Manuale di Utilizzo Avanzato di AWS

Alcuni consigli su come scrivere buone avventure

Scrivere un'avventura richiede prima di tutto un'ispirazione, cioè l'idea base della trama, l'ambientazione e lo schema generale della storia.
Una volta che si ha ben chiaro in mente quello che si intende fare, bisogna passare al dettaglio. Per poterla trasformare in un programma, la trama deve essere accuratamente definita in ogni particolare:

- La mappa dei luoghi in cui si svolge l'avventura.
- Il dizionario, cioè l'elenco delle parole conosciute dal programma.
- Gli oggetti ed i personaggi che l'avventuriero può incontrare.
- Le condizioni/azioni, cioè le frasi che hanno effetto sul gioco nelle varie situazioni.

Non commettete anche voi l'errore di mettervi a scrivere un'avventura senza aver ben chiara in mente (o, meglio ancora, su carta) la trama ed i dettagli. Risparmiereste, sì, un poco di lavoro all'inizio, ma finireste col pagarlo caro. Andando avanti nella scrittura del gioco, finireste col trovarvi persi in una miriade di dettagli che non collimano, come in un giallo scritto male. Meglio dunque perdere qualche ora per progettare l'avventura nei minimi particolari.
Tutto gli esempi che ho disseminato per questo manuale, non sono improvvisati ma frutto di analisi a tavolino e su carta. Insomma, mangio della mia stessa cucina! Ovviamente, a fini didattici, nel manuale costruisco gli esempi un passo alla volta, aggiungendo man mano nuovi dettagli. Lo faccio soltanto per non presentarvi le difficoltà tutte insieme. Me li sono preparati nei dettagli prima di iniziare a scrivere.

La trama deve essere coerente. Le incoerenze spezzano il filo della vicenda, e rompono il fragile incantesimo che tiene avvinto il lettore/giocatore (è fiction interattiva, non dimenticatelo mai!). Parole magiche non hanno senso a bordo di un'astronave, un tesoro stona in un'avventura di spionaggio. Entrambe le cose fanno cadere la tensione e distraggono il giocatore dallo scopo principale.
L'atmosfera è importante, per fare in modo che il giocatore si immedesimi col personaggio. Anche senza lunghe descrizioni (comunque possibili in AWS), basta spesso un aggettivo per stimolare la fantasia del giocatore. "Una sala" è semplicemente una sala, ma "Una sala silenziosa" è tutt'altra cosa: c'è chi immagina la polvere dei secoli depositata sugli antichi arredi, chi sospetta che qualcosa si celi dietro il silenzio innaturale... In ogni caso, non è un posto qualunque (anche se, magari, non c'è niente di utile ai fini del gioco).

I dettagli contano

Una volta definita la sequenza principale del gioco, non abbiamo finito, anzi. Infatti, l'avventura che ne risulterà non sarà di certo interessante. Ci saranno molti "non capisco" o simili, e le uniche azioni valide saranno quelle che portano direttamente alla soluzione. E badate che negli anni ’80 e ’90 (ma mi sa anche oggi) c’erano in circolazione avventure che non si discostavano molto da questo modo di ragionare: o si fa l'azione giusta, o niente. Eppure, contrariamente a quanto pensano alcuni autori, perché un'avventura sia divertente non serve che abbia una mappa particolarmente estesa, o che presenti problemi difficilissimi. Deve invece essere molto curata nei dettagli, perché sono proprio questi a dare al giocatore l'illusione di vivere realmente dentro l'avventura, in un mondo fantastico creato da voi.
Però la rifinitura dei dettagli è la parte più laboriosa dell'intero lavoro, e potete aspettarvi di impiegare per questa ben oltre la metà del tempo complessivo che impiegherete a scrivere il racconto. Per ottenere un buon risultato, ci sono due cose da fare:

- Aggiungere tutti i dettagli che vi vengono in mente (e non copriranno mai tutto quello che un giocatore può scrivere);
- Far collaudare l'avventura a molte persone ed aggiungere nuovi dettagli che vengono fuori dalle loro sessioni di gioco. Scoprirete tentativi di fare cose a cui non avreste pensato in un milione di anni.

A questo punto armatevi di pazienza, ed introducetele (ampliando il vocabolario con le nuove parole). È così che si costruisce una bella avventura.
Molte volte cosa manca in un’avventura? Il finale. È ovvio che ci vuole una qualche forma di conclusione ma molto spesso, tipico di sceneggiatori trascurati, il finale è insulso, del tipo:”Bravo, hai vinto”.
E certo, ho passato ore ed ore a sputare (in senso metaforico) sudore e sangue e questa misera frase è tutta la mia ricompensa? Ci vuole qualcosa di più gratificante, direi non meno di una decina di righe dedicate ad osannare il successo nell’impresa, no?
Stessa cosa per il finale negativo. Un semplice:”Sei morto” non va assolutamente bene. Leggetevi questi due finali di Enrico Colombini, scritti per una sua avventura/tutorial relativa ad un agente segreto e dei microfilm da trafugare. Sono semplici accenni, nulla di approfondito e sofisticato. Eppure la differenza è notevole.
Finale negativo:
Fedeli agli ordini, i tuoi colleghi eliminano un agente incapace. La tua missione e' fallita. Di te non resterà che un polveroso fascicolo, presto dimenticato.

Finale positivo:
- Missione compiuta! -

Hai salvato il tuo Paese e ti sei guadagnata la promozione a:
######
(il tuo nuovo titolo e' così segreto che non posso nemmeno nominarlo).
Congratulazioni, ######!

Lo scopo dell'avventura è il divertimento dei giocatori, non il sadico compiacimento dell'autore che nessuno riesca a risolverla. I problemi devono essere abbastanza difficili da dare soddisfazione quando li si risolve, ma non impossibili, altrimenti non ci si diverte. Bisogna collaudare l'avventura, farla giocare a vari amici. Se troppo pochi riescono a risolvere un certo problema, è il caso di fornire qualche indizio in più. Evitare un errore grave, che le difficoltà siano risolubili con una sola azione, descritta con una sola frase valida. I collaudi servono soprattutto a questo. Se avete previsto l'azione "Rispondi al telefono", vi accorgerete che molti scrivono invece "Prendi il telefono" o "Alza la cornetta" o simili. Se ottengono solo risposte del tipo "Non capisco" o, peggio, "Non succede niente", probabilmente dopo un po' lasciano perdere.
Sono i dettagli che rendono interessante un'avventura, molto più che la complessità della vicenda. Più risposte inutili o secondarie sono state previste, più il gioco è appassionante ed i giocatori si divertono. Per esempio, in risposta ad "Accendi il televisore", una risposta tipo "Stanno trasmettendo il 15° episodio della quarta stagione di Dottor House" è senz'altro meglio del solito "Non capisco".
I test sono molto importanti. Prendete nota di tutto quello che scrivono i vostri amici, ed aggiungete risposte adeguate. I giocatori non riusciranno a staccarsi dal computer, e si chiederanno come diavolo fa il programma a *capire* quello che scrivono alla tastiera, non sapendo che siete stati voi a prevedere tutte le azioni che loro stanno facendo.

Se il nostro obiettivo è di scrivere IF che possa essere sia buona fiction che un buon gioco, è essenziale rendere vivi i personaggi, senza ricorrere all'intelligenza artificiale! Tutti i personaggi dovrebbero essere in grado di superare la domanda dell'editor del libro:"Perché il nano vi tira un'ascia?". Il bigliettaio prende il biglietto perché è il suo lavoro, il desiderio di dominare il mondo spinge Sauron a cercare l'Anello e così via. La risposta non deve essere profonda ma deve essere evidente dal contesto e dalle informazioni che sono state fornite.

Il giocatore dovrebbe riuscire a completare la storia senza usare informazioni guadagnate "morendo". Una tipica violazione di questa regola: nascondere una parola magica sul fondo di un pozzo pieno d'acqua cosicché la possiate vedere solo un attimo prima di affogare e poterla così utilizzare nella prossima reincarnazione.

Una violazione meno ovvia: il puzzle a tentativi fatali. Immaginate quattro porte identiche, di cui una che conduce fuori ed una che nasconde un esplosivo letale. Nella storia risultante dalla soluzione di questo problema, sarebbe molto più soddisfacente per il lettore/giocatore se ci fosse un modo per capire quale porta nasconde l'esplosivo, piuttosto che riuscire a risolvere l'enigma con un tentativo, fortunato si, ma a casaccio. La traccia potrebbe essere abbastanza difficile che il giocatore preferisca l'approccio della forza bruta, salva – prova – muori – ricarica – riprova. D'altronde chi potrebbe pensare a "Ascolta la porta a nord" per sentire il ticchettio della bomba? Ma almeno è li a spiegare le azioni del protagonista della storia in un modo soddisfacente da un punto di vista narrativo. Anche la sopravvivenza nel mondo reale può spesso dipendere da pura fortuna, la finzione può fornirvi pochi colpi di fortuna prima che il sospetto si insinui.

Una buona avventura contiene i seguenti ingredienti:

- una trama ben definita ed interessante
- una buona atmosfera
- originalità ed inventiva nei problemi
- molte risposte a molti possibili input da parte del giocatore
- un pizzico di humour

La trama

Tentate di creare una trama originale, senza abusare dei classici raccogli i tesori o salva la principessa. Siate sicuri che la vostra trama funzioni come avventura (un aereo dirottato potrebbe non essere il massimo). Le prime volte, tuttavia, non abbiate paura di utilizzare un qualche scenario scontato. Potrete diventare più originali in seguito.

Atmosfera

L'atmosfera dipende dalle descrizioni e dai messaggi. Una buona atmosfera non ha bisogno di grandi quantità di testo ma di un buon uso di aggettivi e avverbi. Per esempio:
Mi trovo in un corridoio. Ci sono torce sulle pareti. Est o Ovest:
oppure
Mi trovo in un corridoio freddo e umido che si snoda da est a ovest. Le pareti sono bagnate dalla luce tremolante di torce.
La seconda descrizione è sicuramente migliore.
A volte può essere utile includere le uscite nella descrizione, disabilitando la stampa delle uscite settando il marker 124. Se volete includerle cercate però di rendere la descrizione interessante: Il sentiero si snoda verso nord, tra le montagne.

Problemi

I problemi sono del tipo supera l'ostacolo oppure interagisci con un altro personaggio od oggetto. Ci dovrebbe sempre essere una ragione per risolvere un problema, di solito l'accesso ad un'altra locazione o il possesso di un oggetto. Importante è che i problemi siano attinenti alla trama. Sembrerebbe alquanto fuori luogo un robot nell'epoca di Re Artù. Il problema inoltre non dovrebbe essere così oscuro da essere irrisolvibile a meno che non siano state date delle tracce da qualche altra parte. La soluzione inoltre dovrebbe essere attinente al problema.
Esempio: per tenere lontano un vampiro il giocatore si aspetta di dover entrare in possesso di aglio, oppure acqua santa o un crocifisso, non certo di togliersi le scarpe! Che dite? Che non lo farebbe nessuno? E secondo voi ho inventato l'esempio? C'era una vecchia avventura in cui il vampiro si allontanava proprio togliendosi le scarpe!
All'inizio inoltre i problemi dovrebbero essere facili, per diventare più difficoltosi man mano che si procede nell'avventura. Evitare le morti improvvise: il giocatore dovrebbe sempre sapere cosa sta succedendo. Esempio: se vuole attaccare un leone a mani nude o camminare sulle sabbie mobili è chiaro che sta cercando guai, ma una passeggiata su un semplice sentiero non dovrebbe portare ad una morte improvvisa.

Risposte

Un'avventura dovrebbe avere molte risposte a input sbagliati, non limitarsi a dire non puoi. Ad esempio: il giocatore è sulla riva sud di un fiume, vicino un boschetto. La soluzione per attraversare il fiume è di costruire una zattera usando l'ascia ed una corda. Si possono allora scrivere messaggi per:
a) tentativo di attraversare il fiume senza zattera (Non so nuotare, il fiume è pieno di coccodrilli, l'acqua è troppo fredda per i miei gusti...)
b) tentativo di costruire la zattera senza ascia (tenti di abbattere un albero con un colpo di karate ma non ci riesci...)
c) tentativo di costruire la zattera senza la fune (i tronchi non sembrano voler stare insieme...)
d) aver costruito la zattera con successo ed aver attraversato il fiume

Humour

In genere lo humour nelle avventure si limita a commenti sarcastici su qualche cosa fatta non correttamente dal giocatore. Tuttavia, se ci si sente all'altezza, si può tentare di scrivere avventure totalmente demenziali.

Ultime raccomandazioni riguardano la credibilità e la persona (o le persone) che avranno il compito di provare l'avventura prima della distribuzione.
Credibilità può sembrare un concetto strano in un'avventura, dove di solito si parla di mondi fantastici, ma è bene sempre tener presente che porre un ghiacciaio vicino ad un deserto può sembrare innaturale anche al più stagionato degli avventurieri.

Le persone che proveranno il gioco

Utilizzatene sempre più di una: dai loro commenti e suggerimenti potrete correggere eventuali errori di ortografia, condizioni che non funzionano correttamente e aggiungere condizioni cui non avevate pensato in risposta a situazioni impreviste. Non suggerite loro le soluzioni ma cercate di capire dove si fermano e perché. Potrete così, eventualmente, aggiungere suggerimenti o altro che migliori il gioco.
Ecco altri spunti interessanti:
Punti: abbiamo visto due modi di fare punti in un gioco, ma potreste decidere che il punteggio non abbia significato nel vostro gioco.
Veicoli: auto, ascensori, mongolfiere, tappeti magici, navi spaziali – o qualunque altro dispositivo consenta al giocatore di muoversi.
Cambiare il giocatore: chi l'ha detto che il personaggio del giocatore debba essere un noioso essere umano? Trasformate l'insospettabile mortale in un proxy della realtà virtuale, un animale fantastico, un fantasma immateriale, un vampiro telecinetico. Non sapete quale scegliere? Allora il vostro gioco può avere più personaggi principali e potete scegliere con chi giocare.
Lo scorrere del tempo, macchine a tempo ed eventi: impostate un timer che segni il tempo, sconosciuto al giocatore ed attaccatelo ad una bomba; una porta che si apre solo una volta ogni dieci turni; un drago di poche parole e poca pazienza; una pattuglia di soldati; un orologio che, inquietante, annunci l'arrivo del tramonto e del destino. Cambiate i "titoli" sulla riga di stato in minuti, od in giorni.
Direzioni che cambiano: a nord c'è il nord? Non necessariamente. Cambiate gli oggetti di direzione del gioco in "avanti", "indietro" e così via. Siete su una nave? "Prua" e "poppa", "babordo" e "tribordo" possono fare per voi. Entrate in uno specchio ed avrete la mappa e tutte le direzioni riflesse.
Personaggi pseudo intelligenti: quanto è imprevedibile il comportamento di quell'impertinente del maggiordomo? Può parlare, muoversi, rubarvi la roba, avvelenare il vostro thè? Reagisce in maniera coerente alle azioni del giocatore? Tiene una sua agenda? Sebbene la creazione di un personaggio pseudo intelligente non sia cosa semplice, tuttavia aumenta tantissimo la sensazione di realtà del vostro gioco.

Funzionalità tecniche: cambiate la riga di stato, od il prompt dei comandi. Pulite lo schermo, o modificate il colore del testo e dello sfondo. Aspettate che il giocatore prema un tasto e quindi eseguite una qualche azione. Visualizzate un messaggio una parola alla volta.
Una furbata semantica l'ho vista tanti anni fa in Subterranean Encounter, un'avventura per il TRS-80 (chi se lo ricorda?) della Toucan Software. Dopo aver girato a nord alla "fork in the road", incontrate un eremita poco amichevole che vi strangola. Se aveste fatto qualcosa di surreale, esaminando la fork, avreste scoperto che "fork in the road" sarebbe alla biforcazione della strada, alla forca della strada, alla forchetta nella strada! Infatti "Examine fork", esamina la fork(etta), avrebbe fornito come risposta che era una forchetta da tavola, adatta a pugnalare eremiti! Perciò prendi la forchetta e poi vai, a superare il puzzle dell'eremita.
Il fantasma che torna a spaventare il suo assassino non ha bisogno di spiegazioni, ma se, per la fine della storia, non scopriamo perché un fantasma cammina su e giù nel bel mezzo di un parco giochi abbandonato, ogni terzo sabato del mese, suonando una tromba … beh, probabilmente ci sentiamo ingannati dall'autore.

Oggetti fuori contesto

>guarda
E' una cucina pulita e ben fornita. Sul tavolo c'è una motosega.
L'oggetto fuori contesto indica che l'indovinello ha preso il sopravvento sul mantenimento di un'atmosfera coerente. E' chiaro che, nell'esempio, abbiamo bisogno della motosega e l'autore l'ha messa in un posto dove possiamo trovarla. Questo può andare bene per il gioco ma male per il racconto. Nel mondo reale le cose si trovano dove si suppone siano e se non ci sono ci deve essere un motivo. Allora la motosega starebbe meglio nel capanno degli attrezzi.
Però ora vogliamo essere creativi e giustificare la presenza della motosega in cucina (e che mi invento ora? ho scritto l'ultima frase senza riflettere!). Pensa, pensa …
E' una cucina ordinata e ben fornita. Sul tavolo vedi la colazione: sei uova fritte, una pila di pancake alta mezzo metro e circa mezzo chilo di bacon fritto! Tutto magnifico per il colesterolo. Una enorme camicia di flanella è drappeggiata sulla sedia e, all'altro lato del tavolo, c'è una motosega.
Ora (forse) la motosega ha un senso in cucina. Dalla camicia di flanella e dalla quantità di cibo possiamo dire che un taglialegna si è appena allontanato dal tavolo, prima di mangiare la sua colazione e la motosega è la sua.
In effetti, mettere gli oggetti nel loro contesto aggiunge pepe al gioco, suggerisce ostacoli realistici per arrivare all'oggetto. In questo esempio potremmo mettere un limite di tempo per poter prendere la motosega e andarcene prima che il

taglialegna ritorni. Potremmo immaginare che non sarebbe troppo felice di vederci andar via con la sua proprietà! Ora, perché il taglialegna stesse mangiando la colazione proprio in quella cucina e perché sia stato chiamato ad allontanarsi … beh, un buon lavoro di fiction risponderà a queste domando, a tempo e luogo. Le risposte non devono essere profonde ma devono avere un senso. Ad esempio: Un enorme, burbero uomo con la barba entra con passi pesanti, asciugandosi le mani con un asciugamano di carta. Questa risposta ci farebbe capire che il taglialegna è stato in bagno, dandoci la motivazione che ci serviva.

Nascondere "serrature e chiavi" come oggetti del mondo reale può contribuire in maniera superficiale al realismo dell'avventura, alla sua atmosfera. Ma una volta che il giocatore intuisce cosa sta succedendo, l'artificio della mappatura uno a uno tra oggetti e problemi diventa stridente. Graham Nelson, quello di Inform, ha identificato questa cosa come la sindrome del prendi-X-usa-X. Dai alla capra una lattina di alluminio e lei sputerà un fazzoletto rosso. Avvolgiti il fazzoletto intorno alla testa e i gitani ti faranno entrare nella caverna. Usa la lanterna che hai trovato nella caverna per oltrepassare la talpa gigante e così via. Queste soluzioni serratura-chiave non rendono giustizia al complesso processo di soluzione dei problemi tipico del mondo reale e dopo un po' diventano noiose.
Per fortuna ci sono molti rimedi strutturali alla prevedibilità del giochino serratura-chiave. Vediamone quattro:

Soluzioni che richiedono più di un oggetto

Non è una novità che un problema potrebbe richiedere più di un oggetto per essere risolto. Adventure e Zork avevano un paio di problemi multi oggetto (ricordo l'orso incatenato e l'esorcismo all'Inferno) e, in generale, questi problemi rendono il gioco più interessante e realistico. Tuttavia anche un puzzle multi oggetto può sembrare artificiale. Immaginate la caccia ai vari componenti di un oggetto molto importante, magari derivato dal Signore degli anelli di Tolkien. Mi riferisco ai nove anelli del potere: raccoglili tutti e dominerai il mondo!
D'altronde la ricerca di parti è integrata in tantissimi scenari, racconti e pure episodi di serie TV. Perché l'IF dovrebbe o potrebbe starne lontana? Ovviamente se l'autore non rende i singoli pezzi degli oggetti interessanti di per se stessi e plausibilmente integrati nella storia, può portare il lettore/giocatore ad annoiarsi.

Oggetti rilevanti per più di un problema/soluzione

Oggetti con più di uno scopo ebbero il loro uso fin dall'inizio, mi riferisco addirittura a Adventure/Colossal Cave. Se ricordo bene, il secondo uso delle chiavi in quel gioco venne fuori in

un momento in cui ero arrivato al principio *un oggetto – un utilizzo* quasi per induzione. Fino a quel momento aveva funzionato così ed io avevo preso confidenza nel lasciare gli oggetti una volta utilizzati. Che sconforto dover tornare indietro, in superficie, per riprendere le chiavi! Oggi è norma comune che molti oggetti abbiano più di un utilizzo. In generale, il realismo aumenta; il giocatore deve abbandonare il meccanismo confortevole dell'un oggetto – un indovinello che, tra l'altro, non ha grosse somiglianze con il problem solving del mondo reale. Tuttavia bisogna stare attenti a non permettere di portare praticamente tutto con se. Il bizzarro risultato sarebbe quello di un avventuriero-sherpa sovraccarico di oggetti che "magari potrebbero essere utili". Ricordate che AWS vi permette dei limiti sul peso e sulle dimensioni. Usateli con sapienza.

Problemi con più di una soluzione

Per me, la differenza cruciale tra un indovinello ed un problema del mondo reale è che il problema reale ha più di una soluzione possibile. Questo è vero anche nel semplice problema di far cadere una banana da un soffitto di tre metri avendo solo una sedia ed un'asta di un metro. Gli scimpanzé sono capaci di stare in piedi su una sedia e di colpire la banana con l'asta. Perciò:"Stai in piedi sulla sedia" e "Colpisci la banana con l'asta" dimostrano che non è solo l'Homo Sapiens ad essere capace di utilizzare uno strumento. D'altronde potreste anche dire:
"Tira la sedia alla banana"
"Metti la sedia sull'asta e colpisci la banana"
"Tieni l'asta e salta verso la banana"
dimostrando grande creatività. Potreste anche dire:
"Bussa alla porta. Chiama lo sperimentatore. Minaccia lo sperimentatore di una causa legale. Sperimentatore, prendi la banana". Questa è tremenda, lo so, difficile anche solo da immaginare. Ma creativa al massimo grado!
A parte gli scherzi, pochi indovinelli in un gioco di IF presentano questa varietà di soluzioni. Il programmatore/scrittore preferisce implementare una varietà molteplice di locazioni piuttosto che aumentare il numero di soluzioni ad un problema come quello della banana.
E i giocatori vogliono affrontare una varietà di sfide e, per questo, sono disposti ad accettare alcune restrizioni come quella di meno soluzioni, specie se le soluzioni alternative sono meno ovvie di quella prevista. Però un indovinello che ignora l'ovvio e il ragionevole, grida "E' un gioco, non è una storia, non è IF!". Per una sorta di convenzione non scritta, alcune soluzioni crude sono da escludere. Mi riferisco al rompere le cose, bruciarle, colpire le altre creature o ucciderle. E difatti i messaggi standard di risposta a questo tipo di azioni in Inform e TADS fanno capire che il protagonista

non è un tipo alla Rambo ma piuttosto alla Hercule Poirot. Ovviamente nulla vi vieta di utilizzare una soluzione non convenzionale, plausibile però, nel risolvere un problema. Ad esempio, nell'avventura Christminster, c'è il problema di far si che un uomo, che dorme su una chiave, si volti al punto di permetterci di prendere la chiave. La soluzione prevede l'uso di una piuma per solleticarlo. Ha un senso come problema ma, nel mondo reale, forse avremmo usato le dita, no? Insomma, dobbiamo essere creativi ma nel modo previsto dal programmatore. Per evitare questo problema di lettura della mente, ci sono tre opzioni:

1. permettere soluzioni alternative;
2. far si che la soluzione alternativa si riveli sbagliata anche se lì per lì funziona (solleticare il tizio con le dita è una stimolazione troppo forte. L'uomo si sveglia e vi coglie sul fatto a rubargli la chiave);
3. programmare in un modo plausibile, che guidi il giocatore verso un'alternativa mediante l'uso di messaggi mirati (esempio: toccare un uomo con le tue mani? Sarebbe quantomeno sconveniente!).

Di queste, la più interessante è la seconda perché da al giocatore una spintarella verso la giusta direzione ma al contempo permette all'autore di mantenere il controllo sulla struttura del problema. In tutta onestà, il giocatore dovrebbe poter capire in anticipo che la soluzione alternativa non è la migliore. Oppure dovrebbe avere la possibilità di rifarla nel modo corretto. Un buon esempio di alternativa "sbagliata" potrebbe essere quello di far mangiare ad un suino affamato una rara collana di perle (che servirà in seguito) quando bastava dargli delle ghiande!

Oggetti irrilevanti per i problemi e problemi senza soluzione

Un giocatore che è interessato solo al gioco tende a vedere oggetti irrilevanti e problemi insolubili come delle noie, delle tracce sbagliate ad arte per portarlo fuori strada, in barba alla (sua del giocatore) regola che ogni cosa è rilevante e che il compito è quello di trovare cosa è rilevante per quale problema. Siccome inserire un mucchio di oggetti e locazioni inutili richiede del lavoro, di solito gli scrittori si adeguano. Ma se vediamo il gioco come ben più di una collezione di problemi, allora una peculiarità come un oggetto o una locazione può non aver nulla a che fare con un problema ma contribuire all'atmosfera della storia.
Insomma, per sintetizzare, in un racconto ben scritto ogni oggetto ed ogni locazione ha il suo scopo anche se non è legato ad un problema.

Guardarsi bene in giro

Una delle azioni fondamentali in ogni avventura è GUARDA. È lecito aspettarsi che un giocatore guardi ogni oggetto, alla ricerca di informazioni o indizi e sarebbe fastidioso rispondere sempre "Non noto nulla di particolare".
Prevedete invece un'azione GUARDA per tutti, o quasi tutti, gli oggetti e utilizzate dei messaggi ben scritti. Esempio:
GUARDA LA CHIAVE
Non possiamo limitare la frase GUARDA LA CHIAVE ad un solo luogo, dato che la chiave può essere presa e portata in giro. Utilizziamo, perciò, le condizioni a bassa priorità, quelle valide ovunque. Scriviamo la condizione in linguaggio naturale, tanto ormai siete dei maestri in AWS e sapete che dovete mettere VERB, NOUN e così via.

```
IF GUARDA LA CHIAVE then mess "E' una Yale di sicurezza." WAIT
ENDIF
```

Perciò, se nel gioco dico di prendere la chiave, AWS mi risponde con il messaggio. Accidenti, ma lo fa anche se non ho la chiave! E già, perché non faccio verificare se ho la chiave con me o se essa è presente nel luogo in cui mi trovo. Perciò:

```
IF GUARDA LA CHIAVE AND CARR CHIAVE then mess "E' una Yale di
sicurezza." WAIT ENDIF
IF GUARDA LA CHIAVE AND HERE CHIAVE then mess "E' una Yale di
sicurezza." WAIT ENDIF
```

La prima condizione verifica se ho la chiave con me. Se ce l'ho, stampa il messaggio ed esce (WAIT).
Se non ce l'ho, passa alla seconda condizione che verifica se la chiave è presente nel luogo corrente. Se c'è, stampa il messaggio ed esce.
Perciò, provando senza la chiave:
Cosa devo fare ? GUARDA LA CHIAVE
Qui non ne vedo.
mentre, con la chiave presa o visibile:
Cosa devo fare ? GUARDA LA CHIAVE
E' una Yale di sicurezza.

Perché questo controllo di possesso o presenza non è fatto automaticamente per tutte le frasi che coinvolgono un oggetto? Perché AWS vi consente così la massima libertà di espressione. Ad una frase come CERCA LA CHIAVE, si può rispondere "Cercatela da te" anche se non è presente, tanto per fare un esempio.
Vi consente anche di usare la decisione AVAI per verificare se un oggetto è disponibile (presente o trasportato).

```
IF GUARDA LA CHIAVE AND AVAI CHIAVE then mess "E' una Yale di
sicurezza." WAIT ENDIF
```

Questa condizione sostituisce le due di cui sopra.

Alcuni esempi avanzati di utilizzo di AWS:

La caccia

Un oggetto si avvicina al giocatore e, quando s'incontrano, succede qualcosa.
Immaginiamo che RI sia la locazione iniziale dell'oggetto e che questo oggetto si muova in maniera casuale ma sempre verso il giocatore. Codifichiamolo prima in simil-basic, per capire il concetto e poi in AWS.
RI = locazione iniziale dell'oggetto
Q = numero casuale da 1 a 6 oppure da 1 a 10, a seconda delle direzioni possibili nell'avventura.
NM = direzione possibile dell'oggetto in direzione Q, cioè CONN RI Q
Se NM = 0 allora non si fa nulla, visto che non si ha una direzione verso cui muovere.
Se NM = PL, locazione del player allora l'oggetto ha preso il giocatore
Se PL<RI and NM<RI allora RI = NM cioè se mi avvicino, visto che il giocatore è in una locazione il cui numero è minore di quella dell'oggetto e la nuova locazione è minore di quella attuale dell'oggetto, allora muovo: RI = NM.
Se PL>RI and NM>RI allora RI = NM cioè se mi avvicino, visto che il giocatore è in una locazione il cui numero è maggiore di quella dell'oggetto e la nuova locazione è maggiore di quella attuale dell'oggetto, allora muovo: RI = NM.
In tutti gli altri casi non faccio nulla, visto che l'oggetto si allontanerebbe dal giocatore
Come si fa in AWS:

Condizioni ad alta priorità

IF RES? 1 THEN SET 1 TO oggetto RI ENDIF
all'inizio pone l'oggetto nella sua locazione iniziale. Una volta posizionato l'oggetto non verrà più rimesso lì, per cui una volta sparito, fine della caccia.

IF THEN CSET 10 RAND 6 RESE 20 ENDIF
estrae un numero casuale tra 1 e 6 e lo pone nel contatore 10, resetta il marker 20. Se le direzioni possibili sono 10 allora si cambia il 6 con 10. Il contatore 10 contiene quindi una direzione da far prendere, se possibile e conveniente, all'oggetto.

IF HERE oggetto THEN t'ho preso ...
se il giocatore e l'oggetto sono nella stessa stanza allora...

IF CONNGT OBJLOC oggetto CTR 10 0
se dalla stanza dell'oggetto si può andare verso la direzione contenuta nel contatore 10

```
THEN
SET 20              setta il marker 20, che indica che si può
muovere l'oggetto
CSET 11 ROOM        pone nel contatore 11 la locazione corrente
CSET 12 CONN OBJLOC oggetto CTR 10          pone NM nel
contatore 12
SUBC 11 OBJLOC oggetto   sottrae al contatore 11 la locazione
dell'oggetto (PL-RI)
SUBC 12 OBJLOC oggetto   sottrae al contatore 12 la locazione
dell'oggetto (NM-RI)
ENDIF

IF SET? 20 AND CTRLT 11 0 AND CTRLT 12 0 THEN TO oggetto CONN
OBJLOC oggetto CTR 10
Se posso muovere l'oggetto e mi avvicino allora muovo l'oggetto
nella locazione indicata dalla connessione della stanza in cui
si trova l'oggetto verso la direzione contenuta nel contatore 10

IF SET? 20 AND CTRGT 11 0 AND CTRGT 12 0 THEN TO oggetto CONN
OBJLOC oggetto CTR 10
Se posso muovere l'oggetto e mi avvicino allora muovo l'oggetto
nella locazione indicata dalla connessione della stanza in cui
si trova l'oggetto verso la direzione contenuta nel contatore 10
```

La fuga

```
E' l'esatto contrario della caccia, per cui, cambiando i segni
delle operazioni di confronto, si ha:
RI = locazione iniziale dell'oggetto
Q = numero casuale da 1 a 6 oppure da 1 a 10, a seconda delle
direzioni possibili nell'avventura.
NM = direzioni possibili dell'oggetto in direzione Q, cioè CONN
RI Q
Se NM = 0 allora non si fa nulla, visto che non si ha una
direzione verso cui muovere.
Se NM = PL, locazione del player allora il giocatore ha preso
l'oggetto
Se PL<RI and NM>RI allora RI=NM cioè se mi allontano, visto che
il giocatore è in una locazione il cui numero è minore di quella
dell'oggetto e la nuova locazione è maggiore di quella attuale
dell'oggetto, allora muovo: RI=NM.
Se PL>RI and NM<RI allora RI=NM cioè se mi allontano, visto che
il giocatore è in una locazione il cui numero è maggiore di
quella dell'oggetto e la nuova locazione è minore di quella
attuale dell'oggetto, allora muovo: RI=NM.
In tutti gli altri casi non faccio nulla, visto che l'oggetto si
avvicinerebbe da solo al giocatore. Praticamente gli IF del
codice della caccia diventano:
IF SET? 20 AND CTRLT 11 0 AND CTRGT 12 0 THEN …
IF SET? 20 AND CTRGT 11 0 AND CTRLT 12 0 THEN …
```

Il cane da guardia

Un oggetto, ad esempio un cane, si muove su un percorso tra due o più locazioni consecutive. La probabilità per il giocatore di evitarlo dipende quindi da:
- l'esistenza di un percorso alternativo
- quante stanze l'oggetto percorre

Codificando in simil-basic, come al solito:

```
RI=locazione dell'oggetto
Q=numero casuale tra 1 e 6 ....
NM=CONN RI Q
Se NM=0 allora non muovo
Se NM<prima locazione OR NM>ultima locazione allora non muovo
RI=NM
Se RI=locazione corrente allora l'oggetto ha preso il giocatore
```

Sulla mappa:

```
                42----------43
                 |          |
     40----------41----------44----------45
```

Cioè il nostro ipotetico cane si muoverà in maniera casuale tra le locazioni 40 e 45.

Potremmo anche avere:

```
     40----------41----------42----------43
```

Allora il numero casuale Q può essere tra 3 e 4 (est o ovest) ed il cane si muove in orizzontale. Analogamente se volessimo farlo muovere in verticale.

Ultimo caso è quello di un oggetto (cane, mummia, quello che volete voi) che si muova lungo un percorso fisso tra due locazioni. Cioè

```
     40----------41----------42----------43
```

Da 40 a 43 e ritorno, incessantemente. Allora, codificando direttamente in AWS:

```
IF IN oggetto 40 THEN CSET 10 1 ENDIF
IF IN oggetto 43 THEN CSET 10 2 ENDIF
IF CTREQ 10 1 THEN CSET 11 OBJLOC oggetto ADDC 11 1 TO oggetto
CTR 11 ENDIF
IF CRTEQ 10 2 THEN CSET 11 OBJLOC oggetto SUBC 11 1 TO oggetto
CTR 11 ENDIF
```

Utilizzo avanzato dei comandi

LOOK: è buona norma usarlo dopo che si è scoperto un oggetto con un esamina o qualche altra azione. Non richiedete al giocatore di scrivere GUARDA, fatelo voi per lui. Ovviamente se comunicate il ritrovamento con un messaggio in cui c'è il nome dell'oggetto, quanto sopra non è necessario.

DESC/Esamina: se volessimo fornire un esamina per ogni oggetto, potremmo fare così:
IF VERB esamina AND NO1GT xx AND NO1LT yy THEN mess NO1 WAIT ENDIF
Qui dobbiamo stare attenti a non usare per nomi ed oggetti, gli stessi numeri delle locazioni.
Potremmo addirittura usare le descrizioni delle locazioni al posto dei messaggi. Il comando DESC ci viene in aiuto. L'idea è di avere una locazione con la descrizione dell'oggetto. A questo punto, con:
IF NO1GT xx AND NO1LT yy AND VERB esamina THEN DESC NO1 WAIT ENDIF
descriviamo il singolo oggetto.
E' ovvio che bisogna stare attenti ai numeri delle locazioni, degli oggetti e altro. Diventa opportuno utilizzare, per le locazioni, i numeri da 200 (o 300) in su. In fondo non c'è alcun obbligo di partire da 1. Gli oggetti invece partirebbero da 1.

IF NO1GT xx AND NO1LT yy verifica che il nome si riferisca ad oggetti nei limiti degli oggetti esaminabili.
Perciò, se esaminate l'oggetto **i**, la stanza **i** verrà descritta. Ovviamente attenzione ai numeri delle stanze "tradizionali", quelle non usate per descrivere gli oggetti.

SWAP: i due oggetti da scambiare devono avere lo stesso peso per non incorrere in problemi. Al limite potete lasciare (DROP) l'oggetto portato, scambiarlo con SWAP e poi riprenderlo con GET.

OBJ: può essere usato per aggiungere del sale all'avventura, ad esempio: un ratto ha rubato l'oggetto OBJ RAND 20. Così il ratto ruba un oggetto casuale tra 1 e 20. Ovviamente attenzione alla descrizione dell'oggetto. Di solito si scrive qualcosa come *un dado*. Leggere *un ratto ha rubato l'oggetto un dado* è un po' brutto. Attenzione alla grammatica!

LIST: si può usare per creare zaini o contenitori di vario genere. Guardate l'apposito capitolo.

TO: spostare oggetti in possesso del giocatore può creare problemi come SWAP. Stessa soluzione.

CONNCORR: si può usare per personalizzare i messaggi di "Non puoi andare da quella parte.". In pseudo linguaggio:
IF VERB direzione AND CONNCORREQ x 0 THEN MESS Uscita bloccata! WAIT ENDIF
Dove x è il valore corrispondente al verbo di direzione (nord =1, sud=2, …).
Esempio: mi trovo in una locazione qualsiasi e voglio personalizzare il messaggio di uscita verso le direzioni nord e sud (1 e 2, ricordate la tabella?). Ecco cosa inserisco nelle condizioni a bassa priorità:
IF VERB (nord) AND CONNCORREQ 1 0 THEN MESS Uscita bloccata! WAIT ENDIF
IF VERB (sud) AND CONNCORREQ 2 0 THEN MESS Non c'è uscita da quella parte! WAIT ENDIF
Ricordate, infatti, che nelle condizioni a bassa priorità vanno inserite le condizioni per gli spostamenti e nel file di base Initfi30.aws ci sono le sei condizioni per le direzioni principali più alto e basso. Sostituendole con quanto esemplificato sopra, abbiamo una personalizzazione dei messaggi. Una ulteriore personalizzazione può essere la scelta casuale tra n messaggi diversi, usando RAND. Vedi, più in la, la stampa di messaggi scelti casualmente tra m1 ed m2.

STRE: si può alterare nel corso dell'avventura, grazie a pozioni o simili. Penso anche all'aumento di forza dovuto ad un esoscheletro in un'avventura di fantascienza.

Utilizzo avanzato delle condizioni

ADVE: frasi come "dai le sigarette (alla) ragazza" o "versa la birra (sul) totem" (da Lo Stregatto) come vanno implementate? Siccome odio le sigarette, analizzerò la seconda frase. "versa la birra sul totem" diventa:
IF VERB versa AND NOUN birra AND ADVE sul AND NO2EQ totem AND AVAI birra AND HERE totem THEN xxx WAIT / OKAY ENDIF
Che tradotto vuol dire: se ho scritto versa birra sul totem e ho (o è qui, AVAI) la birra e c'è il totem (HERE basta, di solito i totem sono grandi e non si trasportano) allora faccio ciò che è previsto e poi termino il comando (WAIT) oppure stampo "Ok." e poi termino il comando.

HERE: se gli ostacoli (mostri, serpenti, scheletri …) sono rappresentati come oggetti, invece di utilizzare un marker si può usare HERE. Immaginiamo che il marker 10 significhi che il troll è morto se on. Noi, da una certa locazione vogliamo andare a nord ma possiamo solo se il troll è morto.
Dovremmo inserire un'analisi del marker 10 nelle condizioni per sapere se il troll è vivo o morto per poi poter stampare il messaggio specifico e andare o non andare a nord. Il metodo alternativo è quello di avere due oggetti, "Un grande, cattivissimo troll" e "Un troll morto". Poi, con questa azione, controlliamo sei il troll è vivo o morto:
IF VERB nord AND HERE troll THEN MESS Il troll ti blocca il passaggio. WAIT ENDIF
Subito dopo aggiungiamo:
IF VERB nord AND HERE troll morto THEN GOTO nord WAIT ENDIF
Ovviamente, come per le condizioni, non dobbiamo inserire nulla nella tavola delle connessioni altrimenti si andrà a nord indipendentemente dal troll.

IN x n: si può usare in modo analogo a HERE. Immaginiamo un negozio: tutti gli oggetti in vendita possono essere messi in una stanza inaccessibile, diciamo la 200. Invece di usare i marker per sapere se un oggetto è in vendita o è stato venduto, controlliamo se IN oggetto 200. Se l'oggetto è in 200 è in vendita e il giocatore può comprarlo.

SET? e **RES?:** si usano per controllare lo stato dei marker. I marker si usano per qualcosa che ha due possibili stati, ad esempio un forziere aperto o chiuso, un bottone premuto o sollevato e così via. Gli oggetti fisici, come porte, torce accese o spente, troll e altro si possono gestire anche con HERE. Ci sono però situazioni in cui i marker sono indispensabili, tipo quando dovete determinare se avete inizializzato il contatore della sequenza di auto distruzione oppure se siete sotto l'effetto di un incantesimo di invisibilità.

Avventure in più episodi

Potreste voler scrivere un'avventura in più parti, magari per non farne una troppo grande o per diversificare il tempo, tipo un viaggio nel tempo o nello spazio. Ci sono alcuni metodi utili. Il primo è semplicemente quello di avere la seconda parte indipendente dalla prima e giocabile anche da sola. Il secondo metodo è quello di inserire una password alla fine della prima parte, password che da accesso alla seconda parte. Il terzo metodo è quello di avere un salvataggio del gioco alla fine della prima parte, salvataggio che, caricato nella seconda parte, la inizializza opportunamente. Ricordate che nel salvataggio ci sono i contatori e i marker, tra le altre cose, perciò all'inizio della seconda avventura basta verificare alcuni marker e contatori per sapere se il giocatore ha risolto la prima parte.

Miscellanea

Per sapere se un oggetto non è da nessuna parte, cioè è situato nel limbo:
IF OBJLOCEQ oggetto 0 THEN xxxxx ENDIF
Es: se la torcia (oggetto 1) non è da nessuna parte IF OBJLOCEQ 1 0 THEN xxxxx ENDIF

Per sapere se un oggetto è da qualche parte:
IF OBJLOCGT oggetto 0 THEN xxxxx ENDIF
Es: se la torcia è da qualche parte IF OBJLOCGT 1 0 THEN xxxxx ENDIF

Per sapere se due oggetti sono nella stessa stanza (che non è quella corrente):
IF OBJLOCEQ oggetto1 OBJLOC oggetto2 THEN xxxxx ENDIF
Es: se il cane (oggetto 2) e il gatto (oggetto 3) sono nella stessa stanza

```
IF OBJLOCEQ 2 OBJLOC 3 THEN xxxxx ENDIF
```

Per stampare un messaggio casuale compreso tra m1 ed m2:
IF THEN CSET 1 RAND (m2-m1+1) ADDC 1 (m1-1) MESS CTR 1
Es: se i messaggi sono:

```
4    Come va?
5    Come stai?
6    Tutto bene?
7    Ti senti a posto?
8    Come te la passi?
```

avremo m1=4, m2=8 per cui (m2-m1+1)=5 e (m1-1)=3. Allora:
CSET 1 RAND 5 mette nel contatore 1 un numero casuale tra 1 e 5
ADDC 1 3 somma al contatore 1 l'offset iniziale, quindi nel contatore 1 ci sarà un numero che ha un valore tra 4 e 8 (3 + il numero casuale tra 1 e 5)
MESS CTR 1 stampa il messaggio il cui numero è nel contatore 1.
Questa cosa può essere utilizzata per stampare un prompt diverso e rendere l'avventura un po' più varia. Il prompt, si può cambiare in maniera casuale ad ogni turno scegliendolo fra una serie preparata di messaggi. Es:

```
100  Cosa devo fare?
101  Cosa vuoi che faccia?
102  Ora?
103  Dimmi:
```

Si toglie il messaggio 1012 dalla lista dei messaggi e come ultima linea delle condizioni hi (che quindi verrà eseguita appena prima di chiedere l'input al giocatore) si pone:
IF THEN CSET 1 RAND 4 ADDC 1 99 MESS CTR 1 ENDIF

SENDALLROOM può servire per togliere tutti gli oggetti al giocatore nel caso ad esempio un'operazione di teletrasporto possa trasportare solo lui.

ISCARRSOME, ISCARRNOTH può servire per impedire al giocatore di effettuare un'azione (es: attraversare un ponte) se in possesso anche di un solo oggetto.

Un paio di comodità

PRESSKEY attende che il giocatore prema un tasto. Magari si può far precedere da un messaggio tipo "Premi un tasto per continuare". Può essere utile per aggiungere un elemento di suspense alla vicenda.
HOLD t mette in pausa l'avventura per il tempo di t secondi. Le pause di attesa possono aggiungere suspense e coinvolgere il giocatore, ma possono anche annoiarlo se sono usate spesso o a sproposito.
In particolare, non mettete mai pause nell'introduzione, o quando volete solo dare il tempo di leggere qualcosa. Usate invece il PRESSKEY.

Una bomba ad orologeria: l'uso del tempo in AWS

Che IF sarebbe se non potessimo aumentare la suspense con il fattore tempo? Immaginate una storia di spie ed una bomba ad orologeria. Dovremmo sfruttare il passar del tempo per aumentare la tensione, magari mentre un artificiere tenta di disinnescarla. AWS permette di inserire il fattore tempo, facendo accadere un dato evento in un momento prefissato od in rapporto ad una o più azioni compiute dal giocatore.
Vediamo, come primo esempio, quanto fatto ne Il conte di Montecristo.

Nelle condizioni hi abbiamo inserito
IF RES? 4 THEN SET 4 CSET 5 30 ENDIF
Poiché inizialmente tutti i marker sono off ed i contatori a 0, RES? 4 è vera (il marker 4 è off). Assegniamo il valore 30 al contatore 5, quello che se è a 0 fa intervenire i gendarmi, e poi poniamo ad on il marker 4. In questo modo questa linea verrà eseguita una sola volta all'inizio.

IF THEN DECR 5 ENDIF
(ad ogni ciclo di gioco decrementa di uno il contatore 5)

IF CTREQ 5 0 THEN MESS 10 HOLD 4 RESTART ENDIF
Se il contatore 5 è arrivato a 0 allora arrivano i gendarmi ed è finita. Lo segnaliamo stampando il messaggio 10 (I gendarmi hanno scoperto il mio tentativo di fuga. La mia resistenza è vana. Mi portano via e non credo sarà più possibile per me salvarmi da questo destino. Addio.). Inoltre, visto che abbiamo perso, attendiamo quattro secondi (HOLD 4) e poi facciamo ripartire il gioco (RESTART).

Ovviamente non siamo obbligati a far partire il conto alla rovescia dall'inizio del gioco. Potrebbe partire da una certa situazione, che so, che l'artificiere ha tagliato il filo sbagliato dell'innesco della bomba. Come si fa? Si usa un marker che, normalmente a 0, viene posto a 1 quando si crea la situazione (si taglia il filo). Vediamolo in linguaggio (quasi) naturale. Vi lascio l'implementazione in AWS come esercizio.
IF TAGLIA IL FILO ROSSO THEN SET 10 CSET 7 15 WAIT ENDIF
al taglio del filo rosso, quello sbagliato, si imposta il marker 10 e si mette il numero 15 nel contatore 7, quello che useremo per contare i turni prima che la bomba esploda. In questa condizione mancano, ovviamente, i controlli sulla presenza della bomba e così via.
Nelle condizioni ad alta priorità inseriremo una condizione del tipo
IF SET? 10 THEN DECR 7 ENDIF
In questo modo decrementiamo il contatore (se il marker 10 è impostato, cioè se siamo nella situazione di pericolo).
Inseriremo anche una condizione come
IF CTREQ 7 0 THEN MESS "Boom! Il tuo tentativo (piuttosto maldestro, in verità) di disinnescare la bomba è fallito. Con

l'esplosione hai aperto un cratere enorme che sarà la tua tomba." RESTART ENDIF
che appena il contatore 7 arriva a 0 (la bomba esplode), stampa il messaggio e fa ripartire il gioco.

Per evitarvi di trafficare con marker e contatori vari, ricordate che i contatori 126, 127 e 128 vengono auto decrementati ad ogni turno di gioco da AWS. Perciò basta impostarli all'inizio e loro scenderanno verso 0 da soli, un'unità alla volta. Questo semplifica la scrittura delle condizioni. Come? Queste sono le tre condizioni del Conte di Montecristo:

```
IF RES? 4 THEN SET 4 CSET 5 30 ENDIF
IF THEN DECR 5 ENDIF
IF CTREQ 5 0 THEN MESS 10 HOLD 4 RESTART ENDIF
```

Se invece del contatore 5 usiamo il 126, le tre condizioni diventano:

```
IF RES? 4 THEN SET 4 CSET 126 30 ENDIF
IF CTREQ 126 0 THEN MESS 10 HOLD 4 RESTART ENDIF
```

Ho risparmiato la scrittura di una condizione. Meno condizioni da scrivere, meno errori.

Ora vediamo come abbiamo fatto un conto alla rovescia in Pattuglia all'alba, usando il contatore 126.

Nelle Condizioni ad alta priorità:

```
IF res? 16 and ctreq 126 0 then cset 126 61 ENDIF
IF res? 16 and ctreq 126 10 then mess 33 ENDIF
IF res? 16 and ctreq 126 1 then mess 34 quit ENDIF
```

Osservate come abbiamo a disposizione 61 turni di gioco per risolvere l'avventura. Il numero 61 viene da alcune prove fatte, è un numero non troppo basso e non troppo alto. Abbiamo perciò impostato il contatore 126 a 61. Tale impostazione è eseguita solo una volta, all'inizio, perché solo finché non gli si cambia valore il contatore 126 vale 0. Inoltre il marker 16 segnala se abbiamo vinto. Se non abbiamo vinto vuol dire che stiamo ancora giocando. Allora, ad ogni turno di gioco, il contatore 126 viene decrementato automaticamente. Quando scende a 10 (seconda condizione), AWS stampa il messaggio 33:" Una voce forte rimbomba nelle vicinanze:Soldato Smith! Dove ti stai nascondendo rifiuto della razza umana?".

Se il contatore arriva ad 1 allora AWS stampa il messaggio di sconfitta:" Senti dei passi alle tue spalle ed una voce burbera grida:Eccoti! Gettatelo nelle segrete del castello per non essersi presentato in servizio!. Due soldati corpulenti ti scortano fino alle segrete e ti rinchiudono in una cella. Ci hai messo troppo ed hai fallito miseramente!".

Se invece hai vinto allora viene impostato il marker 16 e quindi il conto alla rovescia non continua.
Impostare un livello (o più di uno) intermedio nel conto alla rovescia, con messaggi adatti, aumenta l'atmosfera e la suspense della narrazione.
Immaginiamo di trovarci in una casa non nostra, magari siamo delle spie che devono rubare qualcosa prima che il legittimo proprietario ritorni. Una serie di messaggi "temporizzati" aiuta molto. Immaginatevi dentro questa casa a forzare una cassaforte e AWS vi dice che:
Sento un'auto che arriva.
Allora vi sbrigate a ruotare le manopole ma non riuscite. AWS vi informa che
Sento aprire una portiera.
Allora provate freneticamente a fare qualche altra cosa ma vi manca un oggetto, una chiave o cosa. AWS, implacabile, dice che:
Sento dei passi.
C'è suspense, vero?

In effetti il passaggio del tempo rende molto più interessante il gioco. Ma non il tempo che il giocatore passa a pensare quale azione compiere bensì il tempo inteso come numero di mosse compiute. Un esempio abusato è quello della torcia la cui batteria si scarica dopo un certo numero di mosse, lasciando l'avventuriero al buio se non cambia le batterie. Come dite? Ne ho abusato anch'io nella Piramide di Iunnuh? Ragazzi, l'ho scritta nel 1984. Allora non era abusare, era una novità! Studiatevi pure l'esempio della Piramide.

Ora affrontiamo un problema differente dal conto alla rovescia. Giochiamo con il contatore 125, quello che contiene il numero di turni. Se volessimo avere una situazione in cui, proprio all'inizio, il giocatore possa morire di fame se non mangia del cibo entro un certo numero di mosse, potremmo scrivere:

condizioni bassa priorità
IF (il giocatore mangia il cibo) THEN SET 5 (altre azioni) WAIT ENDIF
condizioni alta priorità
IF CTRGT 125 20 AND RES? 5 THEN (uccidi il giocatore) WAIT ENDIF
con ciò uccidendo il giocatore se non mangia entro le prime 20 mosse! Se mangia è segnalato dal marker 5, da noi scelto all'uopo.
Durante il gioco invece potremmo voler far morire di sete nel deserto il giocatore se non beve entro un certo numero di turni. Useremo allora il marker 5 per indicare che il giocatore ha bevuto, il marker 6 per indicare che è entrato nel deserto, il contatore 5 che conterrà il numero di mosse permesse senza bere, 10 nel nostro esempio.

Condizioni locali (al posto di una connessione nella tabella delle direzioni)
IF (entra nel deserto) THEN CSET 5 10 SET 6 GOTO (deserto) WAIT ENDIF

Condizioni a bassa priorità
IF (bevi acqua) THEN SET 5 (altre azioni) WAIT ENDIF
Hi
IF RES? 5 AND SET? 6 THEN DECR 5 ENDIF
IF CTREQ 5 0 AND SET? 6 THEN (uccidi il giocatore) ENDIF
In questa sequenza ogni turno nel deserto senza bere decrementa il contatore 5. Quando questo raggiunge 0 allora scatta la condizione hi che uccide il giocatore.
Potremmo ovviamente far si che se il giocatore esce dal deserto, il marker 6 si resetti e il giocatore non possa più morire di sete. Insomma, la vostra fantasia è il limite.

Vediamo altri due esempi di utilizzo di contatori quali la gestione di soldi o monete e il problema dell'orologio.

Mettiamo in un contatore, diciamo il 100, la somma posseduta dal nostro eroe.
Se incassa moneta:
IF (incassa) THEN ADDC 100 (somma) WAIT ENDIF
Se spende o perde moneta
IF (spende/perde) THEN SUBC 100 (somma) WAIT ENDIF
Se vuole contare i soldi
IF (conta) THEN PRIN CTR 100 WAIT ENDIF
Ovviamente mancano i controlli per vedere se quando spende ha la somma da spendere ma, insomma, non sono poi così difficili.

Per l'orologio, invece, utilizziamo tre contatori, uno per i minuti, uno per le ore ed uno per i giorni (simulati). Diciamo che il contatore 10 conterrà i minuti, l'11 le ore ed il 12 i giorni.
Condizioni ad alta priorità
IF THEN INCR 10 ENDIF
Il contatore 10 viene incrementato ad ogni turno
IF CTREQ 10 60 THEN INCR 11 CSET 10 0 ENDIF
Il contatore 11 viene incrementato ogni 60 'minuti'. Il contatore 10 ritorna a 0.
IF CTREQ 11 24 THEN INCR 12 CSET 11 0 ENDIF
Il contatore 12 viene incrementato ogni 24 'ore'. Il contatore 11 ritorna a 0.
Se il giocatore chiede l'ora, ovvero il tempo trascorso:
Condizioni a bassa priorità
IF (chiede l'ora) THEN PRIN CTR 12 (giorni) PRIN CTR 11 (ore) PRINT CTR 10 (minuti) WAIT ENDIF
Ovviamente con gli opportuni messaggi per abbellire l'output.

Come esercizio provate a implementare questa situazione. Vi trovate in una villa, c'è una cassaforte e avete una bomba il cui detonatore è collegato ad una sveglia meccanica.
Avrete bisogno di questi oggetti:

- Una sveglia (di buona marca);
- Una solida cassaforte;
- Una cassaforte sventrata;
- Un cumulo di macerie.

La sveglia si trova, all'inizio, in camera da letto, e la cassaforte sta in cantina (non prendibile).
La cassaforte sventrata ed il cumulo di macerie sono, ovviamente, nel Limbo (locazione iniziale 0). Appariranno dopo.
La chiave è nel Limbo, apparirà solo una volta distrutta la cassaforte.
Usate un timer per la sveglia/bomba.
L'azione CARICA LA SVEGLIA farà partire il timer con un tempo abbastanza breve, diciamo 10 mosse.
L'azione GUARDA LA SVEGLIA ci dirà se è carica o meno.
Ad ogni turno il timer scrive un "tic, tac" se si ha la sveglia in mano, altrimenti è troppo lontano per sentirla.
Se il timer va a zero, la bomba esplode. Ci sono tre casi:

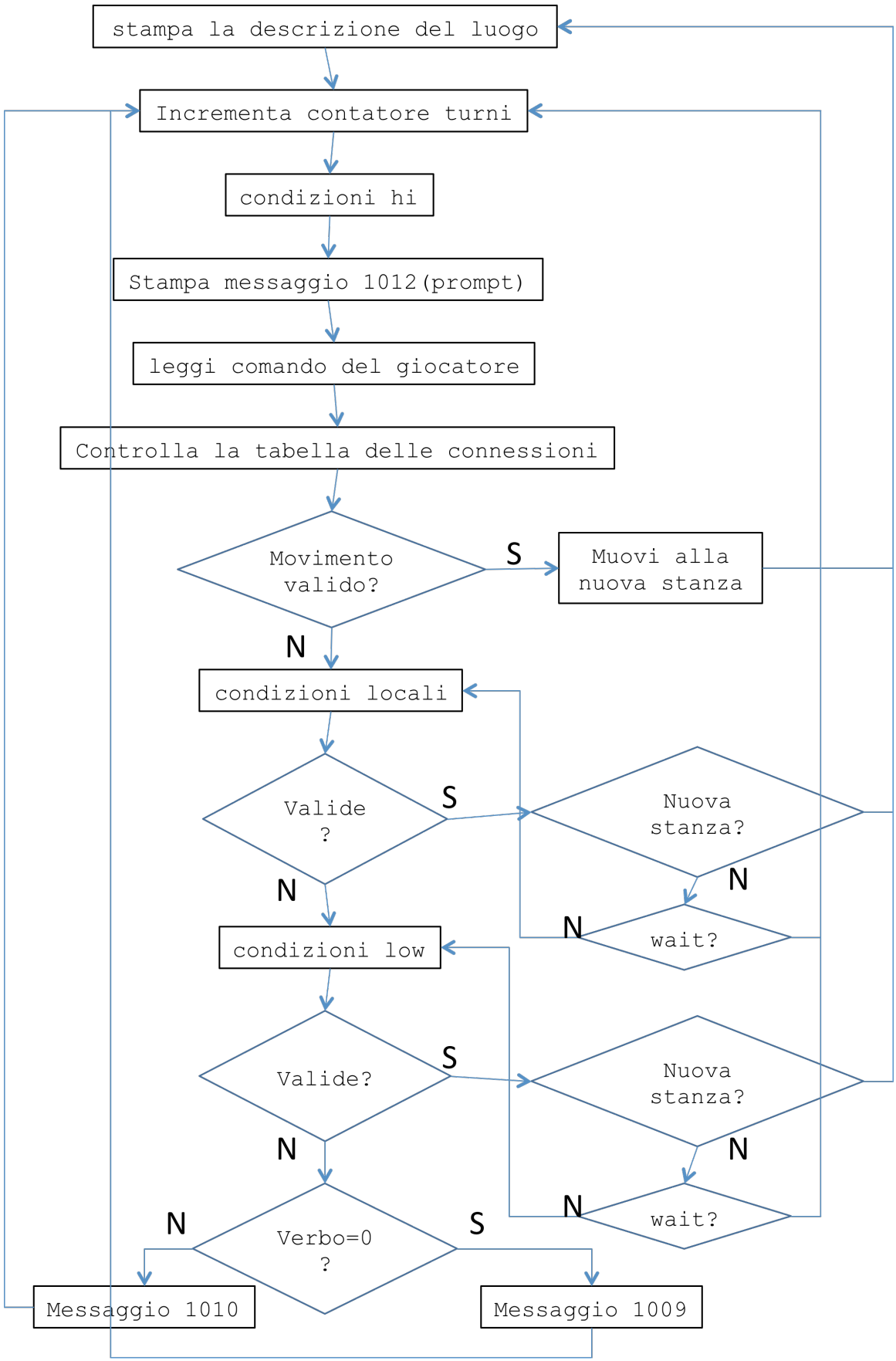
- È nello stesso luogo del giocatore, trasportata o presente. Le conseguenze sono ovvie, fine del gioco;
- È in cantina. Scompare la cassaforte ed appaiono cassaforte sventrata, macerie e chiave. Si potrebbe anche non far apparire la chiave e richiedere che il giocatore scriva GUARDA LA CASSAFORTE o ESAMINA LE MACERIE.
- È in un altro luogo. Appaiono solo le macerie (la chiave è nella cassaforte).

In ogni caso, quando la bomba scoppia, si sente di sicuro in tutta la villa. Perciò sono inutili i controlli di luogo. E la sveglia/bomba sparisce.

Dizionario esteso

A							
ABBANDONA	AFFERRA	APRI	ATTIVA	ACQUISTA	ARRAMPICATI	AFFETTA	ASCOLTA
ATTRAVERSA	ACCENDI	ATTACCA	AMMAZZA	ARRAMPICA	ABBRACCIA	ANNUSA	ASSAGGIA
ASSAPORA	AGITATI	ACCAREZZA	ASPETTA	ATTENDI	ALLACCIA	AGITA	ANNODA
B							
BLOCCA	BEVI	BRUCIA	BACIA				
C							
CHIUDI	CAMMINA	CORRI	CONTROLLA	CERCA	COLPISCI	COLMA	CANTA
COPRI	CHIEDI	COMBATTI	COMPRA	CONSULTA	CONGIUNGI		
D							
DESCRIVI	DISATTIVA	DÌ	DI	DICI	DOMANDA	DAI	DORMI
E							
ESAMINA X	ESCI	ENTRA					
F							
FAI	FAI INVENTARIO	FISSA					
G							
GUARDA G	GIRA	GRIDA	GUSTA				
I							
INVENTARIO I	INSERISCI	INDOSSA	IMPOSTA	INV			
L							
LOTTA	LASCIA	LANCIA	LEGGI	LUSTRA	LUCIDA	LEGA	
M							
METTITI	METTI	MANGIA	MOSTRA	MUOVI	MEDITA		
N							
NO	NUOTA						
O							
ON	OFFRI	OFF	ODORA				
P							
POSA	PAGA	PRESENTA	PREMI	PULISCI	PENSA	PALPA	PARLA
PRENDI	PELA	PREGA					
R							
RISVEGLIA	RIMUOVI	RACCOGLI	RUOTA	RISPONDI	ROMPI	RIEMPI	RISVEGLIATI
RICERCA	RIFLETTI	RIPULISCI					
S							
SBLOCCA	SIEDI	SIEDITI	SDRAIATI	SVUOTA	STA	STAI	SCENDI
SERRA	SPEGNI	SOFFIA	SCALA	SALI	SCOPRI	SCASSINA	SFONDA
STRAPPA	SPOSTA	SFRONDA	SFOLTISCI	SPACCA	SCAVA	SORSEGGIA	SPINGI
SPOLVERA	SCUSA	SCHIACCIA	SPIACCICA	SPREMI	STROFINA	SPIACENTE	SU
SONNECCHIA	SVEGLIATI	SVEGLIA	SI	SÌ	SENTI	SALTA	SCUOTI
SALUTA							
T							
TROVA	TOGLI	TRASPORTA	TRASFERISCI	TORTURA	TRANGUGIA	TAGLIA	TOCCA
TIRA	TRASCINA						
U							
UNISCI	UCCIDI						
V							
VEDERE	VAI	VA	VISITA	VEDI L			

Il ciclo di gioco nel dettaglio:



I tipi di file generati da AWS

AWS utilizza dei file per contenere i dati di un'avventura. Il file base ha estensione **.aws** e contiene, in chiaro, tutti i dati necessari al gioco. Se aprite tale file con un editor di testo (non modificatelo, se non sapete cosa state facendo), potrete vedere una serie di stringhe e numeri. Ci troverete una serie di sezioni con la loro intestazione:

AWS 'stringa identificativa del file
VERSIONE
3.0 'versione del file salvato, non coincide con la versione del software.
7 'colore testo
4 'colore sfondo
7 'colore del testo in caso di buio
0 ' colore dello sfondo in caso di buio
Lo stregatto 'titolo dell'avventura
Marco Vallarino 'autore
2014 'anno
Dummy 'note
129 'codice
Courier New ' nome del font
14 'dimensione del font
0 'stile del font (grassetto, corsivo …)
14 ' locazione iniziale
FALSE 'avventura grafica o meno
100 'massimo peso trasportabile
100 ' massima dimensione trasportabile

CONDIZIONIHI

CONDIZIONILOW

CONDIZIONILOCALI

DIZIONARIO

LOCAZIONI

MESSAGGI

OGGETTI

FINEDATI

Esistono poi altre tipologie di file. Quando salvate un file **.aws**, la sua versione esistente su disco non viene sovrascritta ma rinominata in **.awb**, che è il backup del file .aws di gioco.

In caso di problemi potete sempre rinominare il file .awb in .aws e recuperare la versione precedente.

Durante il gioco potete salvare la situazione su disco. In questo caso si creano file **.awp**.

Infine ci sono i file criptati, da giocare con il player o che vengono incorporati nell'eseguibile (il **.exe** di Windows). Tali file hanno estensione **.awc**.

Mentre del .aws vi ho dato la struttura, anche per permettervi di ricostruirlo a mano, se necessario, di questi file non vi dico nulla. I file .awb e .awp sono in chiaro perciò potete studiarli da soli con un editor di testi e certamente sarete in grado di capirne la struttura.

Il file .awc, invece, è quello che protegge il gioco da occhi indiscreti che potrebbero leggere i vari messaggi e condizioni e capire come risolvere il gioco. Tale file è quello che deve essere distribuito, un po' come un linguaggio di programmazione (e in effetti AWS è proprio questo, un linguaggio di programmazione): il sorgente (.aws) rimane in mano al programmatore, l'eseguibile/interpretabile (.awc) viene distribuito. Il player può caricare un .awc e permettere di giocarlo ma potete anche distribuire i giochi in forma di eseguibili con il codice awc incorporato al suo interno.

Il debug del codice AWS

In effetti non esiste un vero debugger integrato, vuoi perché il sistema è semplice, vuoi perché il flusso di programma è lineare, senza salti o cicli.
Le cose fondamentali, i testi, si correggono in maniera visuale, per il codice il discorso è leggermente diverso.
Dovete sempre tenere a mente il flusso di processo, quello che parte dalla stampa della descrizione ed attraversa le varie condizioni. E' qui che possono annidarsi gli errori. Poiché, però, il codice interessato è sempre di poche linee alla volta, basta capire qual è il comando male interpretato e risalire alle condizioni locali o quelle a bassa priorità dove potrebbe essere l'errore. Se invece sono le condizioni generali (dovrebbe essere buio e non lo è, il timer non funziona e così via) allora esaminate le condizioni ad alta priorità.
Tuttavia un aiutino serve e comunque fa comodo. Allora inserite nel dizionario la parola *room* come verbo con codice 1100 e poi implementate la seguente condizione locale:

```
IF VERB 1100 THEN PRIN ROOM WAIT ENDIF
```

In questo modo, quando digiterete *room*, AWS risponderà con il numero della locazione corrente.
Con questa logica, poiché non userete certo tutti i 128 marker o i 128 contatori, implementate azioni come:

```
IF VERB marker THEN PRIN mrk1 PRIN mrk2 PRIN mrk3 PRIN mrk4 PRIN mrk6 PRIN mrk7 WAIT ENDIF
```

I marker che inserirete saranno quelli a voi necessari.
Ovviamente stessa cosa per i contatori.
E' chiaro, poi, che potrete anche inserire dei messaggi di debug, che escono solo sotto certe condizioni. Insomma, il limite è la fantasia.

Ecco gli errori che l'interprete emette in caso di errore:

- Locazione fuori range: è stata richiesta una locazione al di fuori di 1 .. 1024
- Marker fuori range: è stato richiesto un marker al di fuori di 1 .. 128.
- Contatore fuori range: è stato richiesto un contatore al di fuori di 1 .. 128.

In questi casi sono stati oltrepassati i limiti di AWS

- IF atteso
- AND/OR atteso

Questi errori sono legati alla struttura della condizione AWS, che è composta da IF decisione AND/OR decisione …

- Azione non riconosciuta
- Decisione non riconosciuta

Nella condizione sono state usate azioni o decisioni non previste dal linguaggio.

- Errore in chkfrase
- Errore in frase
- Errore nell'analisi della frase
- Errore in decisione
- Non riesco ad analizzare la decisione
- Funzione scritta non correttamente
- Errore in FUNZIONE: numero atteso
- Errore: funzione attesa

Questi errori fanno riferimento a parti del codice di AWS (chkfrase, frase, funzione …) e puntano l'accento su cosa, nella condizione in errore, è stata la causa scatenante di detto errore.

- Direzione non riconosciuta in SETCONN
- CONN: valore fuori range
- Valore di direzione fuori range in CONNEQ
- Valore di direzione fuori range in CONNGT
- Valore di direzione fuori range in CONNLT
- CONNCORR: valore fuori range
- Valore di direzione fuori range in CONNCORREQ
- Valore di direzione fuori range in CONNCORRGT
- Valore di direzione fuori range in CONNCORRLT

In decisioni o azioni che implicano una direzione, è stato usato un valore al di fuori del range previsto 1 .. 8.

- CONN senza un argomento/due argomenti.
- CONNCORR senza argomento
- CTR senza argomento
- RAND senza argomento
- OBJLOC senza argomento
- WEIG senza argomento

Le azioni mancano di uno o più degli argomenti necessari.

- L'argomento di PROB deve essere tra 0 e 100

Autoesplicativa

Molti di questi errori sono evitabili se le condizioni le scrivete con l'ausilio dell'editor avanzato e le controllate prima di uscire mediante l'apposito pulsante.