Numero 10 – Ottobre 2006

Indice

Editoriale

di Roberto Grassi

Benvenuti al numero 10 di Terra d'IF.
E' un numero speciale, per noi, perchè rappresenta il giro di boa del terzo anno. Un risultato davvero sorprendente, per un'avventura nata sulle ali dell'entusiasmo. Speriamo di essere stati utili, in questi tre anni, a dare informazioni e supporto alle persone che si avvicinano al mondo della narrativa interattiva e speriamo, d'altra parte, di continuare a farlo per molti anni a seguire.

Veniamo ai contenuti. Nella sezione articoli compare un bel contributo di Paolo Vece sulla storia e i giochi della Infocom. Segue un articolo di Giovanni Riccardi (perennemente in ritardo :)) sulle prime valutazioni di Inform 7. "1-2-3″ e "All Alone" sono i giochi recensiti in questo numero. Per quello che riguarda l'intervista, Riktik e' riuscito a scovare Marco Zanchi, titolare della CIPA Informatica, che ha creato molte avventure testuali (tutt'ora scaricabili dal sito di IFItalia). La rubrica "Design Review" si sofferma sulla gestione delle 'locazioni irraggiungibili', un aspetto solitamente trascurato nello sviluppo di una IF. Infine, nella rubrica dedicata alla programmazione Paolo Lucchesi contribuisce con un bell'articolo su "Inseguiti e Inseguitori", utilizzando la classe Follower. Alla prossima...

Infocom - Dalla Z di Zork alla A di Arthur

Un rapido viaggio sui successi (e gli insuccessi) di una compagnia unica.

di Paolo Vece

Il 1979 è un anno importante per gli appassionati d'Interactive Fiction perché segna la nascita della più famosa, leggendaria ed affascinante compagnia di software del mondo: la Infocom. Responsabile di alcuni dei più bei giochi per computer mai prodotti, partendo dal fantasy di Zork, proseguendo con il mistery di Deadline, la fantascienza di Starcross e Planetfall, attraverso la saga di Enchanter ed il successo planetario di The Hitchhicker's Guide to the Galaxy, fino ad arrivare alla mitologia di Arthur. Una compagnia che ha certamente segnato il decennio 1979-1989 e di cui ancora oggi si apprezza la freschezza e l'attualità dei suoi prodotti.

Il tutto, però, era cominciato qualche tempo prima nel 1979, nel laboratorio d'informatica del MIT, con un progetto che aveva il curioso nome di Zork…

Il MIT

Durante la metà degli anni '70 i computer erano grandi e costosi, e quindi era normale - e conveniente - che più utenti fossero connessi allo stesso computer principale tramite dei terminali per sfruttarne al massimo le capacità. In quel periodo era anche attiva ARPAnet, la rete che poi si sarebbe evoluta nell'Internet che oggi conosciamo, e chiunque avesse avuto un account sulle macchine presenti in ARPAnet poteva collegarvisi.

All'inizio del 1977 Adventure (conosciuto anche come Colossal Cave), il gioco che viene accreditato essere la prima avventura testuale, era in giro per la rete, e naturalmente approdò anche al laboratorio d'informatica del MIT. Nel succitato laboratorio si creavano progetti interessanti, tra i quali un gioco multiutente in cui i partecipanti vagavano per un labirinto sparandosi contro, oppure il gioco Trivia, in cui ci si sfidava a colpi di domande e risposte. In un ambiente stimolante come quello, ovviamente, Adventure ebbe subito successo: orde di utenti che si collegavano solo per portare a termine l'avventura con il massimo dei punti previsti.

Poiché stiamo parlando di un laboratorio d'informatica, accadde la cosa più logica che sarebbe potuta accadere: pensarono di scrivere il loro Adventure. Un Adventure migliore dal punto di vista della trama, degli enigmi e dell'interazione con l'utente.

Dave Lebling scrisse un prototipo di parser, e partendo da questa base Marc Blank, Tim Anderson e Bruce Daniels scrissero un gioco di prova. Era un gioco semplice, fatto di poche locazioni, ma la cosa importante era il come era stato programmato, avendo creato un modo per cambiare in maniera semplice i vari aspetti di cui era composto.

Per ‘parser’, nel mondo delle avventure testuali, si intende quella parte di programma che si occupa di capire quello che ha scritto il giocatore ed agire di conseguenza. Ci sono parser più o meno sofisticati, in grado di capire molte parole o frasi complesse ed un numero variabile di vocaboli.

Ed è proprio partendo da quel prototipo che in sei settimane venne fuori la prima versione di Zork...

Zork

Zork era, nel gergo usato al MIT, una parola senza senso, usata per creare delle frasi senza un particolare significato, ed il team che stava scrivendo il gioco la usava per dare un nome ai progetti che ancora non avevano un nome definitivo.

Ben presto, per la rete, si sparse la voce che esisteva un altro gioco tipo Adventure e gli utenti aumentarono. Con l’aumento degli utenti la base su cui provare le nuove invenzioni si allargava. Ad un certo punto, visto il successo, qualcuno del gruppo di sviluppo pensò bene di dare un nome definitivo alla creatura, ed il nome scelto era Dungeon. Fortunatamente per tutti, il nome era parte di un marchio registrato di un altro famoso gioco con un nome molto simile, e si tornò a chiamarla Zork.

Il lavoro su Zork andò avanti per quasi due anni, tra aggiunte di elementi, creazione di nuovi enigmi e miglioramenti del parser. Praticamente, le diverse sezioni in cui si espande il mondo di Zork vengono create in questo periodo, e quello che ancora oggi giochiamo come Zork è il lavoro di un gruppo di studenti che si appassionarono ad un determinato prodotto e lo portarono avanti migliorandolo notevolmente versione dopo versione.

Soprattutto, quello che abbiamo è la creazione della base sulla quale si baseranno i successivi prodotti della Infocom. Sembra che, durante lo sviluppo, Marc Blank fosse diventato ossessionato dai parser e ne scrisse una quantità industriale. Il parser di Zork era veramente allo stato dell’arte: rispetto al più semplice parser di Adventure, che accettava comandi di due parole nella forma verbo+oggetto (ad es. PRENDI LAMPADA), quello sviluppato al MIT era in grado di capire frasi complesse, di discernere tra più oggetti simili e capire da solo quale fosse l’oggetto più appropriato da usare in un determinato momento (in Zork erano possibili frasi come PRENDI TUTTO DAL TAVOLO TRANNE LA LAMPADA). Avere un parser sofisticato vuol dire migliorare l’interazione degli utenti con il gioco e la possibilità di sviluppare degli enigmi più complessi ed elaborati.

Oltre al parser, quello che si è andando sviluppando in questo periodo è il cosiddetto world-model (modello del mondo). Con world-model si intende quella struttura sulla quale si poggiano le interazioni tra i vari oggetti che compongono il mondo del gioco: il poter mettere un oggetto sopra un tavolo, o sotto allo stesso tavolo. Avere con sé una borsa in cui infilare della roba. Salire su una barca per attraversare un fiume. Interagire con altri personaggi presenti nel gioco. Sono tutte cose che fanno parte del mondo che ci troviamo ad esplorare e che, in qualche modo, devono funzionare. Quello che è stato fatto con lo sviluppo di Zork è stato di definire queste regole in maniera tale da poter essere utilizzate - e modificate - facilmente.

In più, in Zork, abbiamo l’implementazione di risposte spiritose e divertenti ad azioni

inaspettate. Sono dei piccoli tocchi di classe che fanno sì che un utente si immedesimi al meglio nel gioco che sta affrontando. Il credere che tutto sia manipolabile e che ad ogni comando scritto esista una risposta adeguata fa parte della magia di Zork, e dei prodotti che l'hanno seguito.

Ed arriviamo, così, a Febbraio del 1979 che vede l'inclusione dell'ultimo enigma in Zork…

La Infocom

La fine dello sviluppo su Zork coincise, più o meno, anche con la fine del lavoro al laboratorio di ricerca di alcuni suoi membri, ed esattamente si trattava di Tim Anderson, Joel Berez, Marc Blank, Mike Broos, Scott Cutler, Stu Galley, Dave Lebling, J. C. R. Licklider, Chris Reeve ed Al Vezza. Alcuni nomi famosi per quanti conoscono ed apprezzano i giochi della Infocom, ed altri nomi meno conosciuti, ma non per questo meno importanti.

Queste persone non volevano perdere il patrimonio di conoscenze accumulato, ed erano convinti di essere in grado di poter lavorare insieme su un qualunque progetto informatico: erano convinti di essere in grado di sviluppare dei magnifici programmi. Programmi di qualunque tipo (office automation, posta elettronica) e non necessariamente giochi.

Anzi, non erano neanche convinti che il mondo dei micro-computer (gli home computer, i piccoli personal che era possibile trovare, come l'Apple II o il TRS-80) fosse appetibile.

Nel frattempo Marc Blank, che si era laureato in medicina, si era trasferito a Pittsburgh per il suo internato. A Pittsburgh si era trasferito qualche tempo prima anche Joel Berez per lavorare nell'azienda di famiglia. I due, essendo grandi amici, s'incontravano spesso per discutere dei vecchi tempi, e di quanto fosse bello lavorare al laboratorio di informatica. Di questi vecchi tempi, ovviamente, faceva parte anche Zork. Anzi, Zork era la parte migliore di quei vecchi tempi, e fantasticavano su quanto sarebbe stato bello vederlo girare su un micro-computer.

Intanto l'ex gruppo di studenti del MIT stava formando la società, scegliendo Infocom come nome. E nella lista di progetti che presero in considerazione entrò anche Zork…

La Z-Machine

Zork era un programma molto grande per l'epoca, occupava molto spazio nel computer in cui era stato sviluppato (un DEC PDP-10), e farlo funzionare su un micro-computer non era una cosa semplice. I micro-computer avevano una memoria di 16k e, solitamente, neanche il floppy disk.

Durante i loro incontri, Berez e Blank avevano cominciato a definire in che modo ridurre le dimensioni di Zork per farlo entrare in un home computer. Si profilava, inoltre, un'esplosione nella proposta di micro-computer tutti incompatibili tra di loro; Apple, Atari, Commodore, IBM, Tandy: queste ditte, erano tutte impegnate nel creare il loro micro-computer. Le priorità, quindi, erano due: far entrare Zork in un micro-computer, e

farlo funzionare su diversi modelli non compatibili fra di loro.

La prima ipotesi fu quella d'usare degli strumenti di sviluppo "portabili" (strumenti che richiedono poche modifiche per essere usati su computer diversi), come l'UCSD Pascal (un linguaggio abbastanza diffuso), ma si resero conto che in Zork c'era troppo testo e non ce l'avrebbero mai fatta.

Giunsero quindi alla conclusione che la cosa migliore da fare fosse quella di creare un sistema apposito per sviluppare e far funzionare Zork. Nei loro piani Zork sarebbe stato eseguito da una macchina virtuale, un programma, cioè, che simulava un computer immaginario da far funzionare su un computer vero. Bastava avere un emulatore di macchina virtuale da far funzionare su una macchina vera per far girare il programma scritto per la macchina virtuale. Quindi, al presentarsi di un nuovo modello di micro-computer sarebbe stato necessario scrivere solo il programma della macchina virtuale e non riscrivere anche Zork (che è, di fatto, lo stesso concetto che sta alla base del ben più famoso Java).

Questo computer immaginario avrebbe preso il nome di Z-machine, e sarebbe stato in grado di eseguire una versione di Zork appositamente scritta. Il programma che implementava la Z-machine si sarebbe chiamato ZIP (inizialmente stava per Zork Interpretive Program, poi divenne un più neutro Z-machine Interpreter Program). Insieme alla Z-machine, i due crearono anche il sistema di sviluppo. Si trattava di un linguaggio di programmazione progettato appositamente per l'implementazione di Zork, con istruzioni specifiche per la gestione del world-model. Questo linguaggio prese il nome di ZIL (Zork Implementation Language). In più, questo sistema di sviluppo sarebbe stato in grado di ridurre lo spazio occupato dal testo mediante un semplice e veloce algoritmo di compressione.

A questo punto si trattava di tradurre Zork dalla sua versione originale in una versione adatta a girare sulla Z-machine. Blank e Berez stimarono che nei limiti della Z-machine era possibile portare circa metà dello Zork originale. Nonostante tutto era ancora troppo grande per essere portato interamente in un micro-computer. Intervenne quindi Dave Lebling che, esaminando la sua mappa del mondo di Zork, tracciò un confine sulla stessa. Quello che si ottenne fu un gioco più piccolo, e tutto quello che restava fuori sarebbe stato usato per un gioco futuro. In questo processo di conversione vennero ripuliti alcuni aspetti, si introdussero dei nuovi enigmi (e soluzioni alternative ad enigmi vecchi) e la geografia del gioco venne resa più coerente.

La prima versione di Zork I disponibile fu quella per PDP-11, su cui era stato scritto uno ZIP. Tutto era pronto per la commercializzazione di Zork…

Personal Software

L'idea iniziale era quella di essere dei fornitori di software, lasciando ad altri il compito di vendere i prodotti. D'altra parte non avevano esperienza in quel campo. Vennero presi contatti con la Personal Software, una compagnia che vendeva software, sia giochi che prodotti di tipo business. L'accordo venne raggiunto anche perché il loro contatto con la Personal Software era stato un giocatore della versione originale di Zork, che sapeva quindi bene di cosa si trattasse.

Nel Novembre del 1980 venne venduta la prima copia di Zork I. E nel frattempo venne scritto il primo ZIP per micro-computer - lo creò nel Dicembre 1980 Scott Cutler sul suo TRS-80 Model II. La possibilità di veder girare Zork su un micro-computer era ormai una realtà.

In circa nove mesi la Personal Software vendette 1500 copie di Zork I (allora solo Zork) nella versione per TRS-80. Per la fine del 1980 venne creato uno ZIP anche per Apple II, e Zork per Apple riuscì a vendere 6000 copie.

Nel frattempo, Dave Lebling era impegnato nella scrittura di Zork II, e non si trattò della conversione semplice di quello che rimaneva fuori dopo aver creato Zork I, ma vennero aggiunti enigmi e non tutto venne incluso nel gioco. Anche per la vendita di Zork II si raggiunse un accordo con la Personal Software.

Però i membri della Infocom non erano molto soddisfatti del trattamento che stavano subendo i propri prodotti: loro, giustamente, erano entusiasti di quanto stavano creando, ma la Personal Software non era così incline ad investire in pubblicità, anche perché si stava spostando sempre più nella vendita di prodotti business, tra cui VisiCalc (il progenitore di tutti i fogli elettronici) e stava, appunto, per cambiare nome in VisiCorp.

A questo punto si trovarono ad un bivio: cercare un altro distributore di software o tentare di diventare degli editori loro stessi. Inutile dire che vinse la seconda opzione. Fu così che crearono la loro divisione pubblicità, si inventarono una nuova grafica per Zork (le famose lettere fatte con dei mattoni), trasferirono la società in una nuova sede più grande, ed acquistarono dalla Personal Software le copie rimanenti di Zork…

I giochi

Con i diritti di distribuzione in mano, e con una grande voglia di promuovere i propri prodotti, il futuro si prospettava roseo per la Infocom.

Ad Aprile del 1982 la Infocom pubblicò Deadline, un mistery e forse il primo esempio di Interactive Fiction che si discostava dal filone Dungeons and Dragons. Scritto da Marc Blank, ci troviamo ad impersonare un investigatore che deve indagare sulla morte di un famoso industriale. Abbiamo dodici ore di tempo per risolvere il mistero, e per farlo dobbiamo interagire con diversi personaggi. Inoltre, Deadline ha segnato l'introduzione dei famosi gadget che arricchivano i giochi Infocom. Oltre al dischetto del gioco, nella scatola si trovavano anche delle prove, una nota del coroner, una lettera dal procuratore della vittima, una foto che mostrava la scena del crimine, ed altro ancora.

Da Deadline in poi la Infocom avrebbe sempre arricchito i propri giochi con del materiale aggiuntivo. Questo materiale era importante perché aiutava a combattere la pirateria (era più bello avere tutta quella roba che il solo dischetto, ed alcuni enigmi erano risolvibili solo facendo riferimento ad alcuni degli oggetti presenti nelle confezioni) e creava inoltre la giusta atmosfera per entrare nel gioco.

La fine del 1982 vide anche l'uscita di altri due giochi: Zork III e Starcross. Il primo era il seguito di Zork II, creato includendo quanto era rimasto da parte della versione originale,

con parecchie aggiunte per arrivare a creare il gioco completo: rispetto ai primi due episodi era composto quasi interamente da materiale nuovo.

Starcross era la passione di Dave Lebling per la fantascienza trasferita in un gioco, ed è stato anche il primo gioco che ha scritto essendone unico autore. Una storia di pura fantascienza che ci vede impegnati come minatori di buchi neri per arrivare ad avere un incontro con una nave aliena. Secondo l'autore, il gioco è un omaggio ad Arthur C. Clarke e Larry Niven.

La fine dell'anno vide, inoltre, la creazione di una divisione di prodotti business. Questa divisione si sarebbe occupata dello sviluppo di un programma per la gestione di data base chiamato Cornerstone. Nelle intenzioni della Infocom avrebbe dovuto essere un programma sofisticato, ma facile da usare.

A Marzo del 1983 venne pubblicato Suspended, un altro gioco di fantascienza scritto da Mike Berlyn, uno scrittore già affermato quando si unì alla Infocom. Suspended è da molti considerato il gioco più difficile gioco Infocom. Il nostro corpo si trova in sospensione criogenica, nel cuore del sistema centrale di un pianeta semi-automatico. La nostra mente è collegata ad una rete di computer e sovraintende al mantenimento dell'equilibrio della superficie del pianeta. Ma succede qualcosa, e ci troviamo a fronteggiare un'emergenza controllando sei robot, ognuno con una determinata caratteristica, per far tornare le cose a posto. Nel frattempo la gente sul pianeta sta morendo. Dobbiamo sbrigarci.

The Witness, uscito nel Giugno del 1983, era il secondo poliziesco della Infocom ed è anche l'opera prima di Stu Galley, un autore che è entrato nel cuore di molti appassionati. Siamo nel 1938, vicino Los Angeles, e veniamo chiamati per proteggere un milionario la cui moglie è stata assassinata. Nonostante il nostro arrivo non possiamo evitare che anche lui venga ucciso e dobbiamo indagare per scoprire l'assassino. Un gioco che sprigiona atmosfera da ogni parte.

Agosto 1983, Planetfall. In una parola: Floyd! Il primo gioco di Steve Meretzky fu anche un grande successo, ed ancora oggi resta un termine di paragone. L'autore aveva cominciato come collaudatore di giochi e, collaudando collaudando, arrivò alla conclusione che avrebbe potuto scrivere un suo gioco. Fortunatamente per noi la Infocom gli diede credito e così abbiamo avuto uno dei più prolifici scrittori della compagnia. In Planetfall ci troviamo su una nave spaziale, impegnati nel compito di pulirne i ponti. Ad un certo punto succede qualcosa e siamo costretti ad abbandonare la nave a bordo di una capsula che ci fa atterrare su un pianeta apparentemente disabitato. Ma sul pianeta incontriamo anche uno dei personaggi più famosi della Infocom: il robot Floyd, che ci accompagnerà ed aiuterà a risolvere la nostra avventura.

Dave Lebling aveva sempre voluto aggiungere degli elementi di magia in Zork, ma tranne qualche accenno, non se ne era fatto nulla. Se ne uscì, allora, con un sistema di gestione degli incantesimi e, insieme a Marc Blank, scrisse Enchanter. Ambientato, in qualche modo, nello stesso universo di Zork, ci fa impersonare un giovane Enchanter che deve liberare la terra dal dominio di un oscuro signore. Il primo di una trilogia.

Il secondo lavoro di Mike Berlyn, scritto insieme a Patricia Fogleman, è Infidel. Ambientato in Egitto, dove interpretiamo la parte di un archeologo, ci svegliamo al campo base dove scopriamo che i nostri compagni di viaggio ci hanno abbandonato.

Mentre Dave Lebling scriveva Enchanter a Steve Meretzky venne in mente che avrebbe potuto scrivere un seguito per il gioco. Così, prima che Enchanter fosse finito, Steve cominciò a lavorare al suo seguito, Sorcerer che venne pubblicato nel Marzo del 1984. Dopo aver sconfitto l'oscuro signore del primo episodio veniamo accolti nella Gilda degli Enchanter. Dopo aver studiato per anni insieme al leader della Gilda, ci accorgiamo che qualcosa non va. Belboz, il nostro mentore, sembra preoccupato. Ad un certo punto Belboz sparisce e sta a noi scoprire cosa è successo e salvare Belboz da un terribile destino!

Seastalker, scritto da Stu Galley e Jim Lawrence, fu il primo di una serie di giochi denominata 'Introductory', adatta ad un pubblico più giovane, ma anche a chi non aveva mai giocato con un'avventura testuale. Jim Lawrence era uno scrittore che aveva al suo attivo una sessantina di romanzi. E Stu Galley era l'autore di The Witness, che era stato nominato Miglior Gioco di Avventura del 1984.

Le notti sull'isola di Hardscrabble sono fredde e solitarie… Così inizia Cutthroats, un'avventura scritta da Michael Berlyn e Jerry Wolper che ci vede impegnati nella ricerca di un tesoro in una nave affondata. Un nostro conoscente, poco prima di venire assassinato, ci lascia quella che sembra essere la vera mappa del tesoro sommerso. Quindi dovremo mettere insieme una squadra per cercare di recuperare il tesoro…

HHGTTG è la sigla con la quale è conosciuto uno dei più famosi giochi della Infocom: Hitchhiker's Guide to the Galaxy. Douglas Adams era un appassionato di giochi Infocom, ed anche un appassionato d'informatica, oltre ad essere il creatore di uno dei più grandi successi letterari contemporanei. Non c'è da stupirsi, quindi, che nel momento in cui gli venne in mente di creare un gioco per computer dalla sua Guida Galattica per Autostoppisti pensò subito alla Infocom. La persona scelta per affiancare Douglas nella stesura del gioco fu Steve Meretzky, perché in quel momento era libero e perché era il creatore di Planetfall, un'avventura di fantascienza condita d'umorismo. Il lavoro di scrittura si svolse tra gli Stati Uniti e l'Inghilterra. Douglas Adams scrisse quasi tutto il testo, mentre Steve fece quasi tutta la programmazione. Nel gioco interpretiamo Arthur Dent, e l'inizio è come nel libro, ma poi la storia segue un percorso diverso.

Nello stesso periodo uscì un nuovo gioco di Dave Lebling: Suspect. E' la notte di Halloween e noi ci troviamo a questa festa in costume, una festa molto esclusiva tenuta dagli Ashcroft. Un giornalista tra la crema della società, chi l'avrebbe creduto? Ma le cose cambiano rapidamente e veniamo accusati di un omicidio. Tutte le prove sono contro di noi, nessuno crede alla nostra innocenza. Sta a noi dimostrare di non essere colpevoli.

La fine del 1984 vide le vendite raggiungere la cifra di 10 milioni di dollari, ma qualcosa era in agguato… A Gennaio del 1985 venne pubblicato Cornerstone, ma le vendite non andarono come sperato. Dopo essere costato tanto in termini di ricerca e sviluppo il ritorno non era dei migliori.

Ispirato da una pietra che s'illumina nel buio, nel Giugno del 1985 venne pubblicato un altro gioco della serie Introductory: Wishbringer. Creato da Brian Moriarty, meglio conosciuto come Professor Moriarty, è la storia di un giovane ragazzo della cittadina di Festeron che si occupa di recapitare le lettere per l'ufficio postale locale. Ma che succede il giorno in cui ci troviamo a consegnare una lettera al Ye Olde Magick Shoppe che si rivela essere una richiesta di riscatto del misterioso Evil One? Il rapito è il gatto della padrona del negozio, e noi le promettiamo di ritrovarlo. Uscendo dal negozio, però, ci accorgiamo che il mondo di Festeron è cambiato in qualcosa di sinistro…

Nel frattempo alla Infocom stavano lavorando ad un'evoluzione della loro macchina virtuale, la Z-Machine, che avrebbe consentito loro di creare dei giochi più grandi, che potevano girare sulle nuove macchine che si stavano producendo in quel periodo. Il programma che implementava questa nuova versione venne chiamato EZIP (per extended) e la linea di prodotti sviluppati con il nuovo sistema si sarebbe chiamata Interactive Fiction Plus.

Il primo gioco della serie Plus fu A Mind Forever Voyaging di Steve Meretzky. Un gioco interessante, con una geografia molto grande rispetto ai giochi più vecchi ed un vocabolario più ampio. Un gioco con meno enigmi, ma con una storia molto coinvolgente. Ed a creare l'atmosfera ci pensa anche la documentazione allegata; ci viene fornito, infatti, un racconto molto evocativo che ci guida fino all'inizio del gioco. Racconto e gioco sono un tutt'uno: nel momento in cui finisce il racconto inizia il gioco, senza soluzione di continuità. Nel ventunesimo secolo gli Stati Uniti elaborano un Piano che riavvicini la gente alla partecipazione politica, visto che il mondo si trova sull'orlo del caos, ma ci sono dei rappresentanti del governo che hanno dei dubbi sulla validità del Piano. Viene quindi creato PRISM, il primo computer dotato di coscienza. Noi siamo PRISM, e dobbiamo interagire con una simulazione di Rockvil, la cittadina in cui ci troviamo, nel futuro, per vedere che effetti avrà il Piano.

Nel Settembre del 1985, però, oltre alla pubblicazione di A Mind Forever Voyaging, la Infocom affrontò anche la sua prima crisi finanziaria dovuta al fallimento commerciale di Cornerstone. Sebbene la divisione giochi guadagnasse molto, la divisione prodotti business era in forte perdita.

Spellbreaker, il terzo della serie di Enchanter, venne pubblicato nell'Ottobre del 1985 ad opera di Dave Lebling, l'ispiratore del primo episodio. Siamo arrivati, ormai, alla guida della Gilda degli Enchanter, ma la crisi è vicina. La magia sta scomparendo dal mondo, gli incantesimi non funzionano più, e come se non bastasse i membri della Gilda sono stati trasformati in anfibi! Un gioco difficile, ma affascinante in cui Lebling è riuscito ad inserire tutto quello che avrebbe voluto fare al tempo di Enchanter.

Ballyhoo, un gioco classico, fitto di enigmi, scritto da Jeff O'Neill venne pubblicato nel Febbraio del 1986. Al termine di uno spettacolo circense, decidiamo di dare un'occhiata in giro. Ma così facendo veniamo a scoprire che la figlia del proprietario è stata rapita e che per il detective incaricato di trovarla sembra più interessante attaccarsi a una bottiglia piuttosto che cercare la ragazza scomparsa. Cosa possiamo fare, se non decidere di trovare la ragazza per nostro conto?

Prima di creare Wishbringer, Brian Moriarty aveva steso la trama per la creazione di Trinity. In quel periodo, però, il sistema extended non era ancora disponibile ed il gioco non avrebbe potuto girare sul vecchio sistema. Con l'EZIP, però, questo era possibile. Siamo a Londra, determinati a goderci la nostra vacanza, nonostante le tensioni tra le superpotenze. Ma la nostra tranquillità viene interrotta dagli allarmi di un attacco aereo, ed una bomba nucleare viene sganciata su Londra. Dobbiamo riuscire a sfuggire all'esplosione attraverso una misteriosa porta bianca, che ci conduce in un altro mondo, un mondo che, scopriamo, unisce i luoghi e i tempi in cui un'esplosione atomica si è verificata: abbiamo, così, la possibilità di cambiare il corso delle cose e modificare la storia.

A causa delle difficoltà finanziare in cui versava la Infocom, questa venne acquistata dalla

Activision, e la firma dell'acquisizione venne posta il 13 Giugno del 1986.

La fine del 1986 vide la pubblicazione di due nuovi giochi. Il primo, Leather Goddesses of Phobos, era un'altra creazione dell'infaticabile Steve Meretzky. Dopo aver affrontato un argomento serio con A Mind Forever Voyaging, era tornato al suo genere: la fantascienza condita di elementi umoristici. Siamo nel 1936 e veniamo rapiti dalle Dee in pelle di Phobos come soggetto da studiare per arrivare alla schiavizzazione dell'intero genere umano. Se riusciamo a fuggire dalla prigione in cui ci hanno rinchiuso ci troviamo a viaggiare per la galassia, in avventure condite di umorismo e di un pizzico di sesso. Nella confezione del gioco si trovano anche delle carte da grattare ed annusare in momenti specifici.

L'altro gioco era Moonmist, della coppia Stu Galley/Jim Lawrence, la stessa che ci aveva dato Seastalker. Un altro gioco di livello Introductory, che ci vede interpretare un giovane detective americano invitato in Cornovaglia dalla sua amica Tamara, perché pare che la sua residenza sia infestata dagli spiriti ed in più è convinta che qualcuno stia tramando per assassinarla.

All'inizio del 1987 venne pubblicato un gioco interessante, Hollywood Hijinx. Scritto da Dave Anderson, con la partecipazione di Liz Cyr-Jones. Lo zio Buddy era un grande produttore di film di serie B, pellicole famose, di grande successo. E noi, da piccoli, passavamo molto tempo con lui e con zia Hildegarde. Poi zio Buddy lasciò la vita terrena e tutto rimase in mano alla zia, compresa la residenza di Malibu. Alla scomparsa di zia Hildegarde veniamo nominati eredi della casa, ma ad una condizione: dobbiamo trovare dieci tesori provenienti dai vecchi film di zio Buddy e sparsi per la residenza, ma dobbiamo farlo in una notte o perderemo tutto.

Bureaucracy, un altro gioco della serie Plus, accreditato a Douglas Adams ed a generici "altri" (pare che gran parte della Infocom partecipò alla stesura del gioco). Che succede quando cambiamo lavoro ed indirizzo? Succede che parte della nostra corrispondenza continua ad arrivare al vecchio indirizzo, ed altra non arriva da nessuna parte. In più, nonostante abbiate compilato un modulo per il cambio d'indirizzo e lo abbiate spedito alla vostra banca, questa continua a mandare la corrispondenza (e gli assegni con i soldi che vi sono dovuti) al vostro vecchio indirizzo.

Stationfall, il seguito di Planetfall, segnò il ritorno di uno dei personaggi più amati, il robot Floyd. Facciamo ancora parte della Pattuglia Stellare e siamo stati promossi a tenente, ma il nostro lavoro è cambiato solo di poco. Da spazzini a passacarte. Dobbiamo andare presso una stazione spaziale per ritirare dei moduli di richiesta ed il nostro compagno di viaggio è il caro, vecchio Floyd. Una volta atterrati sulla stazione spaziale, però, scopriamo che questa è stranamente deserta. Cosa diavolo sarà successo?

Insieme a Stationfall la Infocom pubblicò anche The Lurking Horror, di Dave Lebling. Una storia horror ambientata in un'università che ricorda molto il MIT in cui i membri della Infocom lavoravano insieme all'inizio. Sembra che nei sotterranei dell'istituto si nascondano delle strane ombre, o che si sentano dei suoni raccapriccianti. C'è qualcosa da queste parti, una presenza oscura che va fermata, prima che sia troppo tardi.

Nord and Bert Couldn't Make Head or Tail of It è forse uno dei giochi della Infocom meno apprezzati. Etichettato come Interactive Short Stories ci vede impegnati in otto racconti costellati di giochi di parole ed enigmi bizzarri. Opera di Jeff O'Neil, lo stesso di Ballyhoo.

Amy Briggs scrisse Plundered Hearts, il primo gioco della Infocom in cui interpretiamo una donna (senza contare Leather Goddesses of Phobos, che ci faceva scegliere il sesso del protagonista). Un'avventura romantica tra i pirati del diciassettesimo secolo. Nostro padre viene tenuto prigioniero da Lafond, e noi dobbiamo fare di tutto per liberarlo.

Beyond Zork è stato l'ultimo gioco scritto da Brian Moriarty per la Infocom, prima di lasciare la compagnia per andare a lavorare per la Lucasfilm Games, dove creò Loom. Scritto per un nuovo avanzamento della Z-Machine, la versione 5 della stessa implementata tramite un nuovo programma chiamato XZIP (per experimental), mostrava sullo schermo una mappa semi grafica ed un'interfaccia a finestre. Ormai l'età della magia è giunta alla fine e la regione di Quendor sta cadendo nel caos. Per evitare che la magia vada completamente perduta, la Gilda degli Enchanter ci manda alla ricerca del più potente oggetto magico esistito: la Noce di Quendor, al cui interno è contenuta la saggezza della Gilda.

Sempre per la stessa versione della Z-Machine, Marc Blank scrisse una spy story: Borderzone. Una storia, divisa in tre capitoli, in cui interpretiamo tre personaggi diversi che attraversano la cortina di ferro. Tre punti di vista diversi per uno stesso evento. In più, ad aumentare la tensione, c'è il fatto che il gioco si svolge in tempo reale: i minuti passano anche se non intraprendiamo alcuna azione.

A Gennaio del 1988 venne pubblicato Sherlock. Al gioco avrebbe dovuto lavorare Brian Moriarty, ma venne portato avanti da un "esterno", Bob Bates, che in seguito avrebbe creato la compagnia Legend Entertainment. Qualcuno ha rubato i gioielli della corona, qualcuno che sa molto bene come opera Sherlock Holmes, quindi il grande detective decide di far condurre le indagini al suo fedele compagno, il dottor Watson. E noi interpretiamo proprio Watson che deve riuscire a trovare i gioielli in 48 ore, in tempo per l'inizio dei festeggiamenti per il giubileo.

Per la fine del 1988 la Infocom sviluppò una nuova versione della sua macchina virtuale, la sesta. In grado di gestire anche la grafica, portava con se anche un'evoluzione del parser.

Il primo gioco che venne pubblicato per questa nuova versione fu Zork Zero, scritto da Steve Meretzky. Ambientato cronologicamente prima della trilogia originale di Zork, dobbiamo annullare la maledizione che minaccia la sopravvivenza del regno. Pieno di grafica e di enigmi, è forse il meno ispirato degli Zork. E l'ultimo gioco che Steve Meretzky scrisse per la Infocom.

L'ultimo gioco di Dave Lebling per la Infocom fu Shogun. Un altro gioco grafico, è l'adattamento dell'omonimo romanzo di James Clavell. Accompagnato da illustrazioni che imitano le stampe giapponesi, e da un ottimo testo, ha degli enigmi lineari che poco aggiungono all'esperienza di gioco.

Journey, previsto come il primo di una trilogia, è un esperimento di gioco a scelta multipla. Scritto da Marc Blank, ha una buona trama, ma rispetto agli altri giochi c'è poca libertà d'azione.

Nel frattempo, a Maggio del 1989, la Activision offrì ai pochi membri della Infocom la possibilità di trasferirsi in California. Solo 5 di loro accettarono, e tra questi nessuno dei più famosi.

L'ultimo gioco della Infocom venne pubblicato nel Luglio del 1989: Arthur. Scritto da Bob Bates, era un altro gioco grafico, ma era possibile giocare anche senza immagini. Nel gioco interpretiamo un giovane Artù, ma non siamo noi a estrarre la spada dalla roccia, bensì il malvagio Re Lot.

Conclusioni

Nel mondo delle avventure testuali non c'è stata un'altra compagnia che abbia creato così tanti prodotti e tutti ad un livello qualitativo così alto. La costante ricerca per il miglioramento dell'interazione con i giochi, la cura impiegata nella scrittura dei giochi, con la stesura delle trame e la discussione delle stesse con più persone, l'evoluzione continua ed il testing approfondito: tutti elementi che hanno portato la Infocom ad essere un punto di riferimento per questo genere di giochi. Ed il fatto che gli stessi siano ancora oggi perfettamente giocabili su un'infinità di computer, e che mantengano la freschezza e l'originalità dopo più di vent'anni dimostra quanto questo tipo d'intrattenimento digitale abbia ancora qualcosa da dire, e quanto la Infocom fosse una compagnia all'avanguardia per i suoi tempi.

Riferimenti

Tim Anderson, *The History of Zork, part 1*, The New Zork Times, vol. 4 no. 1, Winter 1985

Tim Anderson, *The History of Zork, part 2*, The New Zork Times, vol. 4 no. 2, Spring 1985

Stu Galley, *The History of Zork — The Final (?) Chapter: MIT, MDL, ZIL, ZIP*, The New Zork Times, vol. 4 no. 3, Summer 1985

P. David Lebling, Marc S. Blank, Timothy A. Anderson, *Zork: A Computerized Fantasy Simulation Game*, IEEE Computer, vol. 12 no. 4, April 1979

Marc S. Blank, S. W. Galley, *How to Fit a Large Program Into a Small Machine*, Creative Computing, July 1980

P David Lebling, *Zork and the Future of Computerized Fantasy Simulations*, Byte, December 1980

Mike Gerrard, *Interview at the end of the universe*, Atari ST User, May 1987

Sean Masterson, *Four Minds Forever Voyaging*, ZZAP! 64, May 1986 e June 1986

Cathy Yakal, Marc Blank: *The Programmer Behind Zork*, COMPUTE!'s Gazette, October 1983

Brian Moriarty, *The Wizard of Wishbringer,* AmigaWorld, January/February 1986

Paul David Doherty, *Infocom Fact Sheet*

Russ Ceccola, *Adventures at Infocom*, Commodore Magazine, January 1988

Un'introduzione a Inform 7

di Giovanni Riccardi

Il 30 aprile è una data importante per l'Interactive Fiction. In prossimità di quel giorno, ogni anno, si crea questa strana agitazione nella comunità internazionale nell'attesa che Graham Nelson, matematico, poeta, creatore del linguaggio di programmazione più utilizzato per scrivere IF e ormai quasi un guru per tutti gli appassionati, si risvegli dal letargo che normalmente lo tiene lontano dalla comunità per interi mesi e annunci qualche novità su rec.arts.int-fiction. Lo scorso 30 aprile è stato presentato Inform 7, evoluzione del famoso linguaggio, che nelle intenzioni dell'autore dovrebbe rivoluzionare il modo di scrivere IF.

Il 30 aprile è il compleanno di Inform.
Nato il 30 aprile 1993, il linguaggio si è evoluto in maniera costante per qualche anno, anche grazie al contributo di una piccola ma attiva comunità di appassionati che lo hanno di fatto reso il più utilizzato nel mondo della IF. L'ultima revisione, la sesta, è però datata 30 Aprile 1996. E se nel corso degli anni successivi la libreria che lo accompagna è stata revisionata ben 11 volte (da Graham stesso o più recentemente da un team dei più esperti programmatori Inform), per 10 anni il linguaggio è rimasto lo stesso senza neanche un piccolo ritocco.
Dopo così tanto tempo, tutti ci aspettavamo qualche novità. Il 30 aprile scorso l'attesa è finita, ma le novità che la settima versione di Inform ha portato con sé non erano assolutamente prevedibili, per non citare lo strascico di polemiche seguito all'annuncio.

In questo articolo vi presentiamo alcune delle novità introdotte da Inform 7. Non vuole essere certamente una descrizione completa ed esaustiva di quelle che sono le sue caratteristiche. Maggiori informazioni si possono trovare sul sito ufficiale (http://www.inform-fiction.org) dal quale potete scaricare l'ambiente di scrittura e la documentazione, mentre una introduzione più completa di questa è senza dubbio quella fatta da Stephen Granade su Brass Lantern ("Introducing Inform 7" su http://brasslantern.org/writers/howto/i7intro.html).

Con Inform 7 non è ancora possibile scrivere avventure in Italiano: I7 è ancora in beta e G. Nelson vuole prima completare il lavoro sulla versione inglese e solo dopo iniziare a lavorare sulle altre lingue.

L'ambiente di scrittura e il linguaggio naturale

La prima cosa che salta all'occhio una volta installato Inform 7 è il nuovo ambiente di creazione dell'avventura: Inform non è più solo un compilatore da riga di comando, non bisogna più scrivere la IF in un editor e poi richiamare il programma per la creazione del file da giocare sull'interprete. In I7 tutto avviene all'interno dell'ambiente di scrittura (per ora disponibile solo per Mac OSX e Windows) che contiene numerosi strumenti che aiutano l'autore nella creazione e nel test della propria avventura: dalla documentazione alla generazione della mappa fino all'interprete integrato.

Ma è proprio il linguaggio di programmazione la vera novità: I7 e Inform 6 sono solo lontani parenti e se quest'ultimo è un linguaggio di programmazione tradizionale, con una sintassi simile al C e qualche elemento di programmazione orientata agli oggetti, la nuova

creatura di G. Nelson è basata invece su un linguaggio naturale, un sottoinsieme della lingua Inglese con cui possiamo descrivere gli oggetti della nostra IF, le loro relazioni e le regole di comportamento.

Ecco come si presenta una semplice IF scritta in Inform 7.

```
"Terra D'IF" by Giovanni Riccardi. The story headline is "Un'avventura di
presentazione del nuovo Inform 7"

Redazione is a room. "Sei nella redazione di Terra D'IF".
```

E questo è il risultato dopo la compilazione:

```
Terra D'IF
Un'avventura di presentazione del nuovo Inform 7 by Giovanni Riccardi
Release 1 / Serial number 061025 / Inform 7 build 3Z95 (I6/v6.31 lib 6/11N) SD

Redazione
Sei nella redazione di Terra D'IF

>
```

Praticamente nel codice sorgente abbiamo descritto gli elementi della IF (il titolo, l'autore, una stanza), ma lo abbiamo fatto nello stesso modo in cui lo faremmo parlando ad un amico piuttosto che a un compilatore. Nel codice sorgente più sopra sono stati evidenziati (in grassetto) gli elementi del linguaggio mentre tutto il resto fa parte della nostra IF.

Ma aggiungiamo qualche altro elemento a questa IF di prova:

```
Redazione is a room. "Sei nella redazione di Terra D'IF. A est c'è una porta
aperta che conduce all'ufficio di Roberto Grassi.".

Ufficio del direttore is east of Redazione. "L'ufficio di Roberto è il classico
ufficio del direttore di una rivista: giornali, libri e fogli di carta sparsi un po'
dovunque."

Roberto Grassi is here.
```

Abbiamo creato una nuova stanza senza specificarlo (non abbiamo scritto "Ufficio del direttore is a room", anche se avremmo potuto farlo) ma semplicemente dicendo in che posizione si trova rispetto ad una stanza precedentemente indicata. Questa è una delle particolarità del nuovo Inform, una caratteristica che può confondere all'inizio, ma che è molto interessante: come nel linguaggio naturale possiamo dire la stessa cosa in modi diversi così in Inform 7 possiamo descrivere gli elementi della IF in modi diversi a seconda del contesto (non è sempre vero, perché alcune volte I7 non riesce a capire le indicazioni che gli vengono date e quindi bisogna quindi essere più espliciti). Lo scopo ultimo è che il

codice sorgente sia leggibile se non proprio come un racconto, il più vicino possibile a questo. Ecco il risultato:

> **Redazione**
> Sei nella redazione di Terra D'IF. A est c'è una porta aperta che conduce all'ufficio di Roberto Grassi.
>
> >e
>
> **Ufficio del direttore**
> L'ufficio di Roberto è il classico ufficio del direttore di una rivista: giornali, libri e fogli di carta sparsi un po" dovunque.
>
> You can see Roberto Grassi here.
>
> >

Volendo scrivere una IF in Italiano, il "codice" da scrivere è certamente un po' strano (in un thread su RAIF dello scorso maggio l'ho chiamato "spaghetti english", un misto di inglese e italiano che suona molto da film di gangster anni '20) e se non si è espliciti nelle descrizioni, I7 utilizza ovviamente quelle della sua libreria che è in Inglese (in questo caso anche i comandi che diamo al parser dell'avventura sono in inglese). Ci basta aggiungere una descrizione:

```
Roberto Grassi is here. "Roberto Grassi è seduto alla sua scrivania. Sembra un po' arrabbiato. Sta sbraitando qualcosa sulla data di pubblicazione della rivista..."
```

per avere un risultato più leggibile:

> **Ufficio del direttore**
> L'ufficio di Roberto è il classico ufficio del direttore di una rivista: giornali, libri e fogli di carta sparsi un po' dovunque.
>
> Roberto Grassi è seduto alla sua scrivania. Sembra un po' arrabbiato. Sta sbraitando qualcosa sulla data di pubblicazione della rivista...

I7 comprende l'inglese?

Ovviamente la risposta a questa domanda è no. Non stiamo parlando di intelligenza artificiale, ma di una interpretazione di quello che viene scritto nel codice sorgente. Se I7

non riesce a decidere su una cosa che abbiamo scritto, ci chiede di essere più espliciti nelle descrizioni e nelle regole.

Per esempio, se scrivo:

```
Roberto Grassi wears un cappello.
```

I7 "capisce" che Roberto Grassi è una persona perché solo le persone possono indossare (wears) qualcosa. Se invece scrivo:

```
Roberto Grassi carries un cappello.
```

I7 non può dire nulla circa Roberto Grassi: anche un carro (oggetto inanimato) può portare (carries) qualcosa.

Con le frasi in linguaggio naturale, I7 riesce anche a costruire le relazioni tra gli oggetti. Questa costruzione è implicita (e "naturale"):

```
Un tavolo is in Cucina. On it is una tazza di the.
```

Poiché abbiamo scritto "on it", automaticamente "un tavolo" diventa un supporter senza doverlo specificare. Anche nel codice scritto precedentemente:

```
Ufficio del direttore is east of Redazione.
```

Inform 7 può trarre le sue conclusioni senza particolari dubbi. Prima di tutto che "Ufficio del direttore" è una stanza e poi che si trova a est di "Redazione": solo una stanza può infatti trovarsi a est di qualcosa che sappiamo già essere una stanza (potrebbe essere anche una porta, ma in quel caso l'avremmo dovuto specificare).

Rivoluzione?

Come già scritto, questa è solo una breve introduzione a Inform 7, soprattutto per quelli che non hanno avuto tempo di seguire le discussioni su RAIF o su ICGAT e di approfondire i dettagli del linguaggio. Molti elementi (uno su tutti, le regole) non sono stati trattati anche e soprattutto per via del fatto che non è ancora possibile usare il nuovo linguaggio con l'italiano.

L'idea alla base dell'utilizzo del linguaggio naturale nasce dalla volontà di aprire il mondo dell'IF anche ai non programmatori. Ci sono ovviamente diverse scuole di pensiero: i detrattori di questo approccio criticano il fatto che il linguaggio naturale di I7 è molto più prolisso dell'equivalente Inform 6, ma soprattutto l'ambiguità di certe costruzioni sintattiche. Non scendiamo nelle questioni tecniche di queste critiche, ma ovviamente ci sono anche quelle e sotto certi punti di vista non sono neanche secondarie.

Nel frattempo, I7 continua il suo percorso, con aggiornamenti costanti che lo stanno pian piano facendo diventare un prodotto maturo. La comunità (internazionale) pare aver recepito bene questa rivoluzione: in pochi mesi diversi giochi sono già scaricabili da IF Archive e più di venti estensioni di terze parti sono disponibili sul sito ufficiale.

Una versione italiana è in fase di studio: già adesso con qualche modifica ai file delle librerie si riesce a utilizzare I7 per scrivere IF in italiano (compilate quindi con INFIT) anche se con un linguaggio naturale che è ovviamente in inglese. Lo scoglio più grande sarà tradurre tutto il linguaggio nella nostra lingua in modo da poter scrivere frasi come

“Redazione è una stanza” oppure “Roberto Grassi indossa un cappello” (al momento non è possibile visto che G. Nelson non ha rilasciato ancora il sorgente dei compilatori), ma anche considerando le critiche che vengono da più parti della comunità italiana, Inform 7 introduce tante e tali novità che probabilmente questa è la direzione giusta per diffondere la IF anche fuori dal gruppo di appassionati.

(Ogni riferimento a cose e persone è puramente casuale)

Recensioni: 1-2-3...

di Chris Mudd

recensita da Roberto Grassi

Un gioco breve ed interessante che meritava una migliore fase di rifinitura. Ho gradito in particolare lo stile narrativo 'fumettistico' e il gameplay 'a vignette'. L'atmosfera inquietante è ben resa ed i cambi frequenti di scena e di punto di vista sono funzionali agli scopi narrativi, dando un gradevole senso di stordimento.

La storia ci parla di un serial killer e di un ispettore che indaga su di lui. È davvero molto semplice e, in realtà, la storia in sé e la caratterizzazione non sono i migliori aspetti di questo gioco. Quello che funziona davvero bene sono le atmosfere 'ossessive' ed 'erotiche' che lo pervadono, che conducono ad un finale interessante e, per certi versi, sorprendente.

La cosa meno riuscita nel gioco consiste nello stile di narrazione con i personaggi non giocanti, che è davvero mal implementato (ci sono altri errori minori di design, ma non sono così importanti per lo stile di questo gioco). Se consideriamo che si tratta di un'IF 'puzzleless' e che è basata soprattutto sulla narrazione, è evidente che si tratta di un errore molto grave.

Per concludere, un gioco gradevole ed interessante, che suggerisco di giocare. Probabilmente meritava una posizione migliore nell'IFComp del 2000.

Recensioni: All alone

di Ian Finley

recensita da Roberto Grassi

Un gioco corto, bello, ben sviluppato e con un design molto rifinito. L'atmosfera di tensione sale dall'inizio alla fine, terminando in un finale improvviso che potrebbe lasciarvi sorpresi ma che è funzionale al replay del gioco.

Impersoniamo una donna, sola nel suo appartamento, mentre la televisione annuncia gli ennesimi omicidi brutali di un serial killer di donne che sono… sole nel loro appartamento, come noi. Il telefono squilla in quell'istante e sentiamo una voce che ansima dall'altro lato della cornetta…

Il mio consiglio è: giocatelo (e rigiocatelo).

Intervista a Marco Zanchi

(CIPA Informatica)

di Marco Falcinelli

Presentati al nostro pubblico e dicci qualcosa di te...
Mah, credo di essere un tecnico eclettico, grazie all'esperienza fatta ai tempi di queste avventure; una passione si è trasformata in un lavoro....
Ora ho 38 anni e come allora mi piace fare un po' di tutto. Le passioni sono un po' cambiate e la più grande è ora il motociclismo. Mi dedico alle piste e quest'anno ho corso un campionato monomarca Ducati, un challenge, sulla mia 999. Una bell'esperienza che consiglio...

Se non siamo indiscreti, di cosa ti occupi oggi?
Ovviamente d'informatica e mi divido tra formazione come IBM Trainer Ufficiale e sviluppo software e consulenza architetturale su piattaforme J2EE.

Come hai scoperto i computer? E com'è nato il tuo interesse per le avventure testuali?
Da ragazzo, un VIC20 invece di un motorino e poi è scoppiata la passione. In particolare, scrivere software ti permetteva di "creare", dare la vita a un codice che "funzionava", e a quell'età era una gran soddisfazione.
Le avventure testuali erano quasi un obbligo: univano il mondo fantastico alle capacità limitate delle macchine di allora e di chi non riusciva a sviluppare granché in grafica....

Qual è la tua avventura preferita?
C'era un gioco per il VIC20 - o forse per il C64? - che si chiamava TOMB OF DREWAN.... era mitico! poi, credo, Mission Impossible ma era già un arcade/platform...

[nota per i curiosi: http://www.xs4all.nl/~itsux/tomb.html]

Tu sei l'uomo dietro CIPA INFORMATICA: com'è nata la software house? Ti occupavi solo d'avventure testuali? Ti guadagnavi da vivere scrivendo avventure?
Eravamo due amici: facevamo un po' tutto quello che capitava nel software, dai gestionali ai giochi. Studiavamo ancora, ma ci mantenevamo da soli e non era facile, così facevamo e inventavamo davvero di tutto: abbiamo scritto uno dei primi antivirus, un sistema di protezione, molto per gli home computer, molto per i primi PC.

Parlaci dei tuoi giochi, le idee... il metodo... gli strumenti...
Le avventure per PC che scrivevamo erano trasposizioni di giochi messi in piedi su MSX in Basic. Ci servivano giochi per una rivista che avevamo creato e che si chiamava PC PLUS: è stato il primo periodico per PC con floppy allegato.... Era faticoso ma divertente: un po' raccoglievamo shareware, un po' scrivevamo noi il software per essere originali... E funzionava....! Andavo ancora a scuola e scrivevo dei mini arcade in HiRes a due colori durante il laboratorio di matematica... In QBASIC!!! Che tempi!

Per quanto tempo hai programmato avventure professionalmente? Quante ne hai realizzate? C'è qualche titolo a cui sei più affezionato?
Come dicevo erano una trasposizione per PC e ne abbiamo fatte almeno una dozzina. C'era un titolo a cui ero affezionato ma ora non ricordo come lo chiamammo. Cominciava su un astronave da esplorare, poi c'erano i soliti alieni che mi mangiavano quasi subito. Era divertente, mi ricordo ancora ora i comandi....

Segui/giochi ancora le avventure testuali? Cosa pensi delle ultime uscite e dei nuovi autori italiani (Lucchesi, Cordella, Niccolai etc...)?
Purtroppo è un mondo che non ho seguito, ma dopo aver scoperto il vostro sito sono un po' curioso. Ora mi diletto con la mia XBOX, la preferisco ai giochi per PC perché sono macchine dedicate, e al mio GameBoy SP. Per i giochi XBOX vado a periodi: ora ho ripreso in mano ProjectGotham2 e lo sto completando. Per GBA uso una Flash in cui scarico delle Rom e le cambio spesso...

Hai mai pensato in tutti questi anni di riprendere a programmare avventure anche solo per hobby e per passione?
Ho pensato di ricominciare a scrivere giochi in generale, soprattutto quando Java è apparso sui telefonini. Devo dire che ho usato spesso i giochi come metodo per imparare a sviluppare in un certo linguaggio, per esempio con Visual Basic. Il primo software che ho realizzato è stata una cover di Space Invaders. Quando scrivi un gioco sfrutti al massimo tutte le caratteristiche del sistema, così sei obbligato ad imparare... Ora il nemico è il tempo: già non riesco a scrivere il codice che dovrei per i clienti, figurati quello per i giochi che sono solo diletto....

> CAVOLO! Bonaventura! Saranno passati quasi vent'anni dall'ultima volta che l'ho visto...

Hai appena ammesso di conoscere Bonaventura, raccontaci un aneddoto...
Mah, non ricordo granché.... Era simpatico, mi ricordo una pettinatura un po' strana... era un tipo Naif....

Un grazie di cuore a Marco per la sua cortese disponibilità.

Design Review: Locazioni irraggiungibili

di Roberto Grassi

In questa puntata vorrei parlare di un aspetto, che potrebbe apparire secondario, ma che invece rappresenta una finezza dal punto di vista del Game Design. Mi riferisco alle risposte che vengono fornite al giocatore quando cerca di spostarsi in locazioni che non possono essere raggiunte.
L'impossibilità di raggiungere la locazione di arrivo può essere di vario genere e dipende, innanzitutto, dal posizionamento dell'ipotetica locazione d'arrivo all'interno (o all'esterno) della mappa del gioco.
In questo senso, si può distinguere in:

- Spostamenti da locazioni al bordo della mappa verso locazioni non mappate, verso l'esterno del mondo di gioco. Si tratta delle "mosse ai bordi". Supponiamo che il giocatore si trovi nella parte più periferica del mondo che abbiamo creato, in una locazione che ha locazioni connesse a nord e ad est. Se il giocatore prova a digitare *"sud"* e *"ovest"* il gioco si trova a dover gestire la "mossa ai bordi", la potenziale uscita dalla mappa di gioco. Approfondirò le risposte possibili più avanti, nel corso di quest'articolo, perché scegliere una risposta appropriata può non essere così banale.
- Spostamenti tra locazioni interne alla mappa. In questo caso l'impossibilità allo spostamento può essere determinata, ad esempio, da un ostacolo fisico, come una parete, o dal fatto che ci troviamo di fronte ad una porta chiusa che deve essere aperta.

La "mossa ai bordi"

Ovviamente il mondo dove il gioco si svolge è per forza finito, e di conseguenza prima o poi il giocatore arriverà al suo confine. Quindi l'autore deve in qualche modo gestire l'impossibilità di superare quella ideale linea rossa tratteggiata oltre la quale non si può andare. Ovviamente la soluzione più ovvia è quella di usare ostacoli fisici, naturali o artificiali, per bloccare il passaggio. E questo può andar bene per la quasi interezza del perimetro, o addirittura, in certi casi, per la tua interezza; se ad esempio il gioco si svolge su di un'isola, il mare è un ottimo confine naturale. Ma in generale il giocatore si aspetterà che ci siano delle vie di collegamento dal mondo di gioco al "mondo esterno", e l'autore dovrà giustificare il fatto che quelle vie siano inagibili. Vediamo alcune soluzioni:

- Motivazione "etica": il personaggio mosso dal giocatore non può allontanarsi dal mondo di gioco a causa dei suoi doveri e viene fermato da risposte come *"Non puoi andartene ora"* o *"Hai una missione da compiere"*. È una tecnica universalmente accettata, anche se abusata.
- Motivazione "di trama": l'uscita dal mondo di gioco è bloccata da un impedimento strettamente correlato alla trama principale del gioco, e risolvere l'avventura è l'unico modo per liberare il passaggio verso l'esterno. Un classico delle avventure di genere "horror" in cui la porta della casa si apre solo alla fine.
- Fine del gioco: attraversare il confine del mondo implica la fine del gioco; in pratica il personaggio muore o, più spesso, rinuncia alla propria "missione" e, in genere, il giocatore non può ritenere di aver risolto l'avventura. Per fare un esempio, è quello che accade se il giocatore ridiscende dalla cantina in *Curses.*

- Fine del gioco condizionata: a metà strada tra le tre precedenti, il giocatore è tenuto all'interno del gioco da motivazioni etiche o di trama, e deve risolvere l'avventura prima di allontanarsi; il gioco termina favorevolmente quando il giocatore lascia il mondo di gioco dopo aver compiuto il proprio dovere.
- Uscita illusoria: il personaggio può anche lasciare il mondo di gioco, ma un qualche meccanismo lo costringerà a rientrare; ad esempio potrà trovarsi su di una strada infinita (come in *Enchanter*) o perdersi nella nebbia e ritrovarsi esattamente dov'era.

Ovviamente questi sono solo degli esempi. L'importante, comunque, è che il giocatore deve percepire di aver raggiunto i limiti del mondo senza perdere quella che si chiama "sospensione dell'incredulità" e senza sentirsi imbrogliato.

Spostamenti interni al mondo

Le cose sono invece diverse se il protagonista si trova all'interno del mondo di gioco, dove si aspetta di potersi muovere liberamente. Un giocatore, per quanto esigente, non potrebbe davvero pretendere che il mondo sia infinito, ma può aspettarsi che al suo interno il mondo sia completo, completamente realizzato. Ciò nonostante, possono e devono esistere motivi per cui alcune vie siano impercorribili.
Talvolta questi impedimenti saranno definitivi, ma altre volte no. Altre volte potranno venir meno o per opera del giocatore o, più raramente, indipendentemente dalla sua volontà.

Il caso più semplice e comune è quello di un impedimento fisico, una barriera solida e reale; essa può essere, ad esempio, una parete, uno specchio d'acqua, un burrone, un bosco troppo fitto, il fianco estremamente scosceso di una montagna. Rappresenta ovviamente la normalità se l'azione si svolge all'interno di un edificio, o al limite in un fitto agglomerato di case, tanto che il gioco non deve in genere giustificare il fatto che non ci si possa muovere in una certa direzione. Per contro all'esterno, in "campo libero", è bene che la risposta al movimento in una certa direzione spieghi chiaramente i motivi per cui questo è impossibile (come *"Non sai nuotare"*, *"Gli alberi sono troppo fitti"* o *"Ti avvicini al ciglio del burrone, guardi verso il basso e desisti dal tuo intento"*). La totalità o la quasi totalità degli ostacoli di questo tipo incontrati in un'avventura saranno definitivi, ma ovviamente le eccezioni sono sempre possibili (un passaggio segreto, un ponte che appare sul baratro, attrezzi da scalatore).

Molto comuni sono anche gli impedimenti temporanei che il protagonista può attivamente risolvere. Sebbene siano più rari rispetto agli ostacoli fisici, sono però rispetto ad essi più facilmente "percepibili"; il giocatore farà ben poco caso ad una parete di mattoni, ma la sua attenzione sarà sicuramente attirata da una porta che deve essere aperta (ovvero l'esempio più classico di questa categoria), o da un passaggio in cui una trappola uccide il personaggio. Per questo motivo questi ostacoli sono, tra tutti, quelli che necessitano della maggior cura.
In genere, se la rimozione dell'ostacolo dipende direttamente dal giocatore (cosa che può rappresentare un enigma, ma questo non è necessario), una volta che esso è stato rimosso le direzioni che erano bloccate rimangono accessibili (a meno che il giocatore stesso non le blocchi nuovamente). Inoltre, se rimuovere un ostacolo è un'azione banale, allora è bene che essa venga in qualche modo resa automatica; ad esempio, è estremamente fastidioso per il giocatore dover aprire continuamente porte (non chiuse a chiave), e sarebbe quindi meglio implementare un meccanismo d'apertura automatica.
Se invece la rimozione dell'ostacolo non dipende dal giocatore, questo può anche riapparire; possiamo definire questo caso come "impossibilità intermittente". Un esempio

può essere rappresentato da un ascensore che si muove automaticamente (aprendo e chiudendo le sue porte) tra un piano e l'altro di un edificio. In questo caso l'autore deve far attenzione a descrivere con sufficiente chiarezza quello che sta accadendo, in modo che il giocatore non rimanga frastornato.

Esiste anche un altro caso: quello di un impedimento definitivo travestito da impedimento temporaneo, come una porta impossibile da aprire e che in realtà non conduce da nessuna parte (ma magari permette ad un png di entrare ed uscire di scena). Questa è una scelta che deve essere fortemente giustificata; l'autore deve usare questo espediente con molta cautela e solo se strettamente necessario. Il giocatore si aspetta infatti di poter esplorare tutto il mondo senza restrizioni, e potrebbe di conseguenza potrebbe molto tempo cercando il modo di attraversare quel passaggio.

Messaggi di risposta

Per concludere questo articolo spendiamo due parole sui messaggi che il gioco deve stampare quando il giocatore tenta di andare in una direzione non consentita. L'argomento sembra banale, ma è fin troppo facile, specialmente in avventure mal curate, imbattersi in messaggi oltremodo generici come *"Non puoi andare da quella parte"*. Ci sono molte altre possibilità, tutte più soddisfacenti per il giocatore; tra queste elenchiamo le due più comuni:

- Se c'è un motivo particolare e non ovvio per cui il giocatore non può muoversi in una certa direzione, è quasi obbligatorio che il programma risponda spiegando questo motivo, ad esempio dicendo *"La porta è chiusa"*, *"Gli scatoloni di cartone ti bloccano il passaggio"*, o *"Non hai le ali"*. Questo tipo di risposta può essere usato anche se l'impedimento è banale, ma un messaggio come *"C'è una parete verso est"* non è molto più brillante di *"Non puoi andare da quella parte"*.
- In alternativa, se il giocatore si muove in una direzione "sbagliata", il programma può rispondere elencando in qualche modo tutte le uscite possibili (ad esempio: *"Il corridoio procede da nord a sud, una porta conduce verso est e una scala a chiocciola porta al piano inferiore"*). Risposte di questo tipo risultano estremamente comode per il giocatore. In particolare un autore può decidere di non includere la lista delle uscite nella descrizione della stanza (cosa che è in effetti abbastanza "antimimetica"), ma in questo modo costringe il giocatore ad andare per tentativi e provare tutte le direzioni (e ciò è male); questo metodo offre un elegante compromesso: le uscite non sono citate nella descrizione, ma vengono citate appena il giocatore prova a muoversi in una direzione non valida.

Riassumendo, abbiamo visto che quest'aspetto dell'implementazione, che forse poteva sembrare di secondaria importanza, se ben curato può far molto per la coerenza dell'ambientazione e per la giocabilità della storia. Ed è solo uno dei molti aspetti che distinguono una buona implementazione, un'avventura ben curata.

INFORM: Inseguiti e inseguitori

di Paolo Lucchesi

Se in un'avventura abbiamo degli npc (non playing characters - personaggi non giocanti) che si muovono da una locazione all'altra, potrebbe essere comodo implementare un comando che permetta al giocatore di seguire un altro personaggio nel suo girovagare, esattamente come Arthur Dent può seguire il robot Marvin da un ambiente all'altro della Cuore d'Oro. Il problema non è banale, e comporta diversi problemi; normalmente, ad esempio, quando vogliamo seguire un personaggio questo si è già allontanato e, di conseguenza, non è più in scope.
In questo articolo vedremo passo dopo passo come si può implementare questa funzione.

Come prima cosa, abbiamo bisogno di una struttura dove memorizzare una lista degli npc che possono essere seguiti e, per ognuno di essi, l'indicazione di come il gioco deve reagire alla richiesta di seguire quel personaggio; questa può essere una direzione in cui muoversi, una stringa da stampare (ad esempio per dire che è inutile tentare un inseguimento) o una routine che può fare virtualmente ogni altra cosa.
Quindi definiamo una classe di oggetti che contenga queste informazioni:

```
Class Follow(8)
      with id 0,
      op 0,
      create [x y;
            self.id = x;
            self.op = y;
            move self to followstore;
      ];
```

dove id conterrà l'npc seguito, e op l'operazione da compiere; la funzione create verrà chiamata automaticamente tutte le volte che creeremo un oggetto di classe `Follow` e, come si vede, si occupa di assegnare i valori alle proprietà `id` e `op`. Nella prima riga notiamo che possiamo avere al massimo 8 oggetti di questo tipo, ma dovrebbero essere sufficienti.
Abbiamo anche bisogno di un contenitore per tenere gli oggetti di classe `Follow` creati; basta un semplicissimo oggetto, senza proprietà o altro.

```
Object followstore;
```

Se andiamo a vedere la definizione della classe `Follow`, notiamo che tutte le volte che un oggetto viene creato, viene automaticamente spostato in `followstore`.

Per evitare problemi, e per poter riutilizzare i nostri oggetti della classe Follow, vogliamo anche cancellare la lista tutte le volte che il giocatore si sposta da una stanza all'altra. Per fare ciò usiamo la routine entry-point `NewRoom`, che viene appunto chiamata tutte le volte che il giocatore cambia locazione, e in essa distruggiamo tutti gli oggetti di classe Follow

che si trovano nell'oggetto followstore, svuotando così la lista.

```
[ NewRoom x;
      objectloop (x in followstore && x ofclass Follow)
            Follow.destroy(x);
      rfalse;
];
```

Veniamo ora alla parte più delicata. Siccome vogliamo poter seguire anche npc che non sono più presenti (ma che sono memorizzati nella nostra lista) dobbiamo definire una routine dedicata che permetta di ridefinire le regole di scope aggiungendo potenziali inseguiti.

```
[ FollowScope x;
      if (scope_stage==1) rfalse;
      if (scope_stage==2) {
            objectloop (x in followstore && x ofclass Follow)
                  PlaceInScope(x.id);
            rtrue;
      }
      L__M(##Miscellany,30);
];
```

e definiamo, assieme agli altri nostri verbi, anche il verbo `segui` in modo tale che possa riferirsi a oggetti presenti o a oggetti artificialmente aggiunti allo scope dalla nostra routine.

```
Verb 'segui' 'insegui'
  * scope=FollowScope                          -> Follow
  * noun                                       -> Follow;
```

In questo modo, come si legge nel capitolo §32 del DM4, se il giocatore immette il comando *SEGUI QUALCUNO*la routine `FollowScope` viene chiamata tre volte, con tre diversi valori della variabile `scope_stage`:

- `scope_stage == 1`: serve a capire se il comando può essere usato con oggetti multipli (*SEGUI TUTTO*); non è il nostro caso, e quindi la routine restituisce false
- `scope_stage == 2`: la fase più importante, quella in cui si definisce quali oggetti vengono aggiunti allo stage; nel nostro caso usiamo `PlaceInScope` per aggiungere tutti gli oggetti per cui abbiamo creato un oggetto di classe `Follow`
- `scope_stage == 3`: serve a stampare un eventuale messaggio di errore; nel nostro caso usiamo la funzione `L__M(##Miscellany,30)` per stampare il messaggio *Non vedi nulla del genere*

A questo punto il più è fatto. Creiamo una routine Sub per il verbo *SEGUI*

```
[ FollowSub x;
      objectloop (x in followstore && x ofclass Follow) {
            if (noun == x.id) {
                  if (metaclass(x.op) == Routine) { x.op();  rtrue; }
```

```
                    if (metaclass(x.op) == String) print_ret x.op;
                    if (parent(x.op) == Compass) < >;
                }
        }
    if (parent(noun) == location or player) {
        if (noun has animate) print_ret (The) noun, " ",
              (isorare) noun, " già qui.";
        "Non @`e qualcosa che tu possa seguire.";
    }
    "Non sai dove sia.";
];
```

La routine cerca nei nostri oggetti di classe `Follow` se ce n'è uno corrispondente al nostro inseguito. Se lo trova controlla la proprietà `op` dell'oggetto di classe `Follow` eseguendola se è una routine, stampandola se è una stringa o muovendo il giocatore se è una direzione. Se invece l'oggetto non è nella lista, allora la routine stampa un messaggio d'errore appropriato.

A questo punto abbiamo finito l'implementazione di tutte le alchimie per realizzare ciò che ci eravamo prefissi. Vediamo però come usarle.

Se un npc si allontana in una direzione ben precisa, ci è sufficiente usare il comando `Follow.create(NPC, direzione)`. Ad esempio:

```
move valeria to giardino;
Follow.create(valeria, w_obj);
"Valeria esce in giardino.";
```

e allora il comando *SEGUI VALERIA* porterà anche il giocatore in giardino (ammesso, ovviamente, che il giardino si trovi ad ovest).

Se invece vogliamo che al comando *SEGUI* il gioco risponda con un messaggio, useremo il comando `Follow.create(NPC, stringa)`. Ad esempio:

```
remove littlejohn;
Follow.create(littlejohn, "Il bosco è troppo fitto, non puoi riuscire
a stargli dietro.");
"Little John ti saluta e fugge tra gli alberi.";
```

Infine possiamo usare una routine che può fare qualsiasi altra cosa, con il comando `Follow.create(NPC, routine)`. Ad esempio:

```
remove fujico;
Follow.create(fujico, seguifujico1);
"Con un balzo, Fujico sale sopra al muro di cinta e salta
dall'altra parte.";
. . .
[ seguifujico1;
      print "Cercando di inseguirla, ti appresti a scalare il
      muro di cinta.^";
      < >;
];
```

Alla fine quindi notiamo che, dopo tutta la fatica fatta tra classi dedicate e regole di scope, per implementare il sistema di gestione degli inseguimenti l'utilizzo delle funzionalità implementate è risultato decisamente semplice e pulito. Basta una sola istruzione `Follow.create` per stabilire chi viene seguito e come il gioco deve reagire all'inseguimento.

Per concludere, questa è ovviamente solo una delle possibili implementazioni. Ogni autore può e deve valutare se usarla così com'è, oppure se apportare le proprie modifiche. Inoltre, molti dei concetti usati per realizzare tutto questo sono applicabili ad altri problemi. Se avete bisogno di memorizzare delle informazioni, potete usare una classe dedicata e creare oggetti di tale classe, se dovete permettere al giocatore di riferirsi in certi casi ad oggetti lontani da lui, potete usare una regola di scope dedicata. O, più banalmente, se dovete far succedere qualcosa tutte le volte che il giocatore si sposta, potete usare la entry-point `NewRoom`.
Insomma, avete voi il comando.