Numero 6 – Giugno 2005

INDICE

Editoriale

di Roberto Grassi

Benvenuti al numero 6 di Terra d'IF.

Anche in questo numero ci sono articoli, recensioni e rubriche di buona qualità. L'interesse e l'attenzione verso il mondo delle avventure testuali sta lentamente salendo. Dopo mesi di silenzio sono stati rilasciati due nuovi giochi a distanza di due settimane e probabilmente qualcos'altro sta per arrivare.

Per quanto riguarda gli articoli segnalo in particolare il lungo articolo di Emily Short sulla caratterizzazione dei personaggi non giocanti, tradotto da Pink (che ringrazio in modo particolare per la sua disponibilità). A seguire, un articolo sulle vecchie avventure testuali per lo Spectrum ed un lungo articolo sul newsgroup delle avventure testuali e su come parteciparvi curato da Vincenzo Scarpa.

L'intervista di questo numero, curata da Andrea Rezzonico, è dedicata a Marco Vallarino, l'autore di *Enigma* e *Il giardino incantato*, le due AT più scaricate in assoluto. È un'intervista molto interessante che consiglio caldamente di leggere.

Nasce da questo numero una nuova rubrica che si chiama "Design Review", che si pone l'obiettivo di focalizzare l'attenzione sul design di un'AT attraverso l'analisi puntuale ed estesamente commentata dei giochi disponibili.

Come di consueto, il numero si chiude con le recensioni e con l'articolo di programmazione.

Caratterizzazione dei PNG

di Emily Short

Traduzione di Pink

Come la mia pagina sul simulazionismo, questa pagina ha lo scopo di fornire consigli basati sulla mia personale esperienza, con la menzione di specifici giochi ogni qualvolta possibile. Partiamo dagli aspetti che considero più facili e procediamo verso i più difficili.

Altre ottime risorse da cui prendere informazioni sul codice dei PNG è il tutorial Inform InfAct di Roger Firth, la collezione dei contributi alla libreria di Inform sui PNG e i consigli dell'Onyx Ring per l'implementazione della conversazione in Inform. Per chi usa TADS, c'è l'articolo sui PNG in T3 di Micheal J.Roberts (alcune parti del quale sono teoriche e potrebbero essere utili a tutti).

Inoltre, nel caso ve lo stiate chiedendo: questa pagina è cambiata notevolmente dalla sua introduzione. Ho esteso alcune sezioni per trattare di nuovi giochi e nuovi materiali e ne ho rimosso altre perché sono giunta alla conclusione che fossero troppo astruse (nel caso della sezione "Writing") o meritassero un trattamento più completo di quello che io posso offrire qui.

Sebbene a questo punto non riesca a stare dietro al suo contenuto, ho comunque mantenuto una vecchia versione dell'articolo come riferimento (vedi http://emshort.home.mindspring.com/NPC3.htm).

Lo scopo dei PNG

Prima di iniziare a programmare dovreste avere ben chiaro il genere di gioco che state scrivendo e lo scopo per il quale debbano esserci dei PNG. Un gioco con una forte enfasi sugli enigmi, dove i PNG sono presenti solo per fornire un altro genere di sfida, avrà un ben diverso trattamento rispetto ad un gioco lineare, orientato alla trama, dove l'interazione con i PNG è il primo scopo del gioco. Un mistero con molti enigmi basati sulla conoscenza dei vari elementi avrà ancora un altro tipo di requisiti, rispetto ad un romanzo dove l'interazio-

ne emozionale è enfatizzata.

Anche dentro lo stesso gioco i diversi PNG possono avere diversi ruoli nella prospettiva del gioco. Una delle migliori trattazioni di questo problema che io conosca è l'articolo sulla teoria Ask/tell di Jim Fisher: afferma di riguardare solo uno specifico sistema di conversazione ma in realtà tratta alcune importanti questioni sull'uso dei PNG in qualsiasi IF. Volete che il giocatore parli in maniera estensiva con i PNG avere un grande grado di flessibilità nel risultato? O preferite avere un controllo preciso su ogni conversazione? I PNG esistono principalmente per dare un po' di colore o hanno un'esposizione vitale per la IF? Devono compiere qualcosa di particolare nella storia?

Le risposte a queste domande influenzeranno il resto del design dei PNG. In un gioco con una trama altamente predeterminata (cioè nella quale ci si può sedere a scrivere una lista di tutte le scene della storia), un approccio simulazionistico nel design dei PNG risulterebbe sbagliato. E cioè avrete probabilmente poco bisogno di variabili per tenere traccia di emozioni e comportamenti in generale e otterrete di più programmandoli specificatamente. Per contro, in un gioco "molto ampio" dove non conta quanto tempo (poco o molto) che il giocatore passa con PNG, potreste ritrovarvi a scrivere un codice molto generico per trattare problemi come il carattere dei PNG, il loro comportamento e i loro obbiettivi.

Caratterizzazione "In Absentia"

Uno dei modi più semplici per presentare un personaggio forte è quello di evitare che il questo personaggio mostri la propria faccia in tutto il lavoro della fiction. Questa non è una novità neanche per gli autori di fiction statica. L'ombra di Long John Silver incombe in tutta *L'isola del Tesoro*; il personaggio eponimo in *Rebecca* di Du Maurier è agghiacciante e memorabile; l'*Odissea* comincia con quattro libri in cui persone parlano di Ulisse prima ancora che il personaggio faccia il suo ingresso nella scena.

Poiché il PNG è forse, tra tutti i brandelli di codice, quello più difficile da rendere convincente, un gruppo di oggetti "sostituti" che puntano alla personalità del personaggio spesso risultano più efficaci. *Nothing more, nothing less* di Gilles Duchesne è interamente pervaso dal personaggio di una donna assente; tutto quello che il protagonista osserva gli fa pensare a qualcosa relativo a lei.

MEMORIA. Se il personaggio giocatore conosce il PNG in questione, potrebbe ricordare eventi riguardanti il PNG anche quando questi non è presente. Kathleen Fischer fa un buon uso di questa tecnica in *The Cove* e in qualche modo anche in *Redemption*. È un modo ragionevole di contrabbandare l'esposizione del personaggio sullo sfondo della storia e insieme di fornirgli un contesto anche se non dovesse mai essere disponibile nell'interazione.

EVIDENZA. Nell'assenza di un'esperienza personale, ci può essere evidenza della presenza del PNG (lettere, fotografie, possessi e così via). *Babel* e *Anchorhead* fanno entrambi buon uso di questo genere di tecnica. Il diario trovato stipato in un posto nascosto è un'espediente comune, ma può essere efficace solo se i contenuti del diario sono ben scritti e rivelano una forte personalità.

CONVERSAZIONE INTERCETTATA O AZIONE OSSERVATA. Il giocatore è in trappola. Il suo personaggio non può agire. Senza scampo, diviene testimone muto di alcuni frammenti di conversazione; o come il personaggio di Jimmy Stewart in *Rear window*, egli osserva qualcosa di cui non può fingersi indifferente. È spesso possibile escogitare una ragione per immobilizzare il giocatore per breve tempo e inserire un *daemon* per pochi tur-

ni. Il meccanismo stesso poi mostrerà se è stato usato in maniera eccessiva. *Delusions*, *Varicella* e *Spider and web* fanno tutti buon uso di una conversazione intercettata che al giocatore non è possibile cambiare.

Descrivere i PNG come oggetti statici

Come qualsiasi altro oggetto nel gioco, il PNG necessita di una descrizione fisica. Più importante è il PNG (quindi più parte del gioco egli o ella ricopre), più dettagliata e stratificata dovrebbe essere la sua descrizione. Spesso per il PNG è opportuno realizzare separatamente l'abbigliamento, elementi di vario genere e anche parti del corpo (e non solo nelle IF "per adulti").

Naturalmente questo approccio può essere portato agli eccessi. *The End Means Escape"* contiene molti personaggi descritti fino sotto le unghie delle mani e nell'ombelico, un effetto che funziona in questo contesto ma che sarebbe poco appropriato in un gioco meno surreale

Movimenti, Azione e Reazione dei PNG

Un PNG che siede su una sedia e non fa niente è meno vivo di un altro che sembra che stia perseguendo una specie di agenda personale.

MESSAGGI "RANDOMIZZATI". I messaggi randomizzati ("casuali" — n.d.T.), che appaiono mentre il PNG è visibile al giocatore, forniscono un minimo livello di vitalità. Aggiungere la frase "Mrs. MacGillicuddy sta passando l'aspirapolvere intorno al sofa" alla fine di un altro evento non connesso non è forse l'innovazione più impressionante nella caratterizzazione (ma fa almeno ricordare al giocatore che il PNG è vivo). Per i PNG di tipo , pochi messaggi random saranno sufficienti.

INTERAZIONE CON GLI OGGETTI NELL'AMBIENTE. Un gradino più su di questo è avere un personaggio che sposta o cambia lo stato degli oggetti nella stanza: posa delle cose, accende lu luci, raccoglie oggetti che in precedenza avevate lasciato qua e là. Questo è leggermente più difficile da realizzare rispetto ai messaggi randomizzati, ma dà l'impressione che il PC e il PNG si muovano all'interno e interagiscano con lo stesso ambiente, piuttosto che essere l'uno per l'altro come dei fantasmi. Il ladro di *Zork* è uno dei primi esempi realizzati di questo tipo di personaggio.

REAZIONE ALLE AZIONI DEL GIOCATORE. Come tu osservi il PNG così anche il PNG osserva te. Se il PNG reagisce alle azioni del giocatore è sulla buona strada per suggerire un'intelligenza separata e attenta.

Textfire golf di Adam Cadre porta, forse, questa tecnica al suo estremo logico: pur non essendoti concesso di conversare con i PNG, essi reagiscono in vari modi al tuo comportamento durante il corso di golf, con il risultato di sembrare interattivi e ben caratterizzati. *Kissing the Buddha's Feet* è un altro esempio anche se meno esaustivo della stessa cosa.

GIRARE ATTORNO ALLA MAPPA. Un PNG che vuole sia realizzato qualcosa è presumibile che voglia girare un po' intorno all'ambiente del gioco. Il modulo di libreria *NPC_Engine* è molto utile per gestire le varie opzioni in casi come questo: permette di far muovere i PNG su cammini casuali o pretederminati, ma anche di lasciarli trovare la propria strada da soli. Inoltre permette ai PNG di aprire e chiudere le porte e consente di progettare reazioni se essi si trovano a metà strada dalla loro destinazione e risultano bloccati. Infine ge-

stisce i verbi per seguire un PNG o chiedergli dove si trovi un altro personaggio. È davvero una piccola gemma di simulazione, con il solo svantaggio appesantire il programma, facendolo girare molto lentamente se si fa attenzione. Personalmente non l'ho mai usato nei giochi da me pubblicati, ma è solo perché non ho avuto occasione di muovere i PNG con quella tecnica.

Anche se si è fatto ricorso a questo sistema così sofisticato, muovere il PNG da un luogo all'altro ricorda al giocatore che essi non sono incollati all'ambiente in un determinato luogo.

SEGUIRE UN'AGENDA PERSONALE. Periodicamente la gente su *rec.arts.int-fiction* discute sul problema di progettare PNG che sappiano svolgere i loro (anche se semplici) compiti in modo da sembrare almeno semi-intelligenti. Potrebbero, per esempio, sistematicamente cercare chiavi perdute cercando in tutte le aree appropriate della stanza, aprire dei contenitori chiusi, guardare sotto oggetti molto grandi e così via fino a che riescono a trovare quello che stanno cercando. Il codice, per riuscire a fare questo, dovrebbe essere abbastanza flessibile per adattarsi a qualsiasi cambiamento che il giocatore ha apportato all'ambiente: il PNG non dovrebbe continuare a cercare le chiavi se il giocatore le ha messe in un luogo ben in evidenza, per esempio.

In verità, i problemi qui sono davvero considerevoli, ma vi sono state alcune innovazioni tecniche che possono rendere queste cose possibili. TADS 3 contiene un sistema di azioni e di validazioni di azioni che rende possibile per il PNG interagire con l'ambiente in più o meno gli stessi termini del giocatore. Un oggetto che sia fuori portata per il giocatore potrebbe essere reso fuori portata anche per altri personaggi. T3 è ancora lontana dalla piena maturità, ma la sofisticatezza del suo *world model* rende possibili alcuni tipi di programmazione dei PNG mai stati visti in precedenza.

Degne di nota sono anche le librerie *Reactive Agent Planner* di Nate Cull, che sono state scritte come estensione della libreria standard di Inform, di Platypus (una libreria alternativa a quella standard di Inform, *n.d.R*) e di TADS 2. Queste librerie non coinvolgono specifiche caratteristiche del *world model*. Invece, permettono al PNG di identificare compiti, creare piani per raggiungere gli scopi e modificare questi piani turno dopo turno per tenere conto dei cambiamenti nell'ambiente di gioco. Il *Reactive Agent Planner* (o RAP) è abbastanza astratto e un po' difficile da comprendere in un primo tempo, ma approfonditane la conoscenza può produrre alcuni dei comportamenti per i PNG più sofisticati e intriganti.

Interfacce di conversazione

La conversazione è sicuramente l'aspetto più difficile da gestire di un PNG. Ci sono doversi problemi, ma uno dei più pressanti è che non possiamo ancora esprimere nell'Interactive Fiction una conversazione nella stessa maniera in cui la esprimiamo nella vita reale. La cosa più naturale da fare, allora, dal punto di vista del giocatore, sarebbe quella di scrivere esattamente cosa egli vorrebbe dire, usando l'inglese (oppure l'italiano o qualsiasi altra lingua) per trasmettere sia il contenuto che il tono. Il PNG ideale capirebbe perfettamente e reagirebbe di conseguenza al comportamento del giocatore oltre che al contenuto reale di quello che sta dicendo.

Al momento, comunque, una conversazione di questo tipo va oltre le nostre capacità tecniche. Inoltre, mentre potrebbe essere una soluzione ideale vista nella prospettiva del giocatore, si rivelerebbe un incubo per l'autore. In altri aspetti del design della IF, è possibile limitare il numero di cose che il giocatore può ragionevolmente fare: ci sono solo alcuni verbi, e solo un insieme specifico di oggetti è disponibile; inoltre lo scopo dell'azione è chiaramente ben comprensibile. Un PNG che capisce tutti i temi di conversazione, tutti i tipi di

tono della stessa e tutti gli stati d'animo, mai potrebbe essere programmato in maniera esaustiva; ci sarebbe sempre qualcosa che non è stato previsto. E un tal personaggio supererebbe certamente i limiti della trama pensata dall'autore. Sarebbe difficile scrivere una storia che abbia un senso di struttura o continuità, se il plot desiderato fosse messo da parte mentre il giocatore insegna al PNG le regole del cricket.

Quindi, invece dell'utilizzo del linguaggio naturale, abbiamo un assortimento di vari sistemi di successo che permettono al giocatore di comunicare le sue intenzioni di conversazione al parser.

Vorrei far notare che qui si discute sull'utilità dei vari sistemi nella caratterizzazione e non sugli aspetti tecnici per arrivare a questi effetti. Per una panoramica di questo argomento, almeno in Inform, consultate il tutorial di Roger Firth *InfAct*. Per una panoramica ulteriore dello stesso argomento, ma con differenti conclusioni dal mio punto di vista, date un'occhiata anche *Choosing a Conversation System* di Michael J. Roberts

CONVERSAZIONE SI/NO. È un sistema limitato: Andrew Plotkin ha fatto un brillante lavoro nel maneggiarlo in *Spider and Web*, ma non è particolarmente adatto alla maggior parte dei giochi. L'ho riportato qui come esempio della forma minima possibile di conversazione.

PARLA A. Quando il giocatore desidera comunicare, egli scrive: `> PARLA A JONES` e la conversazione avviene senza ulteriori comandi da parte sua. Qui i vantaggi sono, primo, che si possono inserire dialoghi realistici, appropriati alla situazione e provenienti dalle bocche sia del PG che del PNG, e poi che non ci si deve preoccupare di fare il parsing di frasi che possono risultare comiche, ma solo di disabilitare (se sono implementati in un linguaggio di programmazione comune) i comandi *ASK e *TELL*. Lo svantaggio è che lascia relativamente poco potere nelle mani del giocatore (la cui unica scelta si ridurrebbe a scegliere di avere una conversazione oppure no). *PARLA A* inoltre chiude il giocatore in qualsiasi caratterizzazione si sia scelta per il PG. *Common round* di Stephen Granade esemplifica molto bene sia i vantaggi che gli svantaggi, almeno secondo me. *Kaged* di Ian Finley e *Masquerade* di Kathleen Fischer, ma anche altri giochi hanno fatto uso del *PARLA A*.

Questa è un'arena nella quale l'abilità di scrittura dell'autore si rivela essenziale per il successo di un gioco o al contrario per il suo fallimento. Se si è un autore pieno di potenzialità e capace di trasportare una quantità di personaggi e emozioni anche in questi pezzi di ambientazione (come fa Ian Finley, a mio parere), si saprà certamente mantenere un senso di immersione.

CONVERSAZIONI A MENU. Quando il giocatore desidera comunicare, scrive `> PARLA A JONES` e sullo schermo appare un menu con un numero di frasi da tre a sei che può dire in quel momento. Ci può anche essere un'opzione per non dire nulla. Jones poi ti risponde e al giocatore appare un altro menu, e così via fino a che la conversazione termina. *Photopia* adotta un sistema simile e alcune delle librerie esistenti per l'utilizzo dei menu in Inform sono, almeno in parte, derivate da quelle di *Photopia*.

Il vantaggio qui è che ancora una volta l'autore ha il controllo sulla forma di comunicazione. Si possono fornire al giocatore alcune battute da dire, caratterizzando sia il PG che il PNG. (Per una caratterizzazione del giocatore che ha luogo in massima parte nella libreria di gestione dei menu e per l'utilizzo di essi da parte del PG, vedi *Rameses* di Stephen Bond.)

Il problema è che i sistemi di menu sono piuttosto restrittivi; a volte il menu non contiene nulla che il giocatore vorrebbe dire e non c'è modo per cambiare le opzioni presenti, ma

neanche l'illusoria sensazione di libertà che viene dal poter scrivere `> CHIEDI A JONES DELLA TEOLOGIA`, anche se non è stata implementata alcuna risposta.

Un ulteriore problema è ciò che Duncan Stevens descrive come **l'effetto tosaerba**. Se mi dai una serie di menu, non devo fare nessuno sforzo per entrare nella conversazione, e posso metodicamente (usando il comando *annulla*, per esempio) tornare indietro e scegliere le diverse varianti, prendendo ora il primo e ora il secondo cammino, finchè non sono sicuro d'averle provate tutte. Il PNG è così finito, senza più pensieri da parte mia di quelli che avrei avuto muovendo un tosaerba.

Dal mio punto di vista questo diminuisce l'immedesimazione. Se si sta scrivendo un gioco molto lineare come *Photopia* o *Rameses* o *Being Andrew Plotkin* (preferibilmente qualcosa di così vivamente scritto che la storia o il senso d'umorismo della narrativa mi farà desiderare di spostarmi in avanti il più rapidamente possibile) questo tipo di conversazione potrebbe essere l'ideale. Se invece si sta scrivendo un gioco basato sull'investigazione, che permette al giocatore di modellare il proprio personaggio, o che lascia molte parti del plot nelle mani del giocatore, allora sarebbe preferibile utilizzare qualcosa di più "aperto" e libero.

ASK/TELL. *ASK/TELL* è una funzionalità standard di Inform (*CHIEDI/PARLA* nella libreria Italiana INFIT, *N.d.T.*) ed è la più comune forma di interazione con i PNG nei giochi Infocom e in alcuni altri lavori della vecchia scuola. Questa funzione permette al giocatore di "chiedere" o "parlare" al PNG di un qualsiasi argomento ed ottenere una risposta. Questo approccio è ovviamente più flessibile per il giocatore che non un menu di conversazione e funziona meglio con degli enigmi basati sulla conoscenza, dove il giocatore aumenta il proprio grado di conoscenza, scoprendo e chiedendo nuove informazioni.

Dal punto di vista del giocatore, comunque, questo significa una caratterizzazione minima del PG e restrizioni severe su cosa è possibile esprimere. Di solito il gioco non accetta più di una o due parole e quindi `> CHIEDI A JONES DELL'ORA DELL'OMICIDIO` fallirebbe. *1-2-3*, uno dei giochi presentati alla "IF Comp 2000" prova ad affrontare questo problema, ma lo fa con un "prompt" incredibilmente noioso oltre ad essere molto poco flessibile circa QUALE lunga stringa di parole accetta come valida.

INPUT IN "LINGUAGGIO NATURALE" BASATO SU PAROLE CHIAVE. Il gioco di Jon Ingold *Insight* dà l'opportunità al giocatore di scrivere domande naturali complesse per esempio *BOB, CHI È TUA MOGLIE?* oppure *FRED,PERCHÈ SEI ARRABBIATO?* Dal momento che non ho visto il codice sorgente, non sono sicura di come funzioni esattamente questo codice, ma presumo che estrapoli le parole chiave (MOGLIE, ARRABBIATO) e identifichi il tipo di domanda (CHI, PERCHÈ, ecc.), effettuando una "triangolazione" a partire da questi dati per generare una risposta appropriata. Ecco un'estratto (relativamente privo di spoiler) dalle fasi iniziali del gioco[o]

> *> man, who are you?*
> "My name's Mackenzie. But I, er, guess you already knew that. What do you want to know? You know it all already, right? I've been working - living - in Olympia. I'm a genetic designer."
>
> *> mackenzie, what is your name?*
> "You already know my name, of course you do," he replies.
>
> *> mackenzie, where is olympia?*
> "Nice enough place, I guess," he says. "We have a lot of problems with the

o L'estratto del gioco è stato lasciato nella versione originale inglese (N.d.T.).

windstorms because of the nearby mountains. I've been working on solutions for that, using plants."

> *mackenzie, do you come from olympia?*
— Please be more specific about what you want to say.

> *mackenzie, who else lives in olympia?*
"I'm sorry," Mackenzie replies. "I didn't quite follow that."

Dal momento che funziona, lo trovo parecchio affascinante. Laddove il parser realizza che non può interpretare la domanda, dà anche una scusa piuttosto soddisfacente. È il caso medio che è il più sconcertante: il gioco capisce la maggior parte delle cose, ma perde alcune sfumature critiche. La risposta per "Where is Olympia?" (Dov'è Olympia?) non è chiaramente attinente; sembra aver afferrato la parola chiave "Olympia" ma non aver interpretato bene il "dove?". E la linea di risposta "Please be more specific" (per piacere, sii più specifico) non è di grande aiuto, giacchè la domanda mi sembra essere abbastanza specifica.

Non intendo essere troppo critica, perché il gioco tenta di fare qualcosa di estremamente difficile. Il problema è che falsificare la comprensione del linguaggio naturale lascia sempre delle notevoli falle. Quello può essere dimenticato in un chatterbot diviene più serio in una IF, dove il successo o il fallimento di una interazione determina la possibilità o l'insuccesso per il giocatore di conoscere il resto del plot e completare il gioco in maniera soddisfacente.

VERBI DI META-CONVERSAZIONE. Varicella di Adam Cadre utilizza una forma modificata di ask/tell che consente al giocatore di avere un po' più controllo sul comportamento del PG. Il sistema ask/tell funziona allo stesso modo di sempre, ma permette adottare uno dei tre toni di voce: ostile, cordiale e servile. Per avere un esempio dall'inizio del gioco:[o]

> *tone cordial*
You adopt a cordial manner.

>*ask steward about nails*
"How's the manicure proceeding?" you ask.

"Shouldn't be much longer, sir," the steward says.

The steward expertly attends to your fingernails with an emery board.

>*tone hostile*
You adopt a hostile tone.

>*ask steward about nails*
"How much longer is this going to take, you mediocre manservant?" you bellow.

"Shouldn't be much longer, sir," the steward says.

The steward lightly blows on your fingertips.

>*tone servile*
You adopt a servile posture.

o L'estratto del gioco è stato lasciato nella versione originale inglese (N.d.T.).

> *>ask steward about nails*
> You're scarcely about to address a common servant in an obsequious tone. For heaven's sake, where is your self-respect?

Le reazioni a questo sistema sono state diverse. Ho trovato divertente girare attorno per vedere quali variazioni interessanti sulle risposte avrei ricevuto cambiando il mio tono di voce, ma ho spesso dimenticato di inserire il tono corretto e mi son ritrovata ad agire in maniera inappropriata. E più ero presa dal gioco, più facilmente dimenticavo del sistema di tono, il che significa che lo usavo come un giocattolo piuttosto che per avere le interessanti variazioni che penso, come risultato, rimangano seppellite lì.

Un altro esperimento simile è *Forever Always* che permette al giocatore di usare avverbi per controllare il tono della conversazione. Il giocatore può, per esempio, SUSSURRARE RAUCAMENTE, GRIDARE RABBIOSAMENTE, PARLARE EDUCATAMENTE, etc. Appare un menu di opzioni, e i suoi contenuti dipendono dal tipo di tono si è scelto di adottare. Questo sistema, diversamente da quello di *Varicella*, lascia vedere al giocatore cosa sta per dire prima che lo dica, così l'effetto di un tono diverso è un po' più ovvio. Il gioco non è impeccabile, ma i problemi nelle ultime scene sembrano imputabili più a dei bug e al poco beta-testing che a problemi del sistema, che risulta chiaramente interessante.

Sia *Varicella* che *Forever Always* hanno sistemi progettati per un tipo di gioco dove gli stati emozionali e le relazioni fra i personaggi sono di primario interesse; l'intrigo di palazzo del primo, la parodia del romanzo sentimentale il secondo. Il sistema di *Forever always* potrebbe non funzionare con un gioco incentrato sulla raccolta di informazioni, poiché il giocatore non può specificare le parole chiave, e non c'è alcuna potenzialità da utilizzare per esguire, ad esempio, gli indizi di un mistero. Dall'altra parte, penso che funzoni meglio dell'approccio usato da *Varicella* per il preciso e limitato scopo di creare IF basate sullo stato emozionale dei personaggi. (Poiché *Varicella* si basa in parte sulla scoperta di informazioni, l'approccio solo-avverbi non avrebbe funzionato).

Comunque, una parola di avvertimento. È importante, se si espande un sistema di conversazione includendo nuovi verbi, non lasciare il giocatore con un numero ingestibile di opzioni. In *Varicella* era possibile tenere tre diversi toni di voce, ma altri consigli che ho sentito (come un comando `>SII COMPRENSIVO`) o provato a implementare da me stessa (un sistema che includa RIPOSATI, INSULTA, CHIEDI SCUSA, AMOREGGIA, SEDUCI, SORRIDI, RIDI) possono portare a confusione: c'è troppo da scrivere per l'autore e il giocatore non ha abbastanza direzioni da seguire. È possibile scrivere un buon gioco con un sottoinsieme di queste opzioni, ma non ne ho mai visto uno finora. L'autore dovrebbe porsi diversi limiti nella progettazione, penso.

ARGOMENTI. Usato in giochi quali *Smarrow's Song* di J.D. Berry (che ha partecipato alla SmoochieComp) e in un vecchio gioco della IF Comp come *She's got a thing for a spring*, è un sistema per certi versi simile a ask/tell, eccetto per i verbi che sono omessi. Il giocatore scrive semplicemente una parola che vuole proporre come argomento di conversazione e questa procede in accordo. Secondo me questo metodo non costituisce una nuova interfaccia essendo un leggero ridimensionamento di una già esistente (ma permette all'autore di non dover programmare reazioni separate per ASK/TELL).

IBRIDO MENU MODIFICATO/ARGOMENTO. Questo sistema combina la libertà di ASK/TELL con un sistema a menu. Quando si inizia una conversazione con qualcuno, compare una lista di possibili cose da dire nella status line, ed è possibile dire una di queste cose, semplicemente scrivendo la lettera corrispondente. Se, comunque, si vuole cambiare il soggetto della conversazione, si può anche scrivere `> ARGOMENTO COLLANA DI DIAMANTI` e appare un nuovo menu. Per esempio, `> ARGOMENTO JONES` potrebbe far apparire un menu

come questo:

> 1. Hai visto Jones da qualche parte?
> 2. Cosa ci fa Jones qui?
> 3. Da quanto tempo Jones lavora per la compagnia?
> 4. Cosa ne pensi di Jones?

Questo ci sbarazza del problema tosaerba e obbliga il giocatore a prendere l'iniziativa scegliendo in che modo la conversazione deve procedere. E questo significa anche che si può consentire al giocatore di fare domande più complesse di quelle che sono permesse nel sistema ASK/TELL, ma senza rovinare il gioco includendo domande del tipo: `> CHIEDI ALLA REGINA SE È VERO CHE HA RUBATO I DIAMANTI DELLA PRINCIPESSA` nello stesso menu con `> REGINA, SALVE` e con `> SAPRESTI DIRMI DOVE TROVARE DELL'ALTRO SQUISITO CAVIALE?`

Passando ad un livello leggermente più tecnico, il sistema permette degli UNDO "a singolo scambio". In una conversazione a menu, un problema comune è che l'intera conversazione avviene in un solo turno di gioco: se non ti piace come si evolve, devi annullare tutto e iniziare da capo. E questo è molto irritante.

Un grosso inconveniente di questo sistema è che per essere utile richiede più testo scritto da implementare rispetto a tutti gli altri. Il semplice ASK/TELL di solito richiede che l'autore debba scrivere due risposte per il personaggio per ognuno dei più importanti argomenti di conversazione; se la trama poi è complicata, o è possibile portare il PNG a diversi stati mentali, allora l'autore deve scrivere alcune varianti a queste risposte. Con il sistema a menu, per dare l'impressione di una piena implementazione, l'autore deve scivere molteplici domande e risposte per ogni argomento e per ogni personaggio. Questo può rapidamente scivolare nel ridicolo.

Memoria e contesto

Al di là del problema di inserire nella conversazione alcuni argomenti, c'è anche quello di renderla plausibile.

CONTRASSEGNARE GLI ARGOMENTI GIÀ DISCUSSI. Uno dei comportamenti meno 'umani' del tipico PNG dell'interactive fiction è quello di rispondere sempre allo stesso modo alle stesse domande, indipentemente dal fatto di quante volte gli siano state poste. Questo non accade nei sistemi di conversazione basati sui menu, in cui si possono diabilitare le domande che sono state già fatte, mentre invece non c'è modo di impedire al giocatoredi scrivere `> CHIEDI A JONES DEL CAPPELLO` dieci volte di seguito.

Al livello più basso, è solo una questione di impedire che il PNG ripeta sempre le stesse cose. Le persone reali non ripetono le stesse parole nella stessa lingua cento volte di seguito, e se un PNG lo fa viene meno la sensazione di realismo.

Sul newsgroup *rec.arts.int-fiction* ci sono state numerose conversazioni su quale tipo di risposta sia la migliore nel caso in cui un giocatore continui a rifare la stessa domanda. Opzioni accettabili, secondo me, possono essere: il parser risponde: "Ricordi che Jones ti ha detto che…"; faare in modo che Jones risponda nuovamente ma in maniera leggermente diversa grazie a del testo casuale (in modo che si ricevano sostanzialmente risposte simili ma non identiche); descrivere la conversazione senza ripetere esattamente le stesse parole ("Jones ti dice ancora una volta che…").

GESTIRE UNA FORMA DI RIASSUNTO. Un'altra cosa carina, derivante dal fatto di contrassegnare gli argomenti di conversazione già trattati è quella di offrire ai giocatori,

una sorta di verbo **RICORDA** per fargli ricordare tutti gli argomenti di cui si è già discusso. In giochi con un alto tasso di conversazioni potrebbe essere l'unico modo per evitare al giocatore di continuare a tener nota di quello che si dice.

ABILITARE NUOVI COMMENTI. Durante il corso di una conversazione, quando un argomento viene affrontato, nuovi commenti vengono naturali. Contrassegnando quali commenti/argomenti sono stati già affrontati, si possono aggiungere nuove voci di menu al momento opportuno, o consentire al giocatore di digitare. `> CHIEDI DI...` riguardo i nuovi argomenti (o ricevere nuove risposte se il giocatore fa domande su argomenti precedenti).

REAZIONI BASATE SUL CONTESTO. Nella vita reale, se state parlando ad una persona e quella inizia a leggere un libro, potreste ricavarne un messaggio. Allo stesso modo, potrebbero esserci dei momenti nella conversazione in cui potrebbe essere più o meno appropriato reagire ai comportamenti dell'altra persona con delle "avances" (`> BACIA JONES`) o con violenza (`> UCCIDI JONES CON LA ROCCIA`). Con un sistema di conversazione che tiene conto di quale sia l'argomento di conversazione, e se qualcosa di questo tipo sta accadendo, si può usarlo per definire reazioni appropriate a particolari tipi di azioni come BACIA, DAI, MOSTRA, COLPISCI, ecc.

In maniera simile ma più sottile, il contesto nella conversazione può anche essere usato per interpretare il significato dei comandi del giocatore. Ad esempio:

> *[Il personaggio giocatore e l'ispettore Lynley hanno discusso di vittime di omicidio.]*
>
> *>Chiedi a Linley di Veronica*
> "Pensi che possa essere stata Veronica?" suggerisci. "Ho sentito per caso che stava discutendo con la vittima la notte scorsa."

Al contrario di:

> *[Il personaggio giocatore e l'ispettore Lynley hanno appena discusso delle loro vite amorose.]*
>
> *>Chiedi a Linley di Veronica*
> "La conosci bene Veronica?" chiedi. "Mi piacerebbe chiederglielo, ma non sono sicuro se le cose vadano bene tra lei e Marcus."

Questo tipo di raffinamento del dialogo è irrilevante, ovviamente, in un sistema di conversazioni basato sui menu, ma per il tipo ASK/TELL può dare un buon senso di profondità. Tuttavia, porta via molto lavoro, per assicurarsi che le domande davvero fondamentali non diventino non accessibili solo perchè il contesto della conversazione non è mai quello giusto. Se il giocatore vuole accusare Veronica di omicidio, sarà colmo di frustrazione se il gioco gli permette solo domande sulla sua vita amorosa.

CONOSCENZA ASTRATTA. Una delle abilità artificiali che ci piacerebbe dare ai nostri PNG, a parte quella di muoversi in modo intelligente nelle locazioni e cercare di raggiungere obiettivi complessi, è quella di capire ciò che viene detto loro: tenere traccia degli elementi di conoscenza che essi hanno fino ad allora, usarli per cambiare i loro piani e i loro scopi, e persino trarre delle inferenze logiche da quello che hanno imparato.

Così come il coinvolgimento dell'intelligenza artificiale nella manipolazione del mondo, questo è un problema teoretico che rimane sostanzialmente irrisolto, e che probabilmente produrrebbe risultati significativi solo in un numero abbastanza limitato di giochi. Ad ogni modo, alcune librerie sono state scritte per iniziare a cercare di risolvere questo tipo di pro-

blemi. Per esempio, ce ne sono alcune nella collezione “Onyx Ring”, che sono progettate in modo da consentire al PNG di imparare fatti nuovi e riuscire a comportarsi di conseguenza.

Usare modelli astratti di conoscenza come questo potrebbe portare ad una maggiore flessibilità, ma potrebbe creare grosse difficoltà sul controllo del gameplay e sul fatto di non rendere troppo meccanica la scrittura del gioco.

Stati d’animo e emozioni

Lo stato emozionale del PNG è effettivamente un altro tipo di contesto per l’argomento che stiamo trattando. È chiaramente facile tracciare emozioni dando al PNG una variabile o una serie di variabili che sono affette da ciascun scambio di conversazione (così che un insulto incrementa la variabile Rabbia, per esempio). È più conveniente adoperare le risultanti variabili per un buon effetto.

GESTI. I segnali emozionali possono essere inseriti in una conversazione attraverso il linguaggio del corpo. Una persona irritata potrebbe avere un tic nervoso; una persona lussuriosa potrebbe lanciare occhiate fiammanti nella tua direzione.

Più importante è lo stato d’animo del personaggio durante il gioco, più bisogno si avrà di segnalare i suoi cambiamenti e il suo stato corrente al giocatore. Modificare la descrizione di un personaggio potrebbe essere sufficiente per far conoscere al giocatore quale sia lo stato corrente in alcuni casi. In altri (specialmente se lo stato d’animo del giocatore cambia frequentemente nel corso di una conversazione in cui è molto preso) può essere necessario rendere quei cambiamenti più espliciti. Dopotutto, se il giocatore sta conversando con un PNG, è lecito supporre (a meno che la conversazione non avvenga per telefono o al buio) quello che si possano vedere l’un l’altro, e che alcuni drammatici cambiamenti nell’attitudine del personaggio saranno subito visibili.

RISPOSTE VARIABILI BASATE SULLO STATO D’ANIMO. Ovviamente una persona irritata reagirà a certi argomenti con più veemenza di una persona calma. Inserire cambiamenti nel dialogo stesso per testare queste variabili è un altro modo fine di gestire stati d’animo e emozioni.

Iniziativa di conversazione

ARGOMENTI GUIDA. I PNG danno l’impressione di essere molto più attivi e intelligenti se mostrano segni di avere un’agenda personale da seguire (che potrebbe includere l’introduzione di nuovi argomenti di conversazione, decidere di tagliar corto in un dialogo, e così via). C’è un altro vantaggio: il PNG che prende iniziativa e non aspetta il giocatore può sembrare più vivo e dinamico di un PNG che aspetta seduto di essere interrogato a capriccio del giocatore, infarcito di turni del PG che fa l’inventario, che guarda sotto i cuscini del sofa, e che apre casseforti. E se si ha una specifica lista di informazioni da dare al giocatore, alle volte è utile avere un PNG che ritorna su quell’argomento finchè non è stato trattato in maniera adeguata.

D’altra parte, se il PNG è troppo attivo e troppo “manovrato”, il giocatore potrebbe avvertire la sensazione che con lui non c’è niente da fare. Potrebbe impappinarsi sugli argomenti di cui si parla, provare a indovinare il nome oppure avere guai con l’interfaccia, e tutto questo mentre il PNG continua tranquillamente a blaterare. Può essere stimolante cercare di far comportare il PNG nel modo giusto, quindi forse la cosa migliore è evitare catene di

conversazione troppo lunghe; se il PG non sta dicendo niente, il PNG dovrebbe fermarsi anch'esso.

Questa è un'altra area che non è stata ancora esplorarata completamente, così non vi sono buoni esempi da fornire. Nel provare una qualunque nuova direzione nello sviluppo dei PNG, io raccomando un lungo e puntiglioso playtest, che inizi addirittura con le versioni alfa. Non strutturare il gioco in maniera rigida prima di un feedback sul suo effettivo funzionamento. È facile progettare qualcosa di molto elaborato, ma completamente ingiocabile.

Spring Thing 2005 e ORGC

di Roberto Grassi

Durante questi mesi si sono tenute due competizioni interessanti per diversi motivi. La prima, la Spring Thing, è una competizione di IF che era nata ed era stata effettuata nel 2001 e 2002 per volontà di Adam Cadre (l'autore di *Photopia*), con lo scopo di privilegiare giochi di IF di durata medio-lunga e che si distinguessero per la qualità. Si indirizzava sostanzialmente a quegli autori e a quei giochi che per diversi motivi non potevano partecipare alla IFComp (considerato a torto o a ragione l'evento principale della comunità mondiale delle IF), in cui sono invece privilegiati giochi di durata non superiore alle due ore.

Negli anni 2003 e 2004 la competizione non è stata effettuata. Quest'anno invece, per iniziativa di Greg Boettcher, la Spring Thing è stata ripristinata, con piccole variazioni al regolamento. Sei giochi sono stati sottoposti alla giuria (chiunque poteva votare, a patto che li avesse giocati e che votasse per tutti) e la classifica finale è la seguente.

1. *Whom The Telling Changed*, Aaron A. Reed — 7.13
2. *Bolivia By Night*, Aidan Doyle — 6.77
3. *Threnody*, John "Doppler" Schiff — 6.25
4. *Flat Feet*, Joel Ray Holveck — 5.92
5. *Second Chance*, David Whyld — 5.38
6. *Authority*, Eva Vikström — 2.79

L'unico che ho giocato per intero è lo splendido *Whom the telling changed*. Trovate la recensione su questo numero di Terra d'IF e lo consiglio caldamente. Ho dato una veloce occhiata agli altri. Per gli amanti di *Sam & Max* (la mitica AG della Lucas, consiglio di giocare Flat Feet).

La seconda competizione è stata la italiana ORGC (One Room Game Competition) organizzata, come da tradizione, da Francesco Cordella. Spiace dire che si è trattato di un mezzo fallimento, come scrive Francesco sul suo sito.

Questo fallimento è determinato dalla scarsa partecipazione; infatti, un solo gioco è stato presentato e di conseguenza ha vinto il premio. Si tratta di *L'Armando* di Andrea Rezzonico, una simpatica IF ispirata alla omonima canzone di Enzo Jannacci.

Vorrei però trovare la nota positiva. Un nuovo autore ha trovato la voglia e il tempo di scrivere un gioco per noi. Diamo atto di questo ad Andrea e lo invitiamo a proseguire.

Vecchie avventure per Spectrum

di Stefano Guida

Le avventure testuali su Spectrum nascono verso la fine degli anni 80. Le prime giravano tranquillamente sulla versione 16K ed erano puramente testuali. Successivamente venne introdotta anche una spartana grafica vettoriale. Una delle piu' celebri e' sicuramente *The ship of doom* che si puo' tranquillamente scaricare sul sito assieme a oltre 5000 titoli di giochi originali internazionali prodotti per lo ZX

In Italia vennero pubblicate inizialmente su *Load'N'Run*, in mezzo ai giochi e alle utility, le avventure semigrafiche tutte molto simili come *Sire Fire* uno dei giochi più celebri che venne riproposto anche in una seconda versione visto il successo ottenuto. Fu la prima avventura a cui giocai e di cui conservo un bellissimo ricordo. All'inizio ci si trova nel "bosco incantato" e lo scopo del gioco è quello di raggiungere il classico castello di cui si deve possedere le chiavi.

Il freddo prompt accetta i classici punti cardinali per muovere il personaggio e bisogna stare attenti quando si digitano i comandi in quanto solo una frase viene accettata dal sistema (per cui solo una frase come "guardo il quadro" o "osservo il quadro" può essere accettata). Purtroppo la funzione "aiuto" genera sempre il solito sconsolante messaggio "mi spiace ma non posso aiutarti".

Ammetto che per trovare la soluzione ho dovuto guardare il disassemblato del gioco e cercare le frasi ma e' stato ugualmente difficile.

Su *Load'n'Run* le avventure erano molto simili, prodotte sicuramente dallo stesso programmatore di cui non si conosce il nome. In ogni caso per chi volesse provare una di questi giochi può tranquillamente scaricarle dal mio sito nell'apposita sezione.

Fortunatamente, alla fine degli anni '80 nelle edicole cominciarono a circolare riviste come **Epic 3000** e **Viking** della Brainstorm Enterprise dedicate esclusivamente ai cosiddetti "adventure": giochi spesso proposti in più parti con un mezzo schermo grafico, ben descritte e il cui autore è Bonaventura Di Bello.

Purtroppo il mercato dei giochi in edicola era in mano alla SIPE che produceva le famose **Special Playgames** e **Special Program**: giochi originali modificati in modo da eludere le leggi sul copyright. Ad un prezzo di sole 8000 lire vendevano nastri con 20, 26 o 30 programmi dalla grafica molto più accattivante delle 3 avventure testuali presenti su Viking ed Epic3000. Questo fece sì che queste ultime chiudessero la produzione dopo pochi anni.

Ecco quindi che sul mercato cominciarono a trovarsi le avventure grafiche: molto più attraenti da vedersi ma che lasciavano meno spazio all'immaginazione e alla fantasia.

Il newsgroup delle avventure testuali

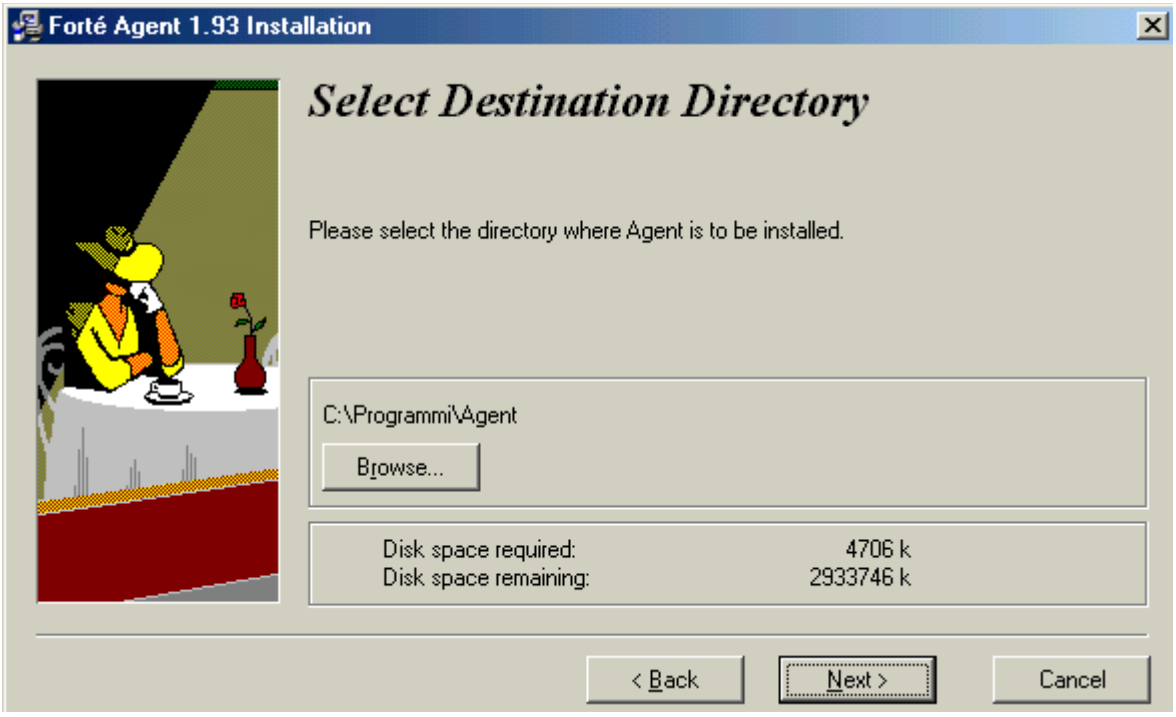
di Vincenzo Scarpa

I newsgroup sono delle pubbliche discussioni, suddivise per argomenti, alle quali si può accedere tramite un apposito newsreader. Spesso infatti, quando si gioca a un'avventura testuale si ha bisogno di qualche piccolo aiuto per andare avanti, e non sempre l'autore di quest'ultima è facilmente reperibile; oppure non riusciamo a creare del codice adatto alle nostre esigenze e non sappiamo a chi chiedere. Niente paura, perché il newsgroup può essere immaginato come un "luogo" all'interno del quale "si incontrano" tutti (o quasi) gli appassionati di avventure testuali ai quali è possibile chiedere qualsiasi tipo di informazione inerente l'argomento e ricevere risposte (quasi sempre) appropriate. Oppure, potete solo leggere i messaggi (fare cioè i cosiddetti lurker) senza per questo dare fastidio a nessuno.

Detto questo, possiamo iniziare ad occuparci del newsreader. Ne esistono parecchi, ma il mio preferito è sicuramente Free Agent[o], prodotto dalla Forté Internet Software e completamente freeware. Vediamo allora come installarlo.

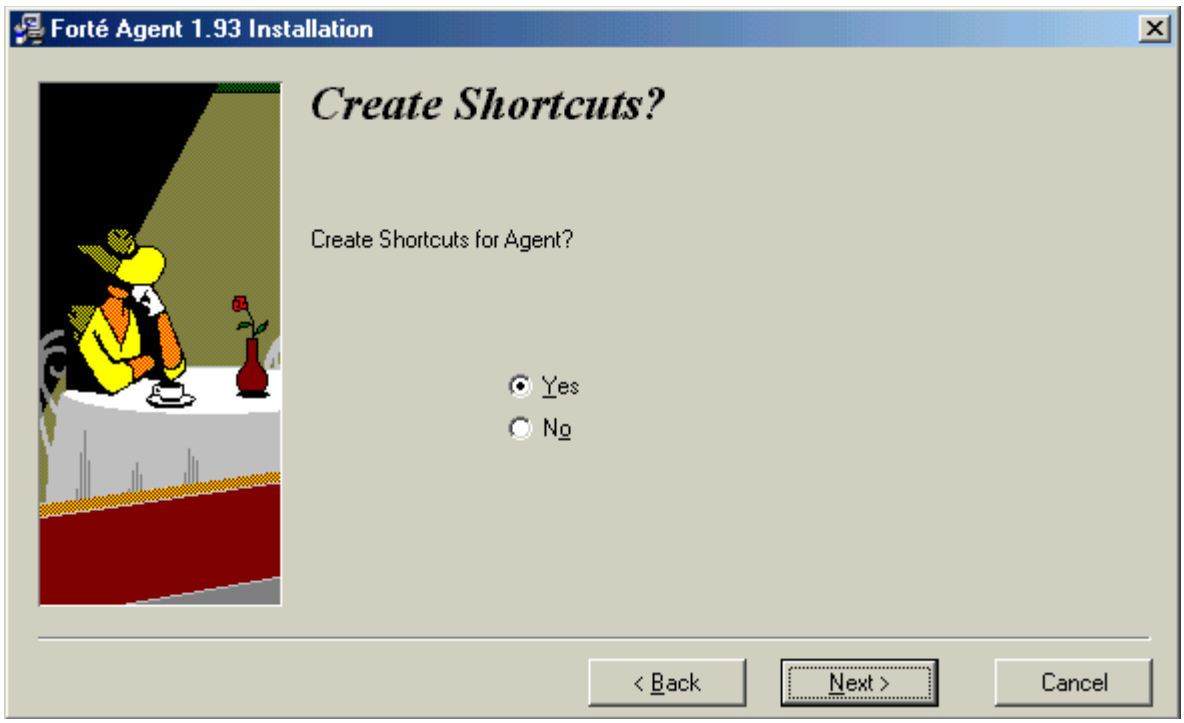
La prima cosa che ci viene chiesta è se vogliamo creare una nuova installazione (quello che a noi interessa) o se aggiornare una copia già installata del programma. Dopo aver selezionato la prima opzione e aver cliccato con il tasto sinistro del mouse il pulsante *Next*, ci appare la seconda videata:

o Anche se è in inglese, Free Agent può essere tranquillamente usato anche da chi questa lingua non la conosce.

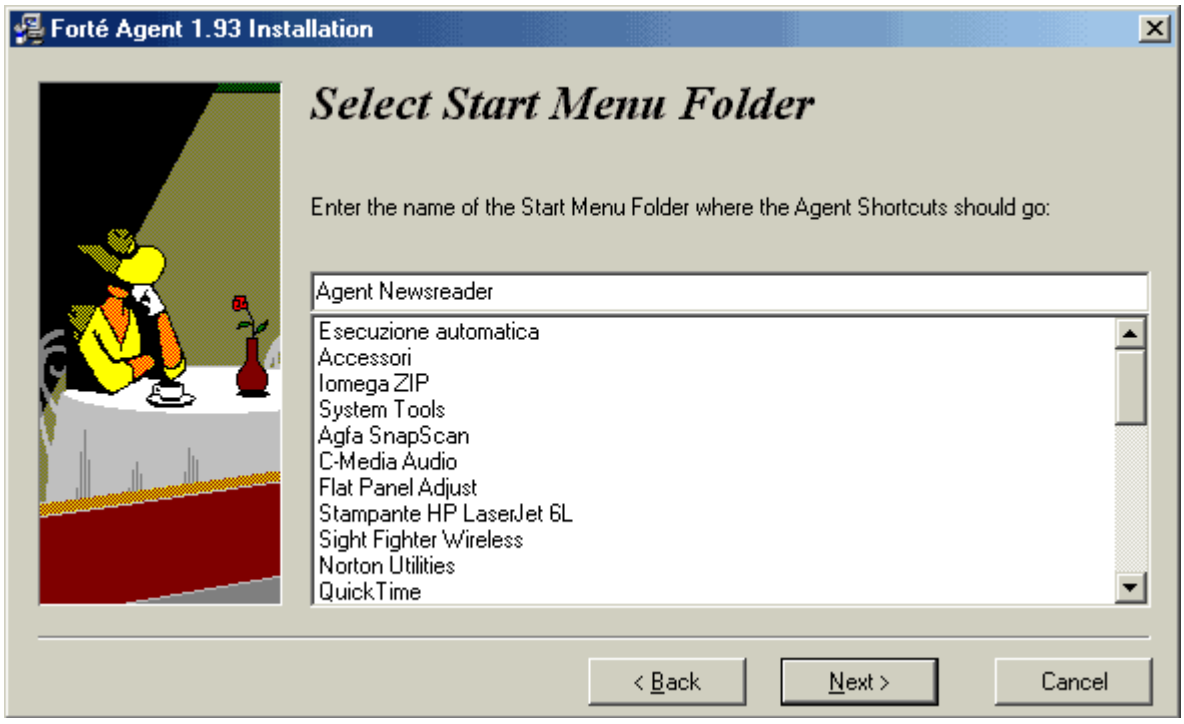


all'interno della quale occorre specificare la directory nella quale installare Free Agent. Lasciamo quella predefinita e proseguiamo con la terza videata:

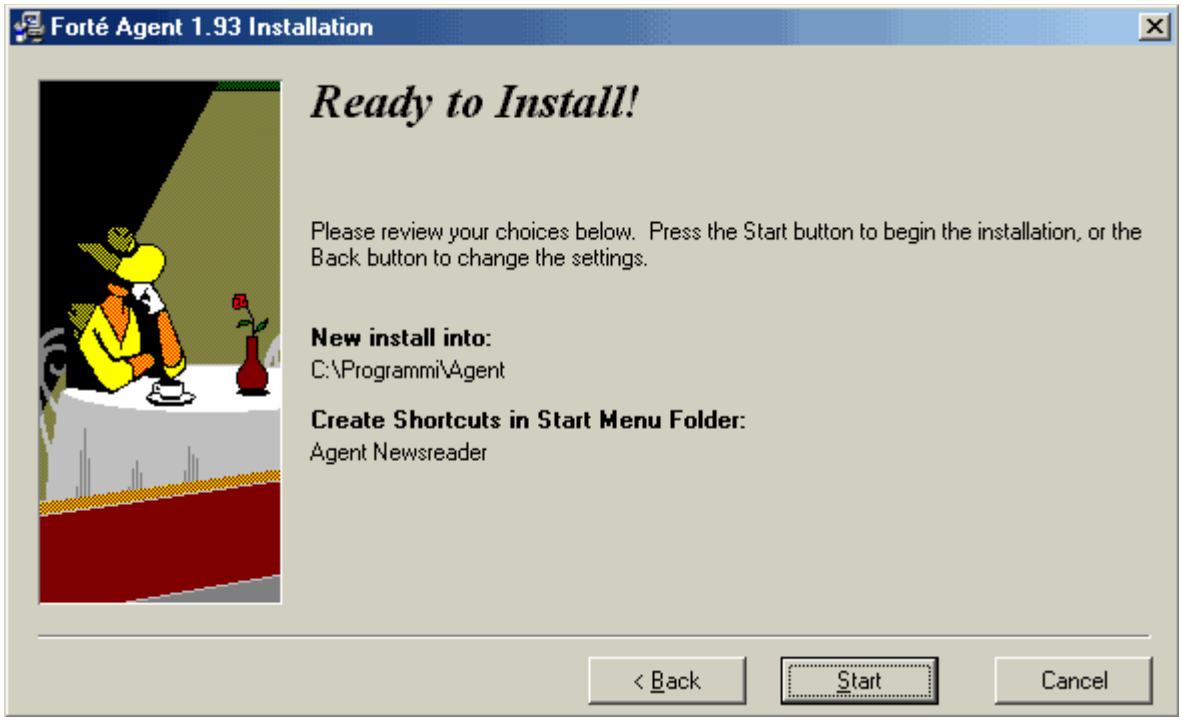
Qui ci viene chiesto se vogliamo installare dei caratteri di supporto per la lingua inglese (non è compreso l'italiano). Lasciamo stare tutto così com'è e proseguiamo:



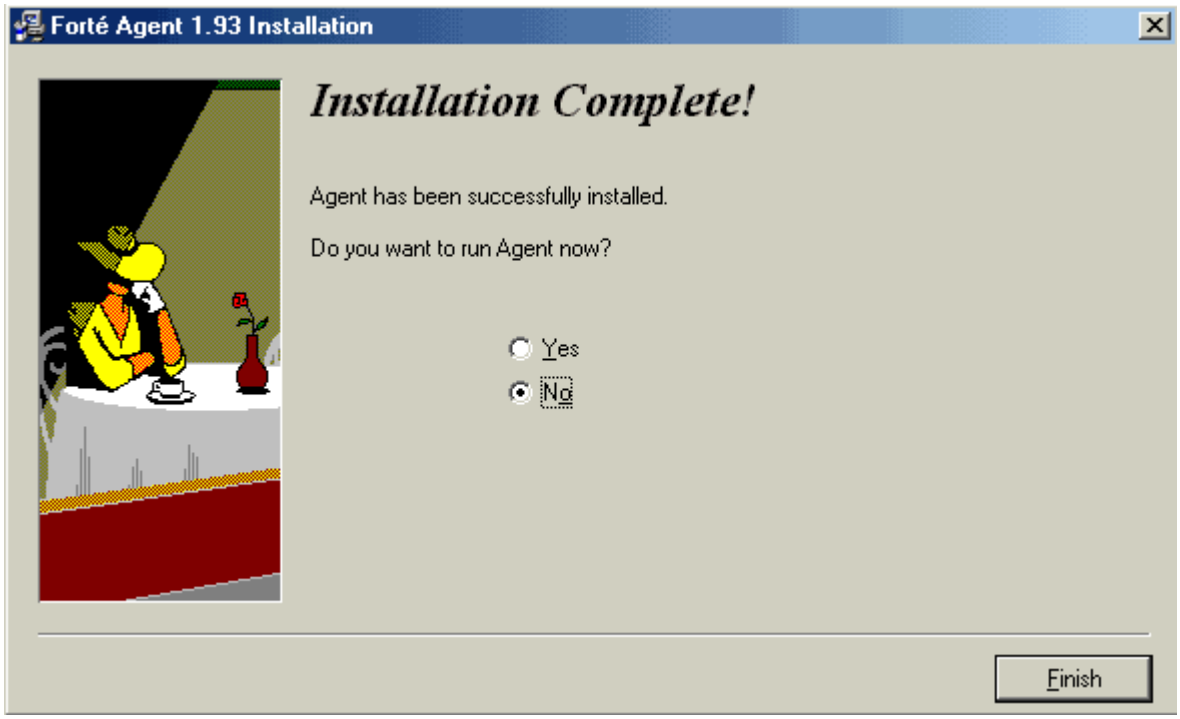
Possiamo scegliere se creare o meno un'icona del programma. Lasciamo selezionata l'opzione predefinita e proseguiamo:



In questa videata dobbiamo inserire il nome della directory che dovrà contenere lo shortcut. Anche qui, come prima, lasciamo il nome predefinito e proseguiamo:

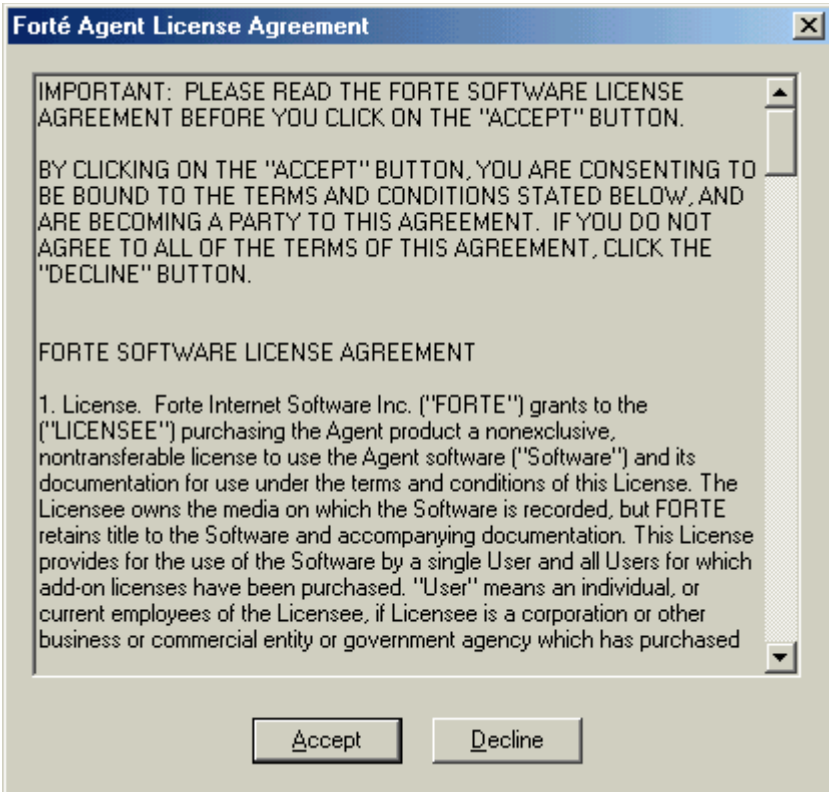


Okay. Tutto sembra essere a posto per iniziare di fatto l'installazione. Se va tutto bene, possiamo cliccare con il tasto sinistro del mouse sul pulsante *Start*, altrimenti torniamo indietro cliccando sul tasto Back. Diamo per scontato che tutto sia a posto e andiamo pure avanti:

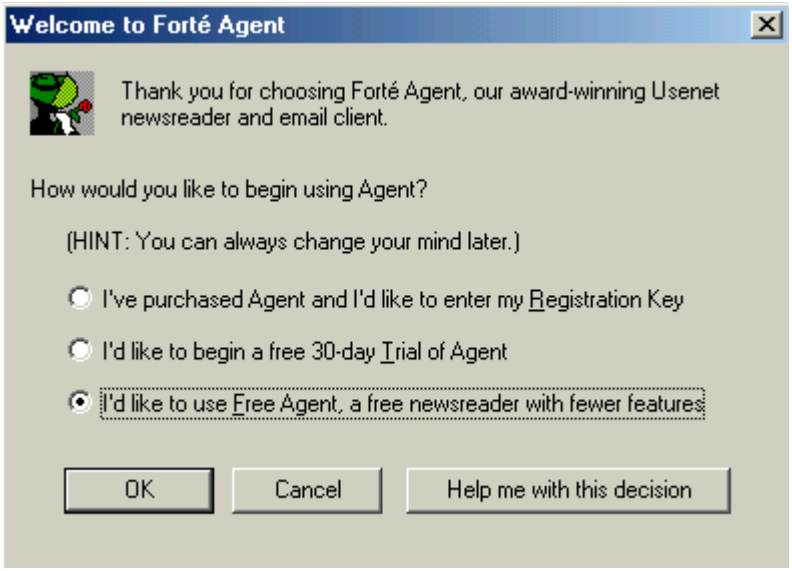


Okay. L'installazione è terminata e a questo punto ci viene chiesto se vogliamo eseguire **Free Agent**. Rispondiamo di no, clicchiamo sul tasto Finish e riavviamo il nostro computer...

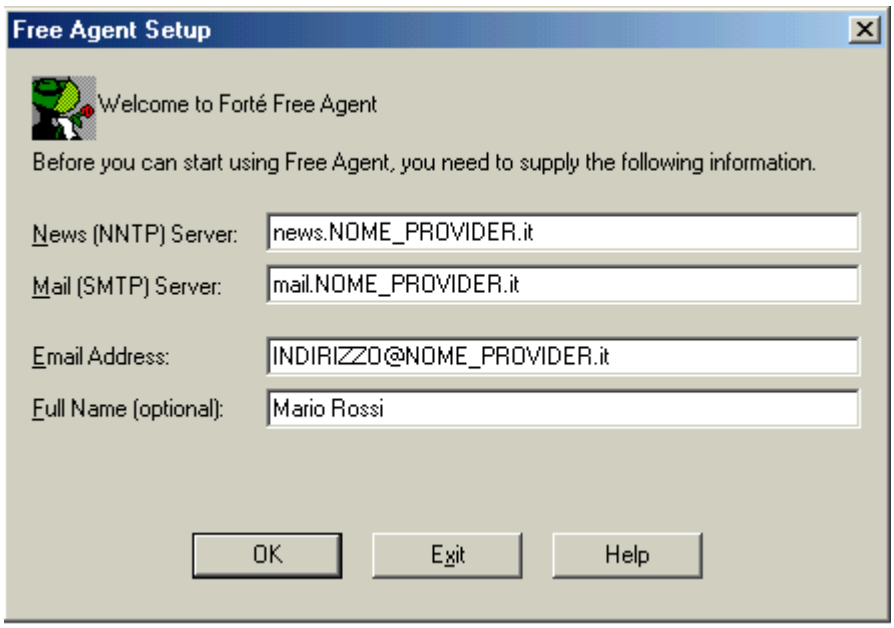
Bene. Ora che Windows è ripartito, possiamo finalmente eseguire Free Agent. Quello che ci appare la prima volta, è l'accordo di licenza che dobbiamo (giustamente) accettare:



Fatto questo, dobbiamo scegliere in quale modalità far partire il programma:

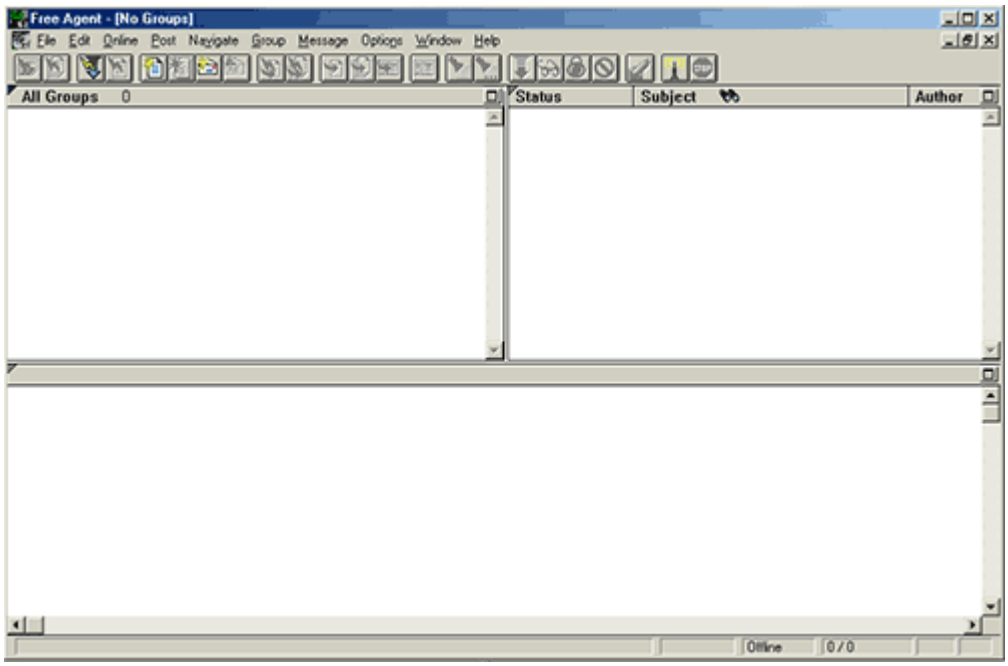


Poiché ci interessa la versione freeware, la terza opzione è quella da selezionare. Infine segue l'inserimento di alcuni parametri che ci vengono forniti dal nostro provider:

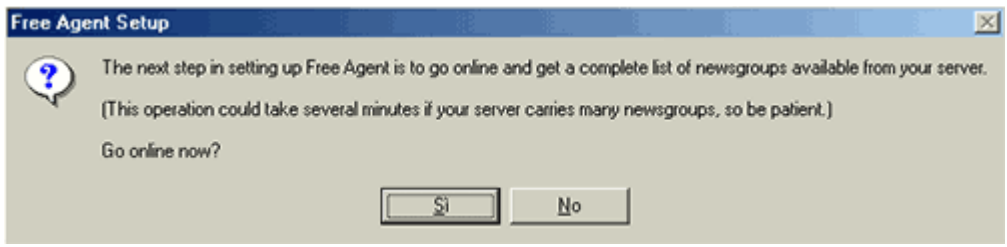


Di tutti questi, occorre notare che il testo che immettiamo nel campo *Full Name* è quello che appare nell'intestazione di tutti i nostri messaggi (fate quindi attenzione a cosa mettete!!!).

Ora è finalmente arrivato il momento di vedere "all'opera" Free Agent. La prima volta che lo eseguiamo, però, più che un luogo pieno di utenti sembra un deserto imbiancato:



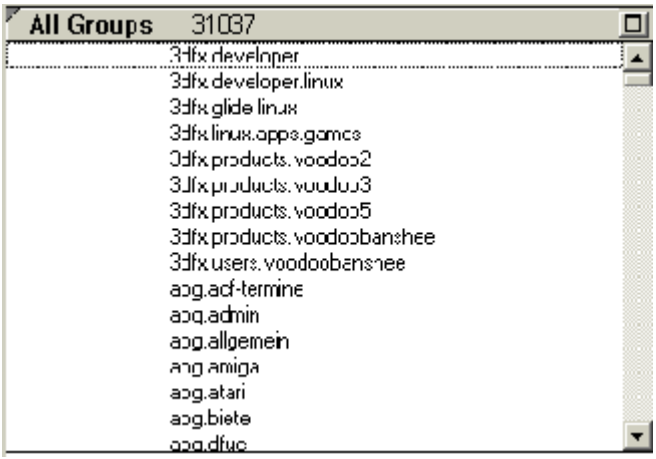
Poi, come per "magia", appare la seguente finestra:



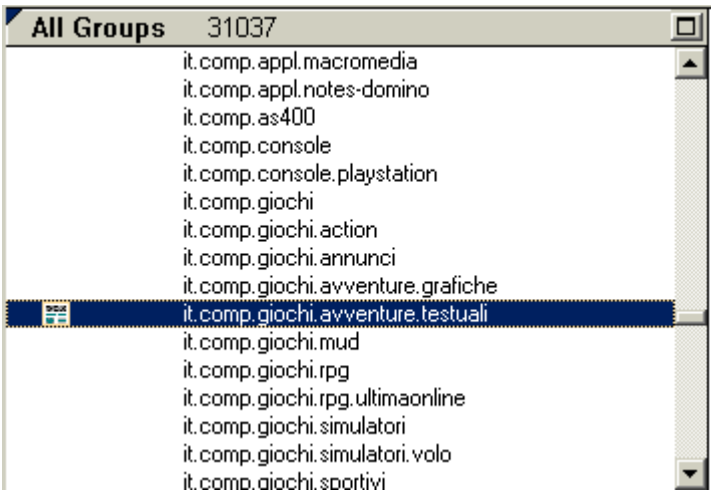
nella quale ci viene chiesto se vogliamo che il programma si colleghi a Internet per reperire

la lista dei newsgroup disponibili[o]. La risposta è ovviamente sì (a patto, naturalmente, di essere collegati a Internet o, per dirla in altri termini, online).

Dopo un po' di tempo (anche alcuni minuti, ma dipende dalla connessione, durante i quali Free Agent sembra bloccato ma naturalmente non lo è), nella prima finestra in alto a sinistra appare il nostro ambito elenco:



Come potete facilmente notare dalla figura, i newsgroup disponibili sono la bellezza di 31037, coprono praticamente ogni tipo di argomento e sono in diverse lingue. Ora, iniziando a scorrere l'elenco, quello che ci interessa è *it.comp.giochi.avventure.testuali* che, una volta trovato, occorre selezionare e sottoscrivere (premendo contemporaneamente i tasti CTRL+S)[o]:



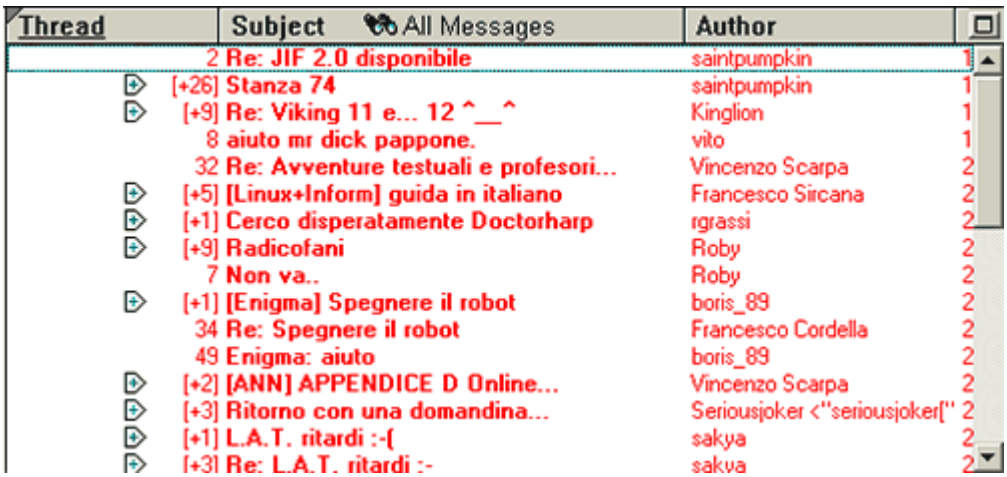
Bene. Ora possiamo accedere ai tanto decantati messaggi degli utenti. Andiamo, con il puntatore del mouse, al menu *Online* e selezioniamo la voce *Get All Headers in Selected Group* (se li vogliamo tutti) o *Get New Headers in Selected Groups* (se vogliamo solo quelli nuovi)[o]. Ad ogni modo, qualunque sia la scelta fatta, ecco quello che appare nella finestra in alto a destra:

---

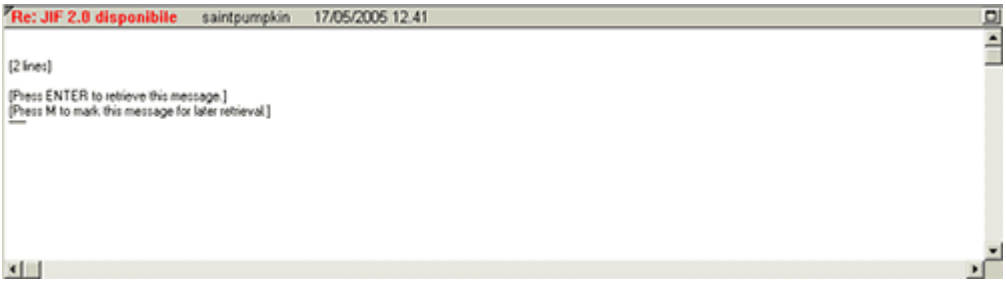
o Questa finestra appare solo la prima volta che si apre Free Agent. Per eseguire la stessa operazione una seconda volta, occorre selezionare la funzione *Get New Groups* nel menu *Online*.

o Una volta che si sono stabiliti i newsgroup da sottoscrivere, è possibile fare in modo che nell'elenco risultino visibili solo loro, andando sulla voce *Show Groups and Folder/Subscribed Groups and Folder* del menu *Group*.

o Questa funzione è anche attivabile cliccando sul secondo pulsante a sinistra della barra degli strumenti.

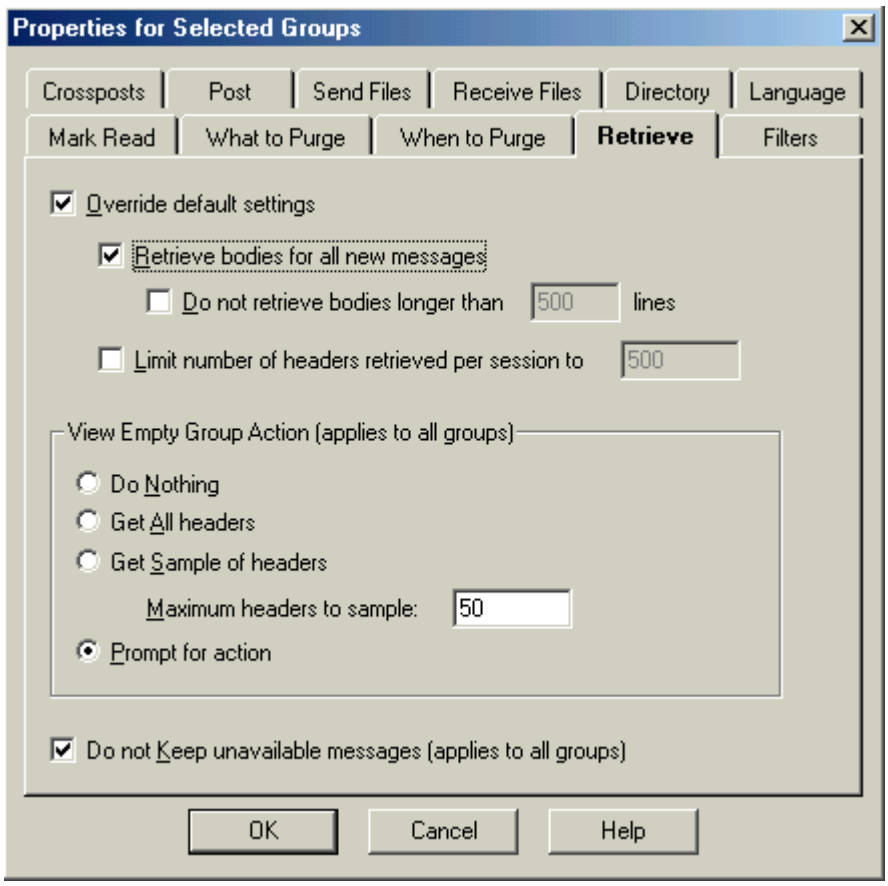


L'elenco dei messaggi è finalmente a nostra disposizione, e se clicchiamo su uno qualsiasi di essi:

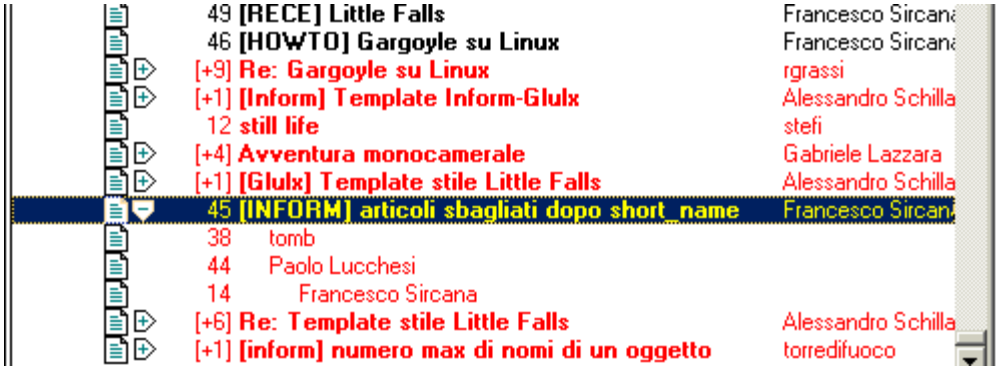


il programma ci informa (nella finestra in basso) che se vogliamo leggerne il contenuto occorre premere il tasto INVIO, perché con le impostazioni predefinite Free Agent scarica da Internet solo l'intestazione dei messaggi:

Questo è molto utile in quei newsgroup che hanno un numero di messaggi giornaliero molto alto. Non avendo il tempo di leggere tutti i messaggi, scarichiamo allora solo quelli che ci interessano di più. Ma in quelli meno frequentati (e purtroppo quello sulle avventure testuali è tra questi) possiamo decidere di scaricare ogni volta i messaggi completi per poterli così leggere comodamente offline (senza, cioè, essere collegati a Internet). Basta selezionare il gruppo interessato nella finestra in alto a sinistra, premere contemporaneamente i tasti ALT+INVIO, e andare alla scheda *Retrieve*:



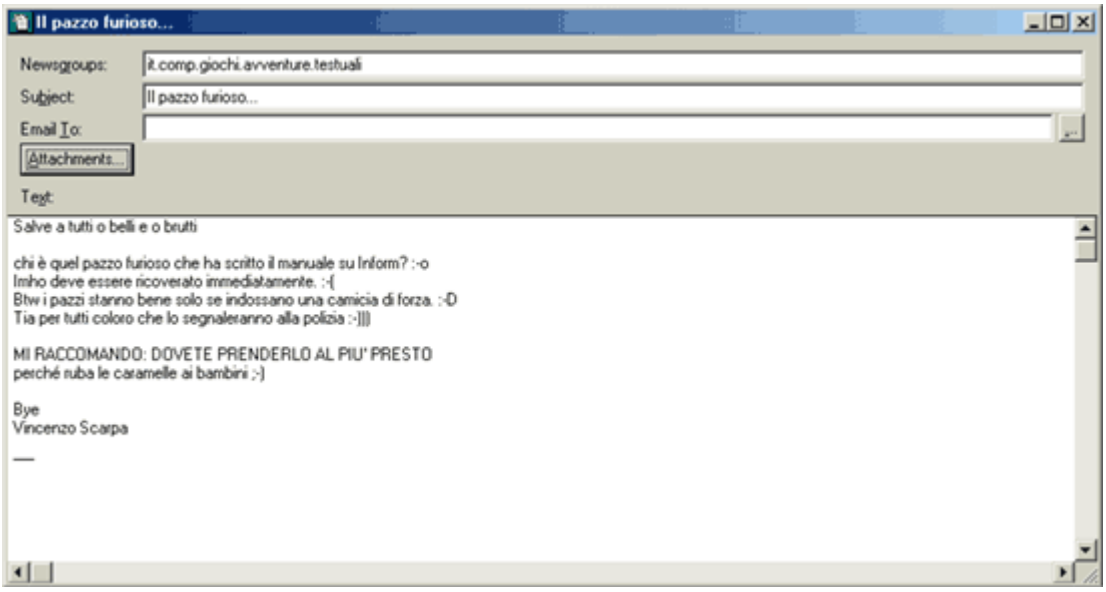
Rendendo attive le prime due voci (che di default non lo sono), scaricando i nuovi messaggi avremo fin da subito sia l'intestazione che il loro contenuto. Occorre ancora notare che esistono sia i messaggi singoli (come quello visto poco fa) che quelli annidati (come mostrato in figura):



Come potete facilmente intuire, tutti i messaggi contrassegnati a sinistra dalla freccetta + hanno sotto di loro degli altri messaggi (le risposte e le risposte alle risposte, che costituisco i cosiddetti "thread"), mentre i messaggi colorati in rosso sono quelli che devono essere ancora letti.

Possiamo inoltre decidere se cancellare i messaggi (con il tasto CANC), tenerli (premendo il tasto K) o se cancellare il loro contenuto (premendo contemporaneamente i tasti CTRL+CANC) per poi magari marcarli (con il tasto M) nel caso decidessimo di rivolere quest'ultimo.

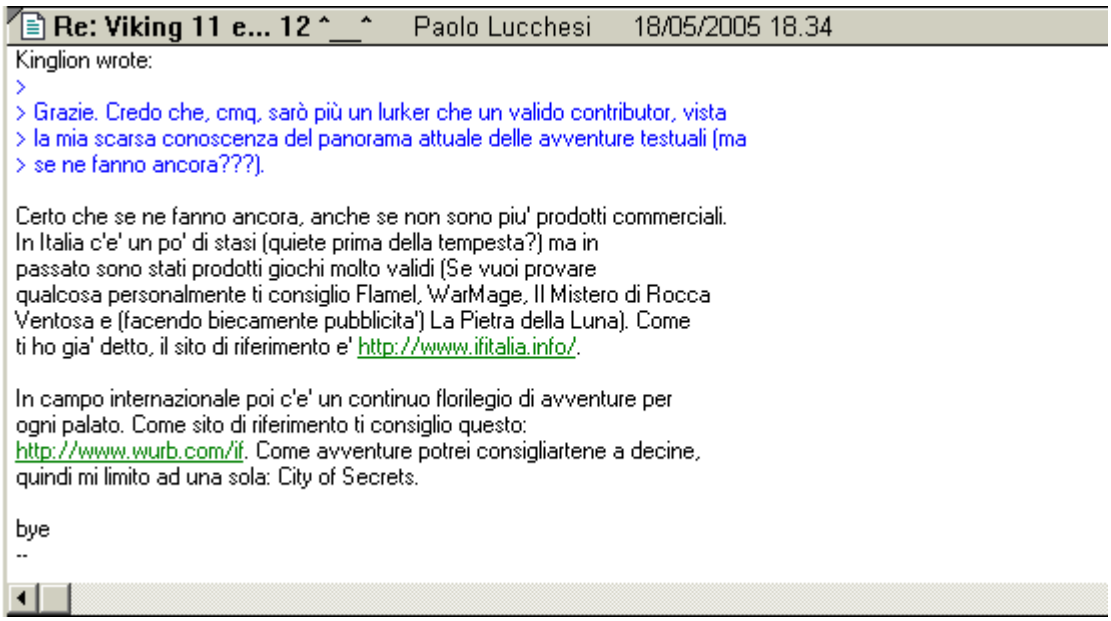
Tutto questo va bene per la lettura. Ma se vogliamo scriverne uno noi? Niente di più facile. Se vogliamo scrivere un nuovo messaggio dobbiamo premere il tasto P, specificare il "subject" (oggetto del messaggio) e scrivere infine il testo vero e proprio:



Ci sono alcune cose da notare: innanzitutto le famose faccine, che esprimono lo stato d'animo di chi scrive[o] (ad esempio `:-)` esprime gioia, `:-(` tristezza e così via), gli acronimi[o] (in questo caso IMHO significa "Secondo il mio modesto parere", "Fra l'altro" per BTW e "Grazie in anticipo" per TIA) e le lettere maiuscole (che equivalgono "all'urlo"). È poi presente anche la funzione di *Attachments* (che permette a chi scrive di allegare, insieme al messaggio, anche uno o più file) che va però usata con molta cautela (chiedendo magari il permesso agli altri utenti del gruppo o — se quest'ultimo è moderato — al moderatore stesso[o]).

Se vogliamo invece rispondere a qualcuno, occorre premere il tasto F:

o Un elenco dettagliato potete trovarle all'indirizzo http://digilander.libero.it/katalog/emo/emoticon.html.
o Un elenco dettagliato potete trovarlo all'indirizzo http://www-utenti.dsc.unibo.it/~cceredi/ig/acronimi.-htm.
o Alcuni newsgroup sono moderati (nel senso che se uno o più partecipanti si comportano in maniera scorretta possono essere ripresi se non addirittura allontanati da una persona apposita — il moderatore). Quello sulle avventure testuali non è tra questi, anche se questo non significa che ognuno al suo interno può fare quello che vuole; occorre infatti cercare di non spedire messaggi non inerenti all'argomento trattato (i cosiddetti off-topic o più semplicemente OT), evitare linguaggi scurrili o comunque altamente offensivi e quoting troppo lunghi. Ricordatevi, inoltre, di usare lo spoiler quando chiedete aiuto per un'avventura che state giocando; dovete, cioè, inserire una serie di puntini posti ognuno su una nuova riga in modo tale che il testo del messaggio non sia immediatamente visibile a tutti gli utenti.



La scritta in blu appartiene al messaggio precedente e rappresenta il cosiddetto “quoting” (che non deve mai essere troppo lungo). Per il resto, occorre ancora notare due cose: OGNI MESSAGGIO È COMPOSTO DA UN’INTESTAZIONE (in questo caso “Kinglion wrote”) E DA UNA FIRMA (”Bye”), e possono essere personalizzate entrambe rispettivamente dalle schede *Introductions* e *Signatures* della voce *Posting Preferences* del menu *Options*.

Se vogliamo invece rispondere a qualcuno tramite e-mail (con la differenza che il messaggio verrà letto da lui/lei soltanto e non da tutti gli utenti del newsgroup), occorre premere il tasto R.

In tutti e tre i casi, comunque, occorre stabilire se inviare il messaggio subito tramite la pressione contemporanea dei tasti CTRL+N (e per questo occorre essere collegati a Internet) o più tardi tramite la pressione contemporanea dei tasti CTRL+L.

Inoltre, È POSSIBILE MONITORARE LO STATO DEI MESSAGGI TRAMITE LA FINESTRA DI OUTBOX (attivabile alla voce *Open Outbox* del menu *Window*), al cui interno vengono elencati tutti i messaggi da noi spediti, contrassegnati da una faccina sorridente (nel caso in cui tutto sia a posto) o triste (se qualcosa non è andato per il verso giusto — in questo caso occorre provare a rispedire il messaggio).

Un altro problema può essere dato dal BACKUP DEI DATI. Come posso, infatti, fare delle copie di salvataggio di tutti i miei messaggi, dei newsgroup sottoscritti e dei dati personali? Semplice: basta copiare la directory `C:\Programmi\Agent\Data` da qualche parte per poi ripristinarla in seguito quando (o se) occorre.

Ricordo, infine, che per quanto riguarda le avventure testuali, oltre al newsgroup italiano citato ne esistono anche altri due: *rec.arts.int-fiction* (dedicato soprattutto alla programmazione vera e propria) e *rec.games.int-fiction* (dedicato ai giochi). Inutile dire che qui si scrive solo in inglese...

Intervista a Marco Vallarino

di Andrea Rezzonico

Marco, raccontaci un po' di te.

Sono nato e vivo a Imperia, città che definisco cordialmente il grande buio, per la totale mancanza di iniziative culturali. Ho ventisette anni ma ne dimostro ventisei e mezzo, dal 2002 collaboro col Secolo XIX di Genova, sia in veste di autore, con racconti ispirati ai principali fatti di cronaca, che di giornalista, con servizi sugli eventi, i locali e i personaggi della Riviera di ponente. Mi piace molto andare a ballare e tirar tardi la notte, anche a casa a leggere, scrivere o a giocare. Le due cose più importanti della vita credo siano i sogni e le sorprese. I primi perché quando riesci a realizzarli sei felice, le seconde perché quando ti capitano ti rendono ancora più felice.

Com'è nata la tua passione per le avventure testuali e, più in generale, per lo scrivere?

Le avventure testuali sono state la prima cosa che abbia letto con interesse nella mia vita, nel 1987, e ovviamente anche la prima cosa che abbia desiderato scrivere. Ho cominciato quasi subito, poche settimane dopo aver giocato la prima (*La valle incantata* di Bonaventura Di Bello, uscita sul numero 4 di **Explorer**). Arrangiandomi col basic del Commodore 64 ho messo insieme alcuni lavori assolutamente orrendi, in gran parte scopiazzati dai giochi di BDB, che sono felice siano andati persi per sempre (in verità mi piacerebbe molto recuperarli, ma senza intenti "bellicosi", solo per curiosità mia).

Quello che mi è sempre piaciuto delle avventure testuali è il continuo stato di tensione cui è sottoposto il giocatore. A differenza di molti altri tipi di videogiochi, in un'avventura non sai mai dove ti porterà l'azione che stai per compiere. E poi di solito si instaura un rapporto più intimo. Un'avventura testuale è un balocco che ti puoi rigirare come vuoi, senza fretta, godendotelo appieno, mappando, appuntando, ragionando e anche tirando pugni sul tavolo (come faccio io di solito). In molti shoot'em up non ricordi neanche che forma avevano le astronavi che hai appena blastato e nei giochi di ruolo hai sempre l'impressione di fare quello che devi (tipo obbedire agli ordini del re) piuttosto che quello che vuoi fare.

Quanto allo scrivere, non ho mai avuto una vera e propria passione per la parola scritta. In realtà quello che mi piace veramente è dare vita a un'idea e in effetti scrivere è uno dei mezzi più immediati per farlo. In tutti i miei racconti c'è almeno un'idea (tipo lo spray al pentotal in *Zombi!* o l'*Unione di Risorse per la Sopravvivenza Sistematica* ne *I migliori anni della tua vita*), non sono mai esercizi di stile. Cerco sempre di curare il più possibile la scrittura, ma più che altro per vanità e gratitudine nei confronti di chi mi leggerà.

Tra le tue tante avventure, Enigma è sicuramente quella che ha avuto il maggior successo: ancora oggi, a quattro anni di distanza dal suo primo rilascio, rende impossibile la vita di molti giocatori. Perché dopo questo grande successo non hai più rilasciato avventure? Hai definitivamente abbandonato l'interactive fiction o prima o poi rilascerai un altro gioco?

Ho pensato spesso a che cosa scrivere dopo *Enigma*. Ho in mente parecchi adattamenti che mi sembrano interessanti, ma ancora pochi puzzle che renderebbero avvincente un'av-

ventura, in qualunque ambientazione si svolgesse. Non ho assolutamente rinunciato all'idea di scrivere avventure, anche se ormai sono, mio malgrado, più orientato ai racconti e ai romanzi non interattivi. Mi piacerebbe tantissimo uscire con un'avventura nuova, ma devo ancora trovare il materiale giusto.

Non comincio mai un racconto senza almeno una buona idea, così come non comincio un'avventura senza una manciata di puzzle abbastanza interessanti (e magari originali). L'aspetto dell'interactive fiction più legato alla trama mi interessa poco, anche quando gioco preferisco sempre cercare dell'esplosivo per far saltare una porta piuttosto che parlare con qualcuno. Per come la vedo io, i personaggi nelle avventure devono servire a recuperare oggetti nuovi o a fare cose per noi impossibili, tipo il buon vecchio Floyd in *Planetfall*.

Nelle ultime settimane avevo così tanta voglia di scrivere un'avventura nuova che ho ricominciato a giocare a tutte le vecchie avventure di Bonaventura Di Bello, molto ricche di spunti interessanti, per farmi venire delle idee nuove. Qualcosa è uscito, ma ancora poco per rimettermi al lavoro.

A proposito di avventure difficili, qual è stata l'avventura con cui hai incontrato più difficoltà?

Negli ultimi tempi *Progetto Atlantide* di Bonaventura Di Bello, uscita nel 1987 sul numero 6 di **Viking**. L'ho rigiocata qualche settimana fa e mi ha messo davvero in crisi. Ambientata su un'isola del Pacifico occupata dai militari di una potenza straniera, ti mette nei panni del super agente segreto Dick Ironside, incaricato di liberare uno scienziato rapito e... recuperare un sottomarino rubato! All'inizio sembra tutto facile, poi non ne vieni più a capo, almeno finché non capisci che i risultati di certe azioni cambiano completamente a seconda dei momenti in cui le compi, cosa assai strana e innovativa per l'epoca in cui era uscito il gioco.

In assoluto, come tutti sanno, l'avventura più difficile cui abbia giocato è la vastissima *Acheton* della **Topologika**. Oltre 400 locazioni da mappare (accuratamente), 55 tesori da trovare e una torcia elettrica che si può scaricare da un momento all'altro e può essere ricaricata solo una volta, nella mitica *Timeless Cavern*. Ricordo bene la notte in cui l'ho finito. Era sabato 29 settembre 2002, ero appena tornato (presto) dalla discoteca e, intorno alle due di mattina, avevo deciso di provare a fare un altro giretto per le caverne di Acheton. Ho finito alle sette di mattina, a pezzi. Senza gli help sheet e i consigli di Adam Atkinson non ce l'avrei mai fatta, ma arrivare nella *Gladiators' arena* e vincere tutti i combattimenti è stata una delle più grandi soddisfazioni della mia carriera di avventuriero!

Enigma era scritta originariamente con il Modulo Base di Enrico Colombini: quali sono state le difficoltà incontrate nel convertire l'avventura in Inform? Hai mai pensato, inoltre, di utilizzare linguaggi diversi da Inform, anche per progetti diversi dall'interactive fiction?

Quando ho convertito Enigma in Inform, alla fine del 2002, da una parte avevo voglia di imparare un linguaggio nuovo, dall'altra volevo mettere a tacere i parrucconi che parlavano male del gioco solo perché era scritto in basic. Le difficoltà incontrate, dovute alla mia conoscenza sommaria e lacunosa del linguaggio, sono state felicemente risolte grazie alla disponibilità di veri e propri guru di Inform come Tommaso Caldarola e Giancarlo Niccolai. Il gioco è rimasto tale e quale, a parte i puzzle nuovi che mi sono venuti in mente strada facendo e che ho inserito a partire dalla versione in Inform. Non credo che il linguaggio scelto per realizzarlo possa influire poi tanto sul gioco. L'importante è sapere dove si vuole an-

dare a parare.

Quanto agli altri linguaggi e ai progetti diversi dall'interactive fiction, mi piacerebbe molto scrivere un gdr con Rpgmaker, ma non ho mai trovato il tempo e la voglia di impararlo come si deve, né di pensare a una storia su cui poi costruire il gioco. Questo è uno dei grandi problemi della mia vita: scrivo — digito più esattamente — un casino (racconti, avventure, articoli, programmi) perché le altre forme di espressione (disegnare, suonare) mi sembrano sempre troppo complicate. Per come la vedo io, tra programmare un'avventura in Inform e realizzare un gioco di ruolo con RPGMaker passa la stessa differenza che c'è tra accendere la luce e programmare un videoregistratore. Non molta, ma abbastanza da farti pensare che forse non ne vale la pena.

L'interactive fiction italiana sta attraversando un periodo di crisi: poche sono le avventure rilasciate (per lo più da autori emergenti, mentre i più conosciuti sembrano essere scomparsi) e scarsa la partecipazione alle competizioni, mentre si sviluppano invece applicazioni per la creazione delle avventure. Come spieghi questa situazione? Forse è più facile dedicarsi ad uno strumento di sviluppo che pensare ad una trama e a degli enigmi?

Questo sicuramente. In giro poi ci sono un sacco di autori bislacchi che spacciano mediocri esercizi di stile, privi della benché minima idea, per racconti o romanzi. Io non sono d'accordo sul rifilare della roba da leggere alla gente, per me bisogna dare storie e, quando va bene, emozioni. Che poi magari facciano pensare, ma questo è un altro paio di maniche e io al momento sono in canottiera!

Tornando alle avventure, per me la crisi dell'interactive fiction italiana è dovuta ad almeno due motivi. Il primo è che forse ci si è resi conto che è molto difficile trovare il materiale per scrivere una buona avventura, e per questo si è rinunciato a "fare presenza" con idee così così, peggiorate magari da realizzazioni frettolose. La seconda è che, forse, anche autori che si sono impegnati anima e corpo nei propri progetti non hanno ottenuto il riscontro sperato.

La comunità di autori e giocatori che bazzica il newsgroup *it.comp.giochi.avventure.testuali* non può certo pretendere di essere autosufficiente, ci sarà sempre un disperato bisogno di tirare dentro forze nuove, che da una parte motivino gli eventuali sforzi realizzativi e dall'altra, magari, portino idee nuove.

Sul tuo sito web affermi che la tua avventura preferita è Acheton della Topologika, un'avventura molto grande che richiede un lavoro certosino per essere portata a termine. Credi che oggi ci sia ancora spazio per un prodotto del genere, o siamo destinati ad un futuro fatto di avventure brevi e di facile risoluzione?

Il sogno della mia vita di autore, più che scrivere un grande romanzo tipo *Moby Dick*, è proprio di realizzare un'avventura che possa essere considerata la *Acheton* italiana. Una caccia al tesoro sterminata, un apologo della fantasia più sfrenata, vasto, imprevedibile, in cui si possa trovare qualunque cosa. Ovviamente, dovessi mai trovare la capacità, temo che mi mancherebbe il tempo.

Scrivere avventure brevi aiuta gli autori a finirle e a prendere consapevolezza dei propri mezzi. Personalmente non sono contrario, l'importante è che l'impianto, anche se limitato, sia sfruttato a dovere con tanti bei puzzle da risolvere, come nelle one room game, genere che però non amo perché per me la cosa più importante in un'avventura, da giocatore, è

sempre la mappa.

Secondo te, qual è il vero limite di un genere come l'interactive fiction? Spero che gli amici del newsgroup mi perdoneranno, ma per me il limite dell'interactive fiction è rappresentato dalla volontà di volerlo rendere qualcosa di diverso da quello che è in realtà e cioè un gioco. Se vogliamo raccontare una storia, scriviamo un racconto o un romanzo, anziché un'avventura. Non era quello lo spirito con cui Crowther e Woods scrissero *Advent*.

Le avventure si leggono, certo, e devono essere scritte decentemente, ma soprattutto si giocano! Io ho pubblicato più di sessanta racconti, ma — pensa un po'! — sono anche contrario a inserire grandi quantità di testo in un'avventura. Il ritmo è fondamentale e poi deve rimanere un po' di mistero, non si può sempre raccontare tutto di tutti. Chi era il GrandE PinaZ, perché è chiuso in manicomio? Chi se ne frega, pensiamo a trovare MaM piuttosto!

Cosa pensi delle avventure e degli autori che popolano oggi la scena italiana? Cosa consiglieresti a chi vuole iniziare a scrivere avventure? E a chi ha già rilasciato qualche opera?

Confesso di non seguire molto la scena italiana. Non tanto per pigrizia o sfiducia, quanto per disinteresse, nel senso che, dopo *Uno zombie a Deadville*, *Flamel* e *Il mistero di Rocca Ventosa*, non sono più uscite avventure che mi ispirassero particolarmente. Mi sono divertito a giocare *Red Hill* e *Schizo*, ma è stato quasi per caso. Aspetto sempre un horror abbastanza forte, lontano dalle parodie, oppure un fantasy vecchio stile, con una bella caccia al tesoro e tante locazioni da mappare.

Per scrivere avventure la cosa che serve di più credo sia l'esperienza. Più avventure si giocano e più cose vengono in mente mentre se ne scrive una. A me sono servite tantissimo quelle bizzarre e ingegnose della Topologika: *Coundown to Doom*, *Kingdom of Hamil*, *Philosopher's Quest* e ovviamente *Acheton*. E poi anche quelle semplici ma efficaci e ineccepibili scritte da Bonaventura Di Bello per Explorer e Viking. Per il fantasy *La valle incantata*, *La legge di Thorus*, *Il segreto di Obnyr* e *L'ombra di Garmon*. Per la fantascienza *Star Plague*, *Incubo*, *Naufragio su Guyoth*, *Metalcrunchers*, *Prigioniero del futuro*. Per l'avventura *L'occhio del Condor*, *Nessuna notizia dal campo base*, *L'isola della paura*, *Bucaneer*, *Oregon*. Per il mistero *Terrore al castello*, *I misteri di Villa Parson*, *L'orrore di Providence*. E anche avventure anomale come *Green dimension*, in cui si deve ritrovare la fidanzata perduta, *Homo sapiens*, ambientata nella preistoria, o *Easy Rider*, in cui si impersona un hippy alla ricerca della terra promessa.

Ovviamente anche chi è già autore di un'avventura non deve mai smettere di giocare e pensare a nuove idee e ambientazioni. Immagino serva anche tenersi aggiornato sulle produzioni degli altri, anche se purtroppo, come detto prima, ho un po' smesso di giocare alle nuove avventure.

Il futuro ci riserva senz'altro tante nuove e belle avventure da giocare. Ricordiamoci però che l'interactive fiction sta per festeggiare i suoi primi trent'anni di vita (ufficiale) e non è detto che le avventure vecchie non siano più buone, anzi.

Recensioni: “9:05”

di Roberto Grassi

9:05, di Adam Cadre, è un piccolo pezzo d’Interactive Fiction che, a mio parere, risulta uno dei più sopravvalutati. Leggo dalla presentazione su Baf’s Guide:

> *“A very short game with a devious twist that the IF theorists out there will find interesting. Gains immeasurably on replay.”*

Sono d’accordo sul ‘twist’ ma non credo che 9:05 possa avere qualche impatto particolare sulla teoria dell’IF. Il ‘twist’ su cui si basa è “scorretto” (non posso dire di più altrimenti annullerei la gioia del gioco) e basato sulla completa discrasia tra giocatore e PG (quello, cioè, che il giocatore controlla).

Mi sembra anche che non abbia un grosso valore di ‘replay’ e certamente non incommensurabile. Le uniche cose che richiederebbero ‘replay’ sono due indizi da trovare all’inizio del gioco e provare i tre finali. Per quello che riguarda gli indizi iniziali, ne avevo trovato uno subito e quindi mi ero perso gran parte della necessità di rigiocare.

I tre finali, invece, sono alquanto deludenti:

- Il primo, che giustifica il ‘twist’ di cui si parlava all’inizio, si basa sulla completa discrasia fra giocatore e PG, rompe la mimesi ed è difficilmente giustificabile in modo logico, a meno di non parlare di stupidità del personaggio (viene infatti giustificato in questo modo, ma questo non basta a redimerlo).
- Il secondo finale è una ’sudden death’, una morte improvvisa, e come tale non ha un valore particolare.
- Il terzo finale, quello ‘positivo’, è raggiungibile unicamente per tentativi. In nessun punto del gioco è suggerito perché dovrei - o potrei - raggiungerlo.

Cadre dichiara di aver scritto il gioco in 4 giorni, e si vede. Credo che vada considerato per quello che è. Un gioco carino da risolvere in una quindicina di minuti, che dovrebbe portarvi ad un finale ‘a sorpresa’ e spingervi a rigiocarlo per provare le altre due strade. Nulla che sia particolarmente importante per la teoria e/o la pratica dell’IF. Niente di più, niente di meno.

Recensioni: “Aayela”

di Roberto Grassi

Aayela, di Magnus Olsson, è un piccolo gioco d’IF del 1996, di genere fantasy.

La trama è abbastanza semplice e banale. La regina si è ammalata di una malattia apparentemente incurabile. Un medico di corte ricorda un’antica leggenda che parla di un cristallo che racchiude una luce magica in grado di guarire.

Naturalmente, viene dato poco credito a questa leggenda ed è questo il motivo per cui, per affrontare questa impresa, veniamo scelti noi, che siamo un cavaliere giovane e di belle speranze. In altre parole… ’sacrificabile’.

Consiglio di giocarlo per due motivi: i finali multipli, facilmente e intuitivamente raggiungibili, ed un’interessante gestione del movimento del giocatore al buio.

Questo secondo aspetto è davvero peculiare. Ho visto e giocato pochi giochi d’IF che consentano un’esplorazione ‘al buio’ basata su altri sensi che non siano quello della vista. Aayela, al contrario, vi consente di esplorare locazioni al buio e di toccare, annusare o ascoltare per rintracciare la giusta strada e identificare e manipolare oggetti utili.

Recensioni: “In the end”

di Roberto Grassi

In the end è un’avventura di Joe Mason del 1996, caratterizzata dal fatto di essere un gioco ‘puzzleless’ e di tentare questa strada qualche anno prima di Photopia.

È pervaso da un forte senso di malinconia: il protagonista, infatti, si trova ad assistere al funerale di un proprio amico e si interroga sul senso dell’esistenza. Uscendo dalla chiesa, potrà interagire con alcuni personaggi e visitare alcune locazioni, ma in nessuna di queste ci sarà un vero e proprio puzzle.

La storia ha un solo finale, che è difficile da intuire perché non viene ’suggerito’ al giocatore durante il gioco. Ragionando a posteriori, tuttavia, sembra che non ne possa avere degli altri. Niente di speciale, tuttavia lo consiglio se volete provare qualcosa ‘puzzleless’ e se vi piacciono le atmosfere ‘tristi’.

Recensioni: “Whom the telling changed”

di Roberto Grassi

Whom the telling changed di Aaron Reed, è uno dei migliori giochi d’IF che abbia giocato ultimamente. Il design e la giocabilità sono molto curati e la modalità d’interazione è ridotta al minimo ma molto efficace.

La cosa più significativa è che ci si lascia trasportare dal flusso della narrazione e ritengo che, in definitiva, sia questo l’aspetto più importante per un gioco d’Interactive Fiction.

Nel gioco, impersoniamo una giovane donna facente parte di una tribù che deve fronteggiare la minaccia di un’imminente invasione. Come ogni racconto che si rispetti, c’è qualcuno che si dichiara apertamente nostro nemico e rifiuta la nostra politica ‘attendista’. Molte aspettative sono riposte nel rito dell’interpretazione dei vecchi miti raccontati da un’anziana donna che ha poteri divinatori e il dovere di mantenere viva la tradizione orale.

Intorno al fuoco dell’accampamento si svolge quindi una sottile guerra dialettica basata sull’interpretazione di quanto dice l’oracolo nel tentativo di convincere gli astanti e far loro decidere se stare con noi (politica attendista) o contro di noi (politica interventista). In base alle risposte che diamo, tuttavia, la trama può cambiare considerevolmente ed equilibri e assensi consolidati possono saltare.

A mio modo di vedere, il gioco ha vinto meritatamente la Spring Thing 2005 ed è senza dubbio uno dei giochi d’IF più coinvolgenti che io abbia giocato recentemente. Merita senza dubbio di essere giocato.

Design Review: “Fear”

di Roberto Grassi

Benvenuti in questo primo numero di una rubrica che intende soffermarsi sugli aspetti di design di un gioco d’IF. Mi auguro che possa essere utile agli autori, per migliorare e raffinare il processo di design di un gioco d’IF, e ai giocatori per apprezzare maggiormente alcune finezze d’implementazione. Il gioco in esame è Fear, del 1996. Iniziamo l’analisi.

All’avvio del gioco compare il seguente testo:

> You are running for your life down dark, labyrinthine corridors, your heart pounding almost as loudly as the heavy boots of your relentless pursuer. But your legs are collapsing under you, your breaths coming in ragged gasps. At the last, strength fails you and you collapse face-down upon the unforgiving concrete. Cold hands grasp your neck, hauling you upright, forcing your unwilling eyes open to gaze into the hard, cruel, familiar face of your captor - and you scream with the horrible recognition that those twisted features are your own.
>
> You are still screaming when you awaken in sweat-drenched clothes, deeply relieved that it was only a dream. Yet something is not right. Why does the darkness beyond the window look so threatening? Why do the sounds of the night bear such menace? Indeed, why are you in such constant...
>
> **FEAR**
> An Interactive Nightmare For information, type “about” Release 1 / Serial number 961012 / Inform v1502 Library 5/12 Standard interpreter 1.1

Mi sembra un buon inizio dal punto di vista formale. Le due porzioni di testo ci parlano di un incubo che abbiamo avuto e del nostro risveglio agitato. La riga di spazio separa la descrizione dell’incubo dal nostro risveglio, facendo capire che c’è uno stacco ‘logico’ tra i due paragrafi. L’ultimo spazio, invece, ci porta alle informazioni sul titolo del gioco, dandoci anche il comando per avere maggiori informazioni.

Anche la narrazione mi sembra ben curata anche se non sono sicuro che la frase: *‘You are still screaming when you awaken in sweat-drenched clothes, deeply relieved that it was only a dream.’* abbia i tempi al posto giusto. Al di là di questi aspetti, arriviamo alla parte di interazione e cerchiamo di capire cosa ci è richiesto di fare.

I primi indizi sono in queste frasi: *‘Yet something is not right. Why does the darkness beyond the window look so threatening? Why do the sounds of the night bear such menace? Indeed, why are you in such constant... FEAR?’*. Abbiamo fatto un sogno da cui ci siamo risvegliati bruscamente e quello che dobbiamo cercare di capire è perché abbiamo questa sensazione di paura. Molto probabilmente ci aspetta un gioco in cui dobbiamo trovare alcuni oggetti o persone che rievocheranno alcuni ricordi o emozioni che ci faranno capire i motivi di questo nostro stato di inquietudine.

En passant, segnalo questa piccola incongruenza. *‘Why does the darkness beyond the window look so threatening?’* Come facciamo a sapere che c’è una ‘darkness’ al di là della fine-

stra se ancora non abbiamo guardato fuori?

Fatte queste prime analisi, procediamo:

> **Bedroom (on the bed)**
> This first-floor room is almost uncomfortably warm and well-lit. The bed sits against the east wall, and curtains cover the windows to the south. A calendar hangs from a hook on one wall, and the exit is to the north. On the bedside table are a pillbox (which is closed) (in which are nine white pills), a small leaflet and a telephone.

Ci troviamo dunque in una stanza da letto, presumibilmente la nostra, sul letto. Infatti ci siamo appena risvegliati dall'incubo. Le frasi successive fanno riflettere: *'This first-floor room is almost uncomfortably warm and well-lit.'*. Si tratta di una stanza al primo piano. Come facciamo a saperlo? E' la NOSTRA casa? Dunque, il personaggio ha qualche conoscenza dell'ambiente in cui si trova. Se è la nostra casa perchè la nostra stanza da letto dovrebbe essere *'almost uncomfortably warm'* ma *'well-lit'*? Abbiamo curato bene la illuminazione ma non la temperatura della camera? Non vi sembra una incongruenza? Può darsi che sia spiegato successivamente nel gioco? Teniamolo presente.

Come vediamo, poi, c'è una descrizione degli oggetti presenti nella stanza. Una descrizione scarna, ma, a mio parere efficace. C'è sicuramente qualche considerazione da fare sugli oggetti con cui è, verosimilmente, richiesta una interazione immediata. In particolare uno, cioè la *'pillbox'*. Come vediamo dalla descrizione, viene detto che è chiusa ma ci sono dentro nove pillole. Come facciamo a conoscere questo dettaglio? Teniamo presente questa cosa, probabilmente l'autore ci risponderà in qualche modo. Iniziamo ad interagire. Una delle prime mosse da fare è "x me", cioè esaminarCI per capire chi siamo e come siamo fatti.

> >x me
> As good-looking as ever.

Ahia. *'Good-looking?'*, ma non ci eravamo svegliati in preda al panico? Il gioco scrive poi il testo seguente.

> You try to avoid looking out into the dark night.
> You feel suddenly claustrophobic as you hear a rustling nearby. What lurks in the shadows, waiting to pounce?

Il gioco ci dice che cerchiamo di evitare di guardare fuori, dalla finestra, presumo, nella notte. Se questo suggerimento è collegato alla considerazione che facevo prima relativa al problema di come facessimo a sapere della *'darkness beyond the window'* allora può essere un punto di merito. L'autore si è reso conto di questa 'debolezza' e ha cercato di porre rimedio in qualche modo. Proviamo a guardare fuori dalla finestra, allora.

> >look out into the night
> I only understood you as far as wanting to look.

Non è un errore grave ma è da segnare. In teoria bisognerebbe prevedere l'accettazione di un'azione che ho appena suggerito. Se il gioco mi dice che evito di *'look out into the night'* dovrebbe accettare e comprendere ed eseguire l'azione. Immagino allora di dover guardare attraverso la finestra.

> >look through window
> Beyond the window is nothing but the cold, and the dark, and the unknown.

> You can’t bring yourself to look through.
>
> The house is strangely unfamiliar, as if you’d read about it in a book but never actually set foot inside before.

Questo è un punto di merito. Mi aspettavo che dicesse che non poteva guardare fuori dalla finestra perchè c’erano le *‘curtains cover the windows’*. L’autore ha invece agevolato il gameplay prevedendo il fatto di poter guardare fuori e, automaticamente, spostare le tende.

Mi sembra che come primo numero possa bastare qui. Nel prossimo numero proseguiremo con l’analisi del design di questo gioco.

Alla prossima.

INFORM: MFS, parte seconda

di Giancarlo Niccolai

La prima parte di questo articolo è stata pubblicata sul numero 5 di Terra d'IF

Combattimenti.

Ecco una delle due "cose grosse" della MFS. Del resto, la libreria si chiama *Magic And Fight System* per qualche motivo. Il sistema di combattimento delle MFS si basa su un semplice e comprensibile sistema di statistiche in stile gioco di ruolo, e permette di realizzare scene di combattimento che vedono impegnati due o più combattenti, e addirittura due o più schieramenti opposti formati ognuno da un numero qualsiasi di combattenti. È importante comprenderne il funzionamento, perché su di esso si basa il sistema di magia orientato al GDR; le MFS supportano anche un sistema di magia semplificato senza sistema di combattimento (alla *Infocom*, per intenderci), ma se si desidera poter usare la magia in combattimento, allora si pensi alla magia MFS come a un'aggiunta a questo modulo.

Il combattimento non richiede la presenza del giocatore; i personaggi non giocanti possono affrontarsi e risolvere il combattimento anche se il giocatore non è presente. Il giocatore potrebbe assistere da lontano al combattimento, e vederne l'andamento e l'esito, oppure potrebbe ignorare completamente quello che sta accadendo.

Il modulo *MFS_War.h* (`Include "MFS_War"`) definisce le seguenti classi:

Warrior — personaggio in grado di partecipare ai combattimenti;
PC_Warrior — adattamento del guerriero al personaggio giocante;
NPC_Warrior — supporto NPC per il guerriero;
Weapon — arma utilizzabile dai guerrieri per l'attacco e la difesa;
ClothingArmor — armatura che può essere indossata.

Inoltre, alcune proprietà assumono una rilevanza speciale, soprattutto nella grammatica. In particolare:

damage — gli oggetti dotati di questa proprietà possono essere bersaglio degli attacchi in stile GDR. Se colpiti, subiscono un danno attraverso la chiamata alla routine *damage(value)*;
filter_source(source_type, sender) — gli oggetti dotati di questa proprietà possono essere immuni a certi tipi di attacco: se la proprietà torna *rfalse*, l'attacco viene considerato inefficace. Il valore di *source_type* è il tipo di danno (si veda *MFS_Damage.h*), mentre il *sender* è l'oggetto che sta causando il danno;
attack, defense, armor, initiative, base_damage — bonus che un qualsiasi oggetto fornisce a varie caratteristiche di combattimento. Se un oggetto con queste caratteristiche si trova posseduto a qualche livello da un personaggio, i punteggi di ognuna di queste proprietà si sommano;
giving_bonus — se un oggetto avente una o più proprietà fra *attack*, *defense*, *initiative*, *armor* e *base_damage* ha questa routine, il bonus viene sommato solo se questa routine torna vero.

Il combattimento si svolge con le seguenti modalità: tutti i personaggi coinvolti "tirano" un dado virtuale che torna un valore (equidistribuito) tra uno e dieci; a questo sommano il

loro punteggio di iniziativa, che viene calcolato sommando tutti gli oggetti che forniscono in quel momento il valore di *initiative* (inclusi loro stessi); i pareggi non sono ammessi (si procede ad uno spareggio). Si stabilisce così una graduatoria che stabilisce l'ordine di gioco. Ogni giocatore dichiara un bersaglio che intende colpire quando arriva il proprio turno; si procede quindi secondo l'ordine stabilito sopra.

Il personaggio con iniziativa maggiore lancia un dado **1d10** e lo somma al punteggio *attack* ottenuto sommando tutto ciò che fornisce questa proprietà che si trova in suo possesso (incluso se stesso). Il personaggio che subisce l'attacco tira **1d10** e somma al suo punteggio di *defense*. Se il tiro di attacco (più il bonus) è maggiore del tiro di difesa (più il bonus), il colpo va a segno, altrimenti niente di fatto.

Se il bersaglio è colpito, l'attaccante somma tutte le sue fonti di *base_damage*, *min_damage* e *max_damage* come fatto in precedenza per le altre statistiche. Il danno inflitto è pari al totale dei *base_damage* più un numero equidistribuito fra *min_damage* e *max_damage* (a caso).

La vittima sottrae da questo punteggio un valore pari al suo totale della proprietà *armor*; se il danno è ancora positivo (maggiore di zero), viene chiamata la proprietà *damage()*; questa può chiamare *filter_source* per personaggi speciali, impedendo completamente la possibilità di fare danni a seconda del tipo di arma (o della fonte dell'attacco).

Tutto ciò che rimane viene sottratto dal punteggio di *health* (salute) del personaggio. Se la sua salute scende a zero o meno, il personaggio muore, e viene chiamata la procedura *dead_proc*. Nel caso della classe **PC*Warrior***, la *deadproc* di default imposta il *deadflag* a 1 (gioco finito).

La sequenza si ripete per tutti i personaggi iscritti ad attaccare in quel turno; se un personaggio muore prima di poter attaccare, perde il diritto all'attacco. Stessa cosa se il personaggio, che un attaccante ha designato come bersaglio, muore prima che l'attaccante possa colpirlo; non è possibile cambiare bersaglio a turno iniziato.

Queste sono le altre proprietà interessanti dell'oggetto *warrior*:

- **alliance** — è un numero che indica a quale alleanza appartiene il personaggio. Personaggi con lo stesso valore di *alliance* sono "alleati" e non combatteranno fra di loro. L'alleanza del giocatore è zero di default;
- **max_health** — salute massima del personaggio: la salute non può salire al di sopra di questo livello;
- **health** — livello attuale di salute. Se scende a zero, il personaggio muore;
- **verbosity** — livello di descrittività delle statistiche; vedi sotto;
- **hands_name** — "nome delle mani nude"; quando un oggetto di classe *warrior* attacca un oggetto senza usare un arma, viene usata questa stringa o routine per visualizzare il nome del suo attacco;
- **base_damage, attack, defense, armor, initiative, min_damage, max_damage** — valori di base delle statistiche del personaggio quando combatte senza altri bonus ("a mani nude")

Oltre a questo, il modulo definisce l'attributo **aggressive**: se un personaggio possiede questo attributo, attaccherà immediatamente qualsiasi Warrior con alleanza diversa dalla propria.

Infine, la *MFS_Grammar* definisce alcune estensioni particolari in presenza del sistema *MFS_War*. L'azione "attacca" viene ridefinita; se l'oggetto bersaglio fornisce una proprietà *damage*, l'azione diventa un *AttackDamageable* e viene inviata alla routine *before* dell'oggetto bersaglio. In caso contrario, nulla cambia e il verbo "attacca" continua a generare l'a-

zione *Attack* da gestire nella proprietà *life* dell'oggetto bersaglio. In poche parole, il sistema *MFS_War* coesiste con il sistema della libreria standard e viene "invocato" solo se l'oggetto bersaglio dell'azione dispone di una routine *damage*.

La grammatica definisce anche le azioni *Draw* e *Undraw* per i verbi "sguaina/sfodera" e "infodera" applicati su oggetti di classe *weapon*. Queste azioni sono inviate alla *before* della *Weapon* in questione; una weapon "sfoderata" è pronta ad agire e somma i propri bonus ai punteggi del giocatore; altrimenti, pur essendo nell'inventario del giocatore, non lo aiuta in alcun modo.

La classe *Weapon* presenta le seguenti caratteristiche (oltre a quelle destinate ai bonus di combattimento):

- **drawn** — 1 se sguainata, 0 se riposta.
- ***dmg_type*** — tipo di danno inflitto; in genere *DMG_CUTWEAPON*.

I personaggi possono intercettare *Draw* e *Undraw* nella loro *react_before* per reagire ad un comportamento che sembra ostile (o amichevole) da parte del giocatore.

La classe *ClothingArmor* si presenta in modo simile alla Weapon; l'unica differenza è che forniscono il loro bonus solo se indossate (attributo *worn*), e possiedono una proprietà type che può assumere uno di questi valori:

- ***MVSW_HELMET*** — Elmo
- **MVSW_ARMOR** — Armatura
- **MVSW_SHIELD** — Scudo
- **MVSW_BOOTS** — Stivali
- **MVSW_GIRDLE** — Cintura

Il sistema consente di indossare solo un tipo di armatura per volta. *MFS_Grammar* non definisce nulla di speciale per quanto riguarda gli oggetti *ClothingArmor*: le routine standard e le azioni Wear/Disrobe della libreria Inform sono sufficienti.

La classe *PC_Warrior* fornisce un supporto speciale per il personaggio giocante visto come guerriero. Se il giocatore deve poter combattere secondo le caratteristiche MFS, deve essere un oggetto derivato da *PC_Warrior*. In particolare, definisce la routine *level_gained(attack defense, base_damage)* che notifica il giocatore delle sue nuove potenzialità e ridefinisce la *dead_proc* in modo che imposti il *deadflag* coerentemente.

La classe *NPC_Warrior* è più complessa; si tratta di una classe dotata di un minimo di intelligenza nello scegliere come combattere:

- **wait** — turni di "studio" prima di iniziare il combattimento;
- **aggr_rate** — numero di turni che passano prima di decidere se attaccare o no;
- **aggressivity** — % di decisione di attacco (0 non attacca mai, 100 attacca sempre);
- **wait_message** — routine chiamata durante l'attesa preliminare prima dell'attacco.

L'*NPC_Warrior* di base semplicemente sceglie un bersaglio a caso fra quelli possibili e combatte fino alla morte. In questa sede non possiamo spiegare il meccanismo di dichiarazione del bersaglio desiderato, ma dovrebbe essere abbastanza semplice scrivere una sottoclasse di *NPC_Warrior* che scelga i propri bersagli in modo più intelligente e fugga quando

capisce che la partita è persa.

Tuttavia, gli *NPC_Warrior* sono sensibili agli ordini del giocatore; quando il giocatore combatte, gli NPC_Warrior di default che abbiano la stessa alleanza attaccheranno lo stesso bersaglio scelto dal giocatore; inoltre, obbediranno a comandi nella forma `<amico>, attacca <nemico>` (a meno che il bersaglio designato non sia un Warrior della loro stessa alleanza).

Stessa cosa avviene per qualsiasi oggetto che fornisce *damage()*: è il trucco usato per far sì che il golem abbatta il muro al posto del giocatore in WarMage. Il muro fornisce *damage*, e se il giocatore lo attacca e il golem è nei pressi, o se il giocatore ordina al golem di attaccarlo, la *damage()* del muro si becca una tranvata da diverse tonnellate; controlla la sorgente, vede che è un danno di tipo da impatto (che il giocatore non può generare direttamente in Warmage), e abbatte il muro.

Il Sistema di combattimento definisce anche due metaverbi che consentono al giocatore di sapere quale sia la situazione sua, dei suoi alleati e dei suoi nemici.

Il verbo meta *diagnosi* genera l'azione *Diagnose*, la quale, se disponibile, chiama la routine *diagnose()* dell'oggetto indicato. Gli oggetti Warrior hanno di default una routine *diagnose()* che dà qualche indicazione sul loro stato di salute con un'approssimazione del 10% (questo anche per il giocatore stesso).

Il verbo meta *stats* o *statistiche* chiama l'azione *PlaySheet*, la quale, se disponibile, chiama la routine *play_sheet()* dell'oggetto indicato. Di default, la routine *play_sheet* dei Warrior mostra le loro caratteristiche di gioco, inclusi tutti i bonus che hanno a disposizione; tuttavia, se la proprietà verbosity è impostata a *MVSW_VERB_LO* i dati saranno "oscurati" o non disponibili. In genere, si vorrà impostare la verbosità dei PNG avversari a *MVSW_VERB_LO*, mentre si vorrà mantenere alta per il giocatore e per gli alleati. In WarMage ho deciso di lasciare la verbosità alta su tutti i PNG per agevolare il giocatore in un gioco dove questa informazione è cruciale.

Magia

Hehe… so che molti di voi aspettavano questo momento. E non solo: qualcuno di voi (mi auguro molti) avrà giocato a WarMage, e potrebbe quindi essersi fatto un'idea particolare del sistema di magia che le MFS mettono a disposizione.

Beh… il sistema di magia di WarMage sfrutta circa il 20% delle potenzialità del sistema MFS; nelle MFS trovano spazio praticamente tutti i sistemi di magia adottati nell'Interactive Fiction dalle sue origini fino ai giorni nostri. A partire da semplici "oggetti magici singoli" alla *WishBringer*, fino a replicare il sistema "impara-lancia" della saga di *Enchanter*, passando per i punti magia alla *Pietra della Luna* per finire con il sistema originale generatore di fatica/temporizzato di *WarMage*, in qualunque modo voi abbiate mai immaginato la magia, e con qualsiasi stile vogliate implementarla nella vostra avventura, le MFS ce l'hanno.

Le entità principali del sistema di magia delle MFS sono gli incantesimi (*Spell*) e le fonti (*Source*).

Gli incantesimi.
Questa è la definizione tipo di un incantesimo:

```
Spell uno_spell
with name "parola_magica",
    purpose "riassunto",
    casting_time 1,
    cost 1,
    power 2,

    description "Descrizione completa",

    fire[] ...
    fire_to_all[] ...
    ;
```

Gli incantesimi hanno tre statistiche rilevanti: il potere, il costo e il tempo di lancio. Il *potere* è il livello di potere che il mago deve aver raggiunto prima di lanciare l'incantesimo. Il *costo* è un numero che indica la fatica o l'energia necessaria a lanciare un certo incantesimo. Il *tempo* di lancio è il numero di turni necessario a lanciare un incantesimo; il giocatore deve attendere un numero di turni pari a quelli richiesti senza essere interrotto per riuscire nel lancio. A seconda delle impostazioni del sistema è possibile ignorare completamente una o più di una di queste statistiche.

Il nome dell'incantesimo è usato per individuarlo e per permettere all'utente di lanciarlo. La proprietà *purpose* è una brevissima descrizione dell'incantesimo, mentre description indica tutti i dettagli del funzionamento.

La classe *Spell* deriva da *Knowledge*; quindi l'attributo *known_about* permette di "conoscere" un incantesimo, che diventa quindi disponibile per il verbo "spiega". Con `spiega <parola_magica>` si ottiene la descrizione dell'incantesimo a diversi gradi di verbosità (a seconda delle impostazioni di sistema).

Gli incantesimi si dividono in due classi; quelli standard e quelli di attacco (classe *AttackSpell*). Gli AttackSpell interagiscono con il sistema *MFS_War* andando a colpire oggetti che forniscono la proprietà *damage()* (occupandosi di gestire gli effetti anche sugli oggetti che non la forniscono o che dichiarano di essere immuni tornando 0 da damage); gli incantesimi standard possono causare un effetto nel loro bersaglio direttamente, oppure, se disponibile chiamando la sua routine *affect*, o ancora attraverso la "collaborazione" dell'oggetto bersaglio. In un'IF ben costruita, gli oggetti dovrebbero riconoscere gli incantesimi più comuni implementando una proprietà *affect* che riceverà, come parametro, una delle seguenti costanti:

- ***SPELL_RECHARGE*** — l'incantesimo vuole ricaricare l'oggetto;
- ***SPELL_UNCURSE*** — l'incantesimo tenta di rimuovere eventuali maledizioni;
- ***SPELL_REVEAL*** — l'incantesimo esamina le proprietà magiche di un oggetto;
- ***SPELL_DISPELL*** — l'incantesimo tenta di distruggere effetti di magie precedenti;
- ***SPELL_INVISIBLE*** — l'incantesimo vuole rendere l'oggetto invisibile;
- ***SPELL_PROTECT*** — l'incantesimo vuole proteggere l'oggetto;
- ***SPELL_DESTROY*** — l'incantesimo vuole distruggere l'oggetto;
- ***SPELL_REPAIR*** — l'incantesimo vuole riparare l'oggetto;
- ***SPELL_LOCK*** — l'incantesimo vuole chiudere l'oggetto;
- ***SPELL_OPEN*** — l'incantesimo vuole aprire l'oggetto;
- ***SPELL_USER*** — disponibile per l'utente.

Aggiungete altri effetti a vostra fantasia; si tratta di numeri interi, quelli al di sopra del 20 sono disponibili.

Un oggetto che riceve *affect* può tornare true se l'incantesimo funziona, e false se l'oggetto non accetta l'effetto. È l'incantesimo che descriverà cosa succede; l'oggetto non è tenuto a spiegare alcunché.

Gli incantesimi devono fornire almeno una delle proprietà chiamate *fire(target)* e *fire_to_all()*. La routine *fire(target)* ha il compito di applicare gli effetti dell'incantesimo su un bersaglio preciso, mentre *fire_to_all* viene chiamato se l'incantesimo può essere lanciato genericamente, senza un bersaglio specifico.

Il modulo *MFS_Grammar* definisce due FakeAction che possono essere intercettate dagli oggetti sia come *before* che come *react_before* come ultima risorsa per l'implementazione degli incantesimi. Si tratta di un metodo inelegante, ma alle volte efficace.

Le FakeAction sono *CastAt* e *FireAt*. L'azione *Cast* viene generata nei confronti dell'incantesimo, e ha come "second" il suo bersaglio; *CastAt* invece ha come noun il bersaglio, e come second l'incantesimo originale. Queste due azioni vengono generate quando il giocatore decide che intende lanciare un incantesimo; intercettarle significa impedire al giocatore di portare a termine il proprio incantesimo. Per fare un paragone, intercettare una Cast o CastAt è come tappare la bocca al mago prima che finisca il suo lavoro.

Fire viene generata con noun che punta allo Spell lanciato e second al suo bersaglio, e *FireAt* viene inviata al bersaglio con lo spell come second, quando l'incantesimo ha effetto. In alcuni contesti (ad esempio, in WarMage), le azioni Cast e Fire sono temporalmente distanti; inoltre, un incantesimo potrebbe tentare di esercitare il proprio effetto (*Fire*) anche senza che un mago tenti di lanciarlo (*Cast*): è possibile ad esempio compiere azioni che causano come effetto collaterale l'avvio di un incantesimo. Il giocatore, ad esempio, potrebbe spezzare una bacchetta magica, causando l'effetto degli incantesimi (*Fire*) senza lanciarli esplicitamente (*Cast*). Se le azioni *Fire* e *FireAt* vengono lasciate filtrare, allora si passa all'atto finale dell'incantesimo, che consiste nella chiamata della sua routine *fire(target)* o *fire_to_all()*.

Alcuni incantesimi potranno permettersi di verificare il tipo di bersaglio in *fire* (o tutti i bersagli o il loro ambiente in caso di *fire_to_all*) ed agire direttamente su di essi. Questo è il caso del "Litdem" di WarMage, che dà semplicemente l'attributo *light* all'oggetto bersaglio in *fire()* o al luogo in cui si trova il 'caster' in *fire_to_all()* (dopo aver verificato alcune semplici regole). Altri si potranno appoggiare sull'eventuale presenza di *affect* nel loro bersaglio, e dichiarare il fallimento in caso contrario. Altri ancora potrebbero richiedere necessariamente di essere intercettati, quindi al momento in cui *fire/fire_to_all* è chiamato dovrebbero dichiarare semplicemente il loro fallimento. Per quanto riguarda la classe *AttackSpell*, la *fire/fire_to_all* è già impostata correttamente ad una routine che verifica la presenza di *damage()* nel bersaglio e si comporta coerentemente rispetto ad essa.

In particolare, è possibile chiamare *damage(o, tipo_di_fonte)* prima di infliggere il danno vero e proprio; la routine tornerà vero se il danno sarà accettato e falso se il bersaglio è immune; quindi l'incantesimo può dichiarare di aver bruciato il bersaglio prima che il bersaglio stampi l'informazione di essere stato ridotto in briciole.

Le "Fonti"

Le altre entità facenti parte del sistema di magia sono le fonti. Queste non sono dichiarate attraverso una classe specifica e si dividono in due categorie: le fonti di lancio (*casting sources*) e le fonti di apprendimento (learning sources).

Un oggetto che fornisca le routine *can_learn(spell)* e *learn(spell)* è una "learning source", e permette ai maghi di imparare gli incantesimi. Allo stesso modo, un oggetto che fornisca *can_cast(spell) e "cast(spell)" è una "casting source", ed è effettivamente l'oggetto che verrà incaricato dal sistema di chiamare in sequenza l'azione *CastAt* e *FireAt*.

La grammatica degli incantesimi è un dettaglio affascinante. *MFS_Grammar* definisce i verbi `lancia <incantesimo>`, `lancia <incantesimo> a <bersaglio>` e `memorizza/impara incantesimo`. È possibile lanciare direttamente un incantesimo anche semplicemente con `<incantesimo>` e `<incantesimo> <bersaglio>`, in stile Infocom. Le modalità tramite le quali queste azioni saranno svolte dipende dalla presenza di learning e casting source pronte ad essere usate; questo ci porta molto lontano, ma procediamo per ordine, e vediamo come si configurano i sistemi pre-definiti di magia. Sarà poi più facile capire come estenderli.

Selezionare i sistemi di magia

Una delle cose più intriganti dell'MFS è che non forza a scegliere un sistema di magia specifico. In realtà, si possono miscelare sistemi di magia diversi nello stesso gioco; infatti, tutto il controllo del sistema è gestito dalla classe *Memory*, o meglio, dalle sue sottoclassi.

Ho predisposto tre sistemi di magia di base, che sono quelli più diffusi e comprensibili, a mio modo di vedere; ognuno poggia su una classe di memoria diversa. Prima di procedere, è importante ricordare che la Memoria è un oggetto dal quale gli incantesimi possono essere lanciati, e infatti implementa l'interfaccia della *Casting Source* (can_cast/cast).

Vediamo quindi come funziona la memoria e successivamente alcuni modelli di memoria (e quindi di sistema) predefiniti nell'MFS:

```
Class BaseMemory with
     owner                      ! il personaggio che possiede questa memoria.
     room                       ! Numero di incantesimi disponibili
     cost_bonus                 ! Bonus se il sistema prevede un costo di lancio
     time_bonus                 ! Bonus se il sistema prevede un tempo di lancio
     timeLeft                   ! Se ci vuole tempo a lanciare, tempo rimasto
prima di ...
     spellToFire                ! ... lanciare questo incantesimo ...
     target                     ! ... a questo bersaglio ...
     mage_power,                ! Livello del mago
     timed_spells               ! MFS_INSTANT o MFS_TIMED
```

Il sistema dei bonus funziona in modo simile a quello presente nel sistema di combattimento: *cost_bonus* e *time_bonus* sono delle proprietà che possono essere fornite da tutti gli oggetti all'interno dell'owner; coloro che hanno un *giving_bonus()* che torna true, o non hanno la proprietà giving_bonus, aggiungeranno il valore alla statistica finale.

Gli incantesimi conosciuti dal giocatore sono all'interno della memoria *(parent(spell)==memory)*; questo perché la relazione di parentela è l'unica dotata della flessibilità adeguata in Inform. Per i PNG, può bastare un vettore che indica quali incantesimi conosce, ma il giocatore necessita di una relazione più dinamica, e Inform fornisce solo la relazione di proprietà a questo scopo.

In particolare, la memoria del personaggio giocante (inclusi i valori di default) è:

```
Class PC_Memory class BaseMemory
with
     known_spells,          ! Numero di incantesimi conosciuti
     memorized_spells,      ! Numero di incantesimi memorizzati
     cost_name "costo",     ! Il nome della "forza magica" nel tuo sistema, es.
"mana"
     multiple_spells MFS_NO_MULTIPLE,   ! previene gli incantesimi multipli
     over_max MFS_FORBID,   ! Impedisce di memorizzare più incantesimi del mas-
simo
     mem_max 0,             ! massimo numero di incantesimi (0 = infinito)

     ! seguono alcuni messaggi "customizzabili" di interesse generale

     !... quando un incantesimo non ha bisogno di essere imparato...
     free_learn "Conosci gi@`a quest'incantesimo perfettamente.",

     !... quando la memoria è vuota ...
     no_spell_string    "Non conosci nessun incantesimo!",

     !... header della lista degli incantesimi ...
     spell_string "Conosci i seguenti incantesimi:^",   ! additional crlf

     ! Quando il numero di incantesimi è pari al massimo e si tenta di
     ! imparare un nuovo incantesimo.
     low_mem_msg "La tua mente non @`e allenata per sopportare lo sforzo di
            apprendere cos@`i tanti incantesimi!",
```

Un'altra routine molto importante è la *inflict_cost*. Viene chiamata quando il costo dell'incantesimo è imputato al giocatore; non si può avere niente per niente...

Il contenuto di una memoria può essere visualizzato dal giocatore attraverso il metaverbo *memoria* o *mem*.

Tramite la semplice configurazione di questi parametri e poco altro, è possibile costruire una gamma molto ampia e varia di sistemi di magia; vediamo quelli predefiniti:

Il sistema "cast & forget"

Il sistema più semplice e quello tradizionalmente conosciuto come "infocom" ed è il sistema "lancia-e-dimentica". Il mago impara una serie di incantesimi da fonti (*learning sources*) diverse: libri o pergamene, oppure gli incantesimi vengono "aggiunti" alla sua memoria in modo diverso (direttamente dal gioco). Il mago ha un numero limitato di "slot" o di incantesimi che può memorizzare. Ogni volta che usa un incantesimo, lo dimentica ed è costretto a reimpararlo nuovamente. È anche possibile imparare gli incantesimi più di una volta, e in questo caso saranno disponibili lo stesso numero di volte.

Il sistema C&F è generalmente adeguato a piccoli giochi, o giochi in cui pochi incantesimi potenti usati correttamente permettono di progredire o risolvere il gioco. Questa è la definizione della memoria C&F:

```
class CFMemory class PC_Memory
with
     cost_name nothing,
     mem_max 4                ! maximum number of spells that can be held
     multiple_spells MFS_ALLOW_MULTIPLE,
     over_max MFS_FORGET,
     ! ... altre ...
```

over_max può essere configurato per impedire al giocatore di apprendere troppi incantesimi (*MFS_FORBID*), o come nel modello infocom, per dimenticare un incantesimo a caso se si tenta di superare le proprie forze (*MFS_FORGET*). Con *multiple_spells* impostato a *MFS_ALLOW_MULTIPLE*, si istruisce la memoria di permettere "l'apprendimento multiplo". *mem_max* può essere configurato per permettere l'apprendimento di più o meno incantesimi; 4 era il numero per *Enchanter*...

Interessante è l'implementazione del costo dell'incantesimo:

```
    inflict_cost [sp;
            self.forget_spell(sp);
     ];
```

Un'applicazione potrebbe voler anche modificare *no_spell_string* e *spell_string*, dato che non è esatto dire in questo contesto che il giocatore "non conosce" gli incantesimi, bensì bisogna parlare di "incantesimi attualmente memorizzati".

Il sistema C&F funziona meglio se non è temporizzato: è comunque possibile temporizzare gli incantesimi in modo che vengano lanciati dopo un certo tempo impostando *timed_spells* a *MFS_TIMED*.

Inoltre, la bassa complessità del sistema fa spesso sì che non ci siano limitazioni di livello sugli incantesimi appresi; in questo caso basta impostare *mage_power* a *MFS_WARLOCK*.

Infine, tutte le statistiche degli incantesimi sono insensate in questo contesto; quindi, è opportuno impostare

```
SpellVerbosity = MFS_SPELLVMED;  ! si tratta di una variabile globale.
```

Questo visualizzerà solo il *purpose* e la descrizione completa dell'incantesimo, senza le statistiche numeriche. In alcuni giochi, parte del "gusto" consiste nello scoprire come usare incantesimi sconosciuti; con

```
SpellVerbosity = MFS_SPELLVLO;  ! si tratta di una variabile globale.
```

si limita la visualizzazione degli incantesimi al solo purpose.

Il sistema "Magic Points"

Questo sistema prevede un pool di "punti magia" che può venire reintegrato durante l'avventura (oppure no), e dal quale il mago può attingere indifferentemente per lanciare i propri incantesimi.

Quindi, gli incantesimi memorizzati non si scordano mai, e non ha quindi senso memorizzare più volte lo stesso incantesimo. Tendenzialmente, il mago è ancora concepito come un "fucile" spara incantesimi, quindi si tende ad evitare di temporizzare questo sistema (richiedere attesa prima che gli incantesimi abbiano effetto).

Questa è la definizione della memoria MP:

```
class MPMemory class PC_Memory with
     mpoints 10,                ! a standard dotation of magic points
     max_mpoints 10,
     cost_name "Magic Points",  ! The name of "mana" in your system
```

La classe verifica anche che, prima di lanciare gli incantesimi, considerando anche i bonus forniti da altre fonti (in termini di *cost_bonus/giving_bonus*), siano disponibili abbastanza punti magia.

Ad esempio, ecco la routine che infligge il costo:

```
     inflict_cost [sp rc;
            rc = sp.cost - self.cost_bonus -
                 GlobalValue(self.owner,cost_bonus,giving_bonus);
            if (rc < 0) rc = 0;
            self.mpoints = self.mpoints - rc;
     ];
```

Come si vede, il costo è pari al costo dell'incantesimo epurato dei *cost_bonus* provenienti dal mago che possiede la memoria e dai suoi oggetti.

Le statistiche degli incantesimi sono generalmente rilevanti, quindi è opportuno impostare

```
SpellVerbosity = MFS_SPELLVHI;   ! valore di default.
```

La routine *explain* della classe *Spell* evita di visualizzare le statistiche non necessarie: se la memoria del giocatore non è *MFS_TIMED*, non visualizzerà il tempo di casting.

Il sistema "Fatigue Generating"

Il Sistema FG vede la magia come "forza vitale" del mago, che esso accumula (in un certo tempo) e poi rilascia; essendo forza vitale, la magia che abbandona il corpo causa un danno al mago.

In termini MFS, manda un messaggio alla sua *damage()* con causa del danno *MFS_DMG_FATIGUE* (che, come per tutto, può anche rifiutare il danno subito, rendendo il mago onnipotente).

Ecco la sua definizione:

```
class FGMemory class PC_Memory
     with
     cost_name "Fatica",

     ! ...

     inflict_cost [sp rc;
            if (player provides damage) {
                 rc = sp.cost - self.cost_bonus -
                      GlobalValue(self.owner,cost_bonus,giving_bonus);
                 if (rc < 0) rc = 0;
                 player.damage(rc, DMG_FATIGUE, self);
            }
     ];
```

Quindi, se il mago esagera... muore. Questo è il sistema adottato in WarMage (con l'aggiunta di `timed_spells = MFS_TIMED`); Un sistema timed si sposa molto bene con il sistema di combattimento, che consente agli avversari di reagire prima di essere inceneriti (forse un magic points sarebbe meglio di un FG in questo contesto).

Come per la memoria MP, la FG non necessita di apprendimento multiplo degli incantesimi, dal momento che questi non vengono mai dimenticati.

Imparare gli incantesimi — libri e pergamene
La routine *Memory.know_spell(spell)* consente di memorizzare un incantesimo nella memoria del giocatore. Questo dà all'incantesimo l'attributo *known_about* e lo muove all'interno della memoria, o ne aumenta eventualmente il conteggio di memorizzazione se questo è consentito.

Il sistema di gioco prevede il concetto di "learning source" per permettere al giocatore di apprendere gli incantesimi da oggetti come pergamene e libri.

Il verbo "studia/impara/memorizza " consente di memorizzare incantesimi scritti su pergamene e libri. Ogni "learning source" può presentare un costo di apprendimento (proprità routine *inflict_cost*), così come le casting source (la memoria, fino ad ora) presentano un costo di casting. Nel caso dei libri magici, questo costo è inesistente; nel caso delle pergamene, il costo consiste nella cancellazione del loro contenuto; quando la pergamena è vuota, viene distrutta.

Le pergamene sono preziose anche perché possono essere utilizzate per lanciare gli incantesimi immediatamente (anche in un sistema timed) e senza l'applicazione di alcun costo sul mago. Le pergamene, infatti, sono casting source, come le memorie; quindi possono essere usate sia per apprendere che per lanciare incantesimi.

Inoltre, le pergamene possono essere srotolate, scaricando tutta la loro magia in un colpo solo (ossia, chiamando la *fire_to_all* di tutti gli incantesimi presenti su di esse).

Le pergamene e i libri magici sono definiti nel file *MFS_SpellWritable.h*.

La classe base *SpellWriteable* definisce i comportamenti minimi per gli oggetti magici che possono essere "scritti" con incantesimi, o che possono essere fonti di apprendimento per gli stessi.

La proprietà *content* è un vettore di incantesimi (possono essere anche ripetuti più volte, dando origine a diverse scritture dello stesso incantesimo). La *MFS_Grammar* ridefinisce il verbo "leggi" e gli assegna l'azione Read; questa viene intercettata e gestita listando il contenuto degli *SpellWriteable*.

Quando il nome di un incantesimo appare per un qualsiasi motivo (ad esempio in seguito ad un `leggi <SpellWriteable>` che elenca tutti gli incantesimi contenuti in esso), riceve l'attributo *known_about*, e può quindi essere oggetto del metacomando "spiega"; ricordiamo che gli *Spell* sono discendenti di *Knowledge*, e funzionano in maniera similare alle leggende (*Tale*).

L'azione "Examine" esercitata sugli *SpellWriteable* non viene intercettata, quindi visualizza semplicemente la descrizione (proprietà *description*) dell'oggetto.

Come dice il nome, gli *SpellWritable* possono essere "scritti" fino alla loro massima capacità; Normalmente, il verbo `scrivi <incantesimo> su <supporto>` (se l'incantesimo è in memoria) è sufficiente, ma può essere richiesto che il giocatore abbia con se un oggetto come ad esempio una penna magica affinché l'azione abbia successo. Questo si può ottenere cambiando la routine *ScribeSub* in *MFS_Magic.h*.

Alternativamente, la Infocom ha usato un incantesimo con lo scopo di … scrivere incantesimi: il famoso *Gnusto*. C'è una versione MFS del Gnusto che può valere la pena studiare in *MFS_StandardSpells.h*.

Oltre ad essere scritte, le pergamene possono essere preparate come fonte di lancio: con il verbo `prepara <pergamena>` si fa più o meno quello che si fa con `sguaina <arma>`. La pergamena viene messa a disposizione del mago; ora, quando questo userà l'incantesimo con

`lancia <incantesimo>`, fra le sue fonti di lancio sarà disponibile la pergamena, che fornirà l'incantesimo al posto della memoria. Si può cambiare idea con il verbo `riponi <pergamena>`.

Magia libera: for_free

La storia del Gnusto mi fa venire in mente l'attributo *for_free*. Si può assegnare questo attributo ad un incantesimo qualsiasi (anche dopo l'inizio del gioco). Gli incantesimi che hanno *for_free* non infliggono nessun costo di lancio, non possono essere memorizzati più di una volta (e non sono mai dimenticati) e vengono lanciati immediatamente anche in una memoria timed. Per questo motivo, vengono listati separatamente rispetto agli altri, quando il giocatore usa il verbo "memoria", con una dicitura del tipo "L'incantesimo XXX è tuo per sempre..; inoltre hai..."

Incantesimi da oggetti

Le pergamene non sono gli unici casting source oltre la memoria. Gli implementatori ne possono creare altri, ma la MFS fornisce già alcuni tipi di oggetti magici predefiniti. La classe *MagicDevice* è la base di una serie di oggetti che possono contenere incantesimi e essere usati dai maghi per lanciali usandoli come fonte di magia al posto della loro memoria.

```
class MagicDevice
with
     charges -1,                    ! infinite charges
     verbosity MFS_SPELLVMED,       ! Medium verbosity in description
     description [;
            ! ...
     ],

     affect [ spell;
            if (spell ~= SPELL_RECHARGE) rfalse;
            ! ...
     ],

     exausting[;
            "^L'energia emanata ", (artda) self, " @`e molto debole.
            Comprendi che non potrai usar", (itorthem) self," pi@`u.";
     ];
```

Come si vede, la classe implementa un la *description* (che riporta lo stato del device, con modalità diverse in base alla verbosità), *affect* (che ricarica l'oggetto), e *exhausting* che è un messaggio generico che segnala l'esaurimento del potere.

Normalmente, i magic device hanno **UN SOLO** incantesimo, ed un numero di cariche ossia, di volte che quell'incantesimo può essere usato dal device. Sempre normalmente, i device devono essere preparati come le pergamene; ecco un derivato utile della classe *MagicDevice*:

```
Class MagicBattery class MagicDevice
with
     ! Le sottoclassi devono definire "content <spellname>",
     charges 3,
     max_charges 3,
     prepared 0,        ! preparata?
! ...
```

Se il sistema di magia prevedesse solo l'uso di oggetti magici per lanciare gli incantesimi, senza prevedere la memoria (usando bacchette, verghe, anelli, feticci etc), sarebbe possibile impostare prepared sempre a 1 e disabilitare i verbi "prepara" e "riponi": non avrebbe infatti senso selezionare gli oggetti magici in un simile contesto... dato che andrebbero comunque usati solo quelli. Con questo semplice accorgimento si può costruire un gioco con magia legata unicamente agli oggetti...

E l'oggetto che segue è ancora più potente.

```
class MagicGenerator class MagicDevice
    with
    content object_spell,           ! l'incantesimo usato da questo generatore
    activation 'shazam',
    charges -1,                     ! cariche infinite
    verbosity MFS_SPELLVMED,        ! Verbosità media nella descrizione.

    describe_contents[i words;
        print "Puoi usare ", (the) self, " pronunciando ";
        ! ...
    ],


    affect[flag;
        if (flag ~= SPELL_REVEAL) rfalse;
        self.describe_contents();
        give self known_about;
        rtrue;
    ],

     !...
```

Et voilà. Una lampada magica che lancia un incantesimo (con *fire_to_all*) al semplice pronunciare una parola, o compiere un'azione...

In particolare, questo oggetto chiama *object_spell.fire_to_all()* quando il giocatore scrive *shazam* (dopo che *affect(SPELL_REVEAL)* ne ha scoperto i poteri).

Altri oggetti magici.
In WarMage ho usato due incantesimi di effetto notevole: il dordum (portale) e l'enjur (golem). Entrambi funzionano muovendo nel gioco due oggetti molto articolati, il portale e il golem, appunto; nel primo caso viene anche compiuta una sorta di "riconfigurazione" (il portale punta ad un'altra area del gioco).

Vale la pena studiare questi due oggetti, insieme ad alcuni "skeleton" per oggetti a vario titolo "magici" in *MFS_MagicObject.h*

Conclusioni

Questo breve (!) articolo non può e non vuole essere una guida di riferimento completa alle MFS. Lo scopo è quello di incuriosire il lettore, e fornirgli le basi "cognitive" necessarie a comprendere il funzionamento in generale della libreria e dei suoi moduli. Per vedere nel dettaglio il funzionamento delle librerie, si rimanda ai commenti inclusi nei file sorgente delle MFS ed all'utilizzo che ne è stato fatto in WarMage.