



Creato da: Natale Scarano, Martino Putignano

**La presentazione della documentazione avverrà nel seguente ordine:**

* presentazione dell’idea generale;
* descrizione del gioco: storia, mappa e istruzioni,
* scelte progettuali e dettagli implementativi.



**PRESENTAZIONE DELL’IDEA GENERALE**

Come definito da traccia, il nostro compito era quello di creare un’avventura testuale in Java, che al suo interno oltre ad avere una storia abbastanza coinvolgente doveva contenere gran parte degli argomenti trattati durante il corso di Metodi Avanzati di Programmazione oltre ad un corretto utilizzo dei concetti di astrazione e delle pratiche di programmazione OO.

Purtroppo data la scarsa conoscenza del tema “avventura testuale” con l’avvento della nuova generazione di videogame abbiamo deciso, prima di iniziare la nostra fase progettuale, di informaci riguardo questa tematica, principalmente trovando alcuni esempi sul web, video di avventure testuali interamente giocate o meglio ancora provando noi stessi a giocarne alcune anche grazie ai vostri consigli.

Da questa fase di analisi ne siamo usciti sicuramente più informati riguardo il tema dell’avventura testuale, scoprendo come quest’ultime venivano create e come apparivano all’utente che doveva giocarle.

Quindi successivamente abbiamo deciso di raccogliere le idee e seguire delle linee guida per la creazione della nostra avventura, ad esempio come la creazione di una ipotetica mappa, una trama e degli oggetti.

Avendo preso nota di alcune tipologie di genere per le avventura testuali, abbiamo deciso di orientarci sul genere “escape”, quindi una trama basata su una mappa “chiusa” dalla quale il protagonista deve fuggire, o meglio trovare una via di fuga dopo aver raccolto determinati oggetti e risolto enigmi.

**DESCRIZIONE DEL GIOCO: STORIA, MAPPA, ISTRUZIONI**

Come detto in precedenza il gioco da noi creato (UNIVERSITY NIGHTMARE) è una semplice e breve avventura testuale nella quale il giocatore si ritroverà inizialmente catapultato il un luogo abbandonato, una vecchia università, dalla quale dovrà fuggire a tutti i costi dopo essersi accorto della sua mano sanguinante che continuerà a togliere i suoi punti vita.

Il giocatore quindi si ritroverà in una stanza iniziale, in questo caso l’atrio dell’università dalla quale inizierà il suo percorso orientandosi grazie ai vari comandi e spostandosi verso le direzioni possibili (nord, sud…).

**La mappa del gioco è costituita da 10 stanze:**

* atrio(stanza iniziale);
* pub del campus;
* laboratorio di informatica;
* ufficio di amministrazione informatica;
* teatro;
* grotta;
* caverna;
* mensa;
* cucina;
* esterno dell’università.

Nel gioco sono distribuiti 4 oggetti:

* Pozione di vita (Pub del campus ): ti dona 5 punti vita, una volta raccolta può essere utilizzata quando lo si ritiene necessario;
* Coltello (Laboratorio di Informatica): viene utilizzato per uccidere l’uomo nella stanza cucina prima di fuggire. Se usato toglie 5 di vita al nemico.
* Libro (Ufficio di amministrazione informatica): il libro è una trappola, se usato ti brucerà e toglierà 5 punti vita.
* Pala(grotta): una volta raccolta deve essere utilizzata per aprire la porta della mensa chiusa a chiave.

Inoltre nel gioco è presente un ipotetico “nemico”:

* Un uomo incappucciato situato in cucina. Il tuo scopo sarà pugnalarlo 2 volte, così facendo morirà e quindi potrai fuggire dall’edificio.

Ha 10 di vita e ti toglierà 5 con un suo attacco.



**Mappa con direzioni**



**COMANDI DA UTILIZZARE PER FINIRE LA PARTITA CON LA STRADA PIU’ BREVE**

1. go sud 8. go sopra
2. prendi coltello 9. go nord
3. go nord 10. usa pala
4. go est 11. go nord
5. go sotto 12. usa coltello
6. prendi pala 13. usa coltello
7. go sopra 14. go nord

**ISTRUZIONI**

(presente anche all’interno della cartella del progetto e può essere visualizzato in console con il comando ‘istruzioni’).

**UNIVERSITY NIGHTMARE**

Scritta da Natale Scarano, Martino Putignano.

Puoi consultare le istruzioni in ogni momento durante il gioco.

----------------------------------------------------

1: Introduzione

2: Interazione con il mondo e comandi

----------------------------------------------------

**Introduzione**

Benvenuti nell' avventura.

Nei prossimi minuti, ti imbarcherai in un viaggio misterioso per sfuggire al

paesaggio in cui ti sei inspiegabilmente svegliato. Lo strumento migliore che hai con te e' il potere dell'esplorazione.

Al fine di fuggire, dovrai esplorare l'ambiente circostante, incontrando una varietà di oggetti lungo

la via.

**Interazione con il mondo e comandi**

I comandi che andrai ad utilizzare non sono CaseSensitive ma sono dotati di sinonimi che puoi utilittare(ES. "esamina" al posto di "guarda").

**GUARDA**

Ci sono molti modi in cui puoi interagire con il mondo che ti circonda. Una delle prime cose che potresti voler fare e'

avere un'idea migliore di cio' che ti circonda. Per fare cio', puoi usare il comando look. Questo restituira' una descrizione

della stanza in cui ti trovi, cosi' come le possibili direzioni che puoi spostare e tutti gli elementi presenti nella stanza in quel momento.

Per eseguire questo comando, e' sufficiente digitare 'guarda' nella console quando viene data l'opportunita'.

**MOVIMENTI**

La prossima cosa che dovrai fare per scappare e' muoverti. Lo spostamento e' abbastanza semplice. Devi essere consapevole di esserlo

puoi spostarti nella direzione desiderata dalla stanza in cui ti trovi attualmente. Queste informazioni sono disponibili

guardando la stanza (GUARDA). Per spostarti, digita 'go' nella riga di comando, seguito da

la direzione desiderata (in termini di nord, sud, est, ovest,sopra,sotto). Puoi anche semplicemente digitare nella direzione che desideri

andare. Se ti sei trasferito correttamente, verra' stampata una descrizione della nuova stanza in cui ti trovi

console.

**PRENDERE**

Gli oggetti sono un aspetto importante. Un modo per interagire con loro e' prenderli. Se ti trovi dentro

una stanza che ha un oggetto, puoi inserire 'prendi' seguito dal nome dell'oggetto, per aggiungere quell'oggetto al tuo

inventario.

**FAR CADERE**

Oltre a prendere oggetti, puoi anche lasciarli cadere. Per eliminare un oggetto dal tuo inventario, digita semplicemente il comando

'lascia', seguito dal nome dell'articolo che desideri eliminare.

**INVENTARIO**

Per tener traccia dei tuoi oggetti raccolti puoi consultare l’inventario deglio oggetti. Quando raccogli un oggetto viene automaticamente inserito nell’inventario, quando lo lasci verra’ tolto.

**SALUTE**

Per tenere traccia dei tuoi punti vita, digita 'salute', cosi' facendo monitorerai la tua salute.

Puoi guadagnare punti vita con la pozione.

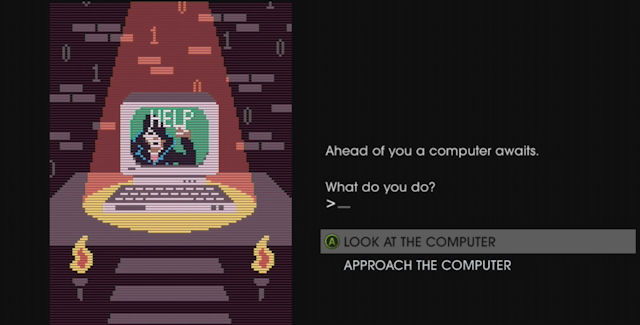
**ATTENZIONE**

Ricorda che la tua mano insanguinata ti fara' perdere 1 punto vita ogni qual volta entrerai in una stanza!!

**STORICO**

Nel menu' iniziale digitando 2, viene stampato lo storico, ovvero le vecchie partite terminate(stampa nome e salute con la quale termina la partita).

Lo storico viene riempito solo se il giocatore termina la partita.



**SCELTE PROGETTUALI E DETTAGLI IMPLEMENTATIVI**

**Comunicazione dell’ utente con i comandi.**

**Classe Parser**

La lettura di un flusso di input in Java avviene attraverso l'oggetto ‘in’ della classeSystem.

System.in appartiene alla classe InputStream (letteralmente flusso di input) la qualemette a disposizione un unico metodo, chiamato read(),il quale legge e restituisce un byte dallo streamdi input (flusso di ingresso

Il modo più comodo e di granlunga più utilizzato dai programmatori Java, anche se non particolarmente efficiente, è quello di far uso del metodo readLine() messo a disposizione dalla classe BufferedReader.

BufferedReader è una classe dedicata alla lettura di buffers (sequenze di caratteri), che il metodo readLine() restituisce sotto forma di stringhe.

La classe BufferedReader, a sua volta, deve essere inizializzata fornendo un reader (lettore) di stream di input. Tale lettore è rappresentato da un'instanza della classe InputStreamReader. L'utilità di un lettore è quella di trasformare la sequenza di byte letti tramite l'oggetto System.in (ogeneralmente tramite un qualsiasi oggetto di tipoInputStream) in una sequenza di caratteri. Secondo gli standard dello schema Unicode adottato Java infatti, un carattere è rappresentato da due byte.

Spesso risulta necessario manipolare dei token di testo, sezionandoli scomponendoli. In Java è presete un la classe StringTokenizer, contenuta nel package java.util.

Per utilizzarla occorre creare in prima istanza l’oggetto StringTokenizer, utilizzando il costruttore dell’omonima classe.

Affinchè si scandisca l’intero testo utilizzeremo un ciclo con condizione di continuazione st.hasMoreTokens(), in pratica invochiamo il metodo hasMoreTokens() sull’oggetto StringTokenizer, questo metodo ritorna true se esistono altri token altrimenti ritorna false.

**Classe CommandWord**

La classe CommandWords controlla la validità dei comandi inseriti in console.

**validCommands[]** è un array costante che contiene tutte le parole di comando valide.

**isCommand** Controlla se una determinata stringa è una CommandWord valida. Restituisce vero se lo è, falso se non lo è.

**Classe Command**

Questa classe contiene informazioni su un comando che è stato emesso dall'utente. Un comando è composto da due stringhe: una CommandWords e una seconda parola (ad esempio, se il comando era "prendi torcia", quindi le due stringhe ovviamente sono "prendi" e "torcia").

Il modo in cui viene utilizzato è: i comandi sono già controllati per essere validi da CommandWords. Se l'utente ha inserito un comando non valido (una parola che non è nota) quindi la CommandWords è 'null'Se il comando aveva solo una parola, la seconda

**Item**

Tutti gli oggetti(Book,Shovel..) ereditano dalla classe astratta Items.

Le classi astratte in Java sono utilizzate per poter dichiarare caratteristiche comuni fra classi di una determinata gerarchia.

Quando estendiamo (o deriviamo) da una classe astratta, la classe derivata deve fornire una implementazione per tutti i metodi astratti dichiarati nella classe genitrice.

**File**

Questa classe gestisce i file txt sia per la sola lettura, per quanto riguarda le istruzioni(istruzioni.txt), sia per quanto riguarda il puunteggio recente dei giocatori(score.txt).

Quest’ultimo file se non è presente viene creato, nel caso contario viene riscritto e aggiornato con gli ultimi punteggi.