicare che la stanza è illuminata, ovvero che il giocatore riconosce ciò che vi è di visibile (ebbene sì: possiamo inserire anche oggetti "invisibili"!) all'interno. Ricordate sempre di aggiungere questo attributo: per capirne il perché, provate a creare un codice sorgente di una stanza senza alcun attributo.

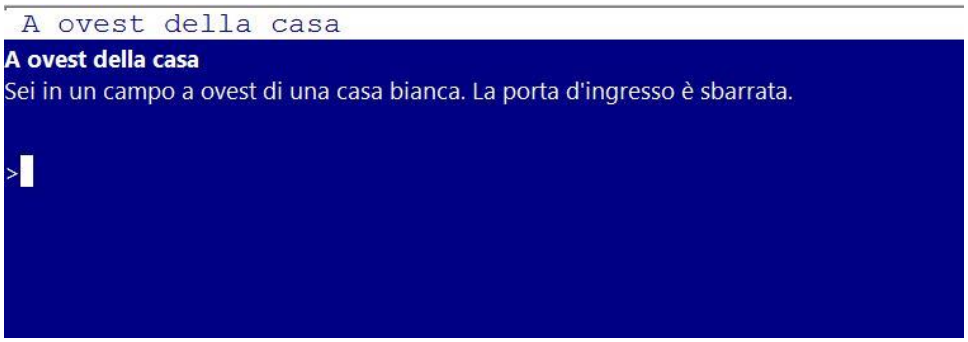
Da notare che questo non è un errore… tutt'altro! Questa situazione viene molto spesso utilizzata nel game design e la incontreremo anche in seguito.

Da notare anche che ogni istruzione termina col punto e virgola, ma nel corpo dell'oggetto il punto e virgola compare solo alla fine; al termine delle proprietà è presente la semplice virgola (laddove vengano aggiunti anche attributi). Questi segni grafici non sono interscambiabili e, avendo un utilizzo preciso, non vanno confusi (per credere, provate a sostituire la virgola col punto e virgola). Non vanno né virgola né punto e virgola nella riga in cui è dichiarato l'oggetto.

In ultimo vediamo una quarta *Include*, che serve per includere una libreria che "traduce" i comandi italiani in inglese, ma anche di questo ci occuperemo più avanti. Anche la posizione di questa Include è fissa, dovendo sempre stare al termine del listato.

Anche per questa volta è tutto.

Termino lasciando un esercizio che consiste nel programmare la prima locazione dell'avventura testuale Zork così come compare nell'edizione italiana a cura di Whovian e Ragfox: cercate di ottenere questo risultato


A ovest della casa
A ovest della casa
Sei in un campo a ovest di una casa bianca. La porta d'ingresso è sbarrata.
>