Hacker

Autore: Mauro Tridici

Table of Contents

Introduzione

Le nozioni, i concetti e le metodologie acquisite durante la fase di studio del programma d'esame di "Metodi Avanzati di Programmazione" hanno consentito di realizzare il progetto richiesto nel rispetto delle linee guida indicate dall'OOP.
Il progetto, un videogioco ispirato alle avventure testuali "di un tempo", è stato concepito con l'intento di riassumere e rappresentare i principali aspetti e le tecniche tipiche della programmazione ad oggetti: ereditarietà, polimorfismo, incapsulamento, astrazione, ecc…

Inoltre, è stato ritenuto importante introdurre nel progetto anche l'uso delle interfacce grafiche, delle basi di dati, dei thread e dei concetti relativi alla serializzazione degli oggetti al fine di evidenziarne l'estrema utilità e semplicità di gestione.

L'aggiunta di una "veste" grafica, ottenuta tramite l'uso delle librerie fornite da "SWING GUI toolkit", ha consentito di migliorare l'interazione e, più in generale, la user-experience durante la fase di gioco pur conservando, tuttavia, il sapore vintage dell'interazione testuale.

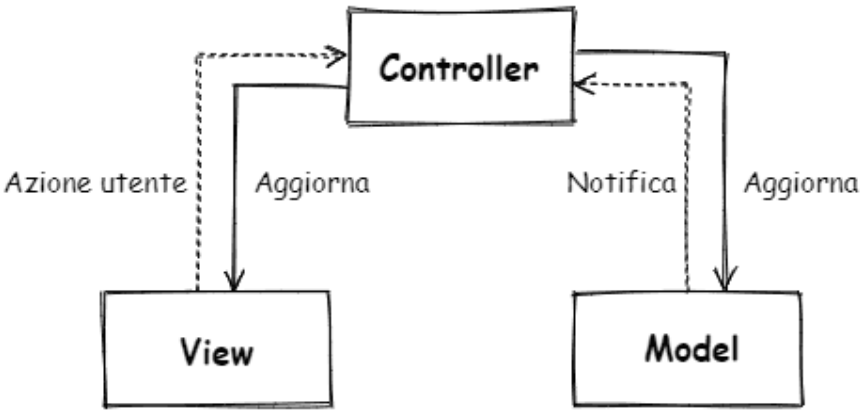
Trama del gioco

Protagonista del gioco è Nicholas Hathaway (Nick), un abile e spregiudicato hacker che si ritrova a scontare una condanna per alcuni reati di pirateria informatica.
"Una svolta per lui arriva nel momento in cui l'agente FBI Chen Dawai decide di avvalersi della sua esperienza per una missione altamente complicata. I servizi segreti si trovano infatti a dover fronteggiare una RAT, ovvero un malware in grado di controllare un sistema da remoto scavalcando le autorizzazioni previste.
Un gruppo di anonimi criminali informatici ha preso il controllo di una centrale nucleare di Hong Kong e del Chicago Mercantile Exange. A patto di un annullamento della pena, Nicholas decide di accettare l'offerta.
Nicholas dovrà dunque indagare sul misterioso hacker (capo del gruppo anonimo), cercando di scoprirne l'identità prima che questi lanci l'ultimo e definitivo attacco alla centrale nucleare.
Per riuscirci, però, si troverà a dover fare i conti con il caso più complesso in cui si sia mai imbattuto.

Architettura del sistema

Il modello MVC

Il progetto implementa il pattern architetturale Model-View-Controller.
MVC prevede un'architettura composta da tre parti diverse: i dati (Model), la visualizzazione dei dati (View) e la gestione degli input (Controller). Questi tre componenti sono interconnessi: il Model viene mostrato tramite la View all'utente, il quale produce gli input con cui il Controller aggiorna il Model.



Mantenerli logicamente separati però ha grandi vantaggi nella gestione del codice, infatti questo pattern favorisce lo sviluppo, il test e la manutenzione di ciascuna parte indipendentemente dall'altra.

I package e le classi

Al fine di separare logicamente le entità definite all'interno del progetto (classi, interfacce, enumerazioni, ecc…) sono stati creati 3 package - denominati "model", "view" e "controller" - destinati a raccogliere ed a raggruppare le stesse entità secondo quanto proposto dal modello architetturale MVC.

Package view

GameGUI: la classe responsabile della gestione dell'interfaccia di gioco.

Package model

Room: la classe contiene gli attributi ed i metodi necessari per definire e gestire le stanze definite nel gioco.

Item: la classe contiene gli attributi ed i metodi necessari per definire e gestire gli oggetti usabili per risolvere i rebus durante la fase di gioco.

Command: la classe contiene gli attributi ed i metodi necessari per definire e gestire i comandi necessari per risolvere i rebus durante la fase di gioco.

Target: la classe contiene gli attributi ed i metodi necessari per definire e gestire gli oggetti passivi (target, appunto) “contro” i quali è possibile esercitare un’azione al fine di risolvere i rebus durante la fase di gioco. Ad esempio, “lancia la dinamite contro la porta” è una frase composta dal comando “lancia”, l’item “dinamite” ed il target “porta”.

Rebus: la classe contiene gli attributi ed i metodi necessari per definire e gestire i rebus che sarà necessario risolvere per arrivare alla soluzione del gioco.

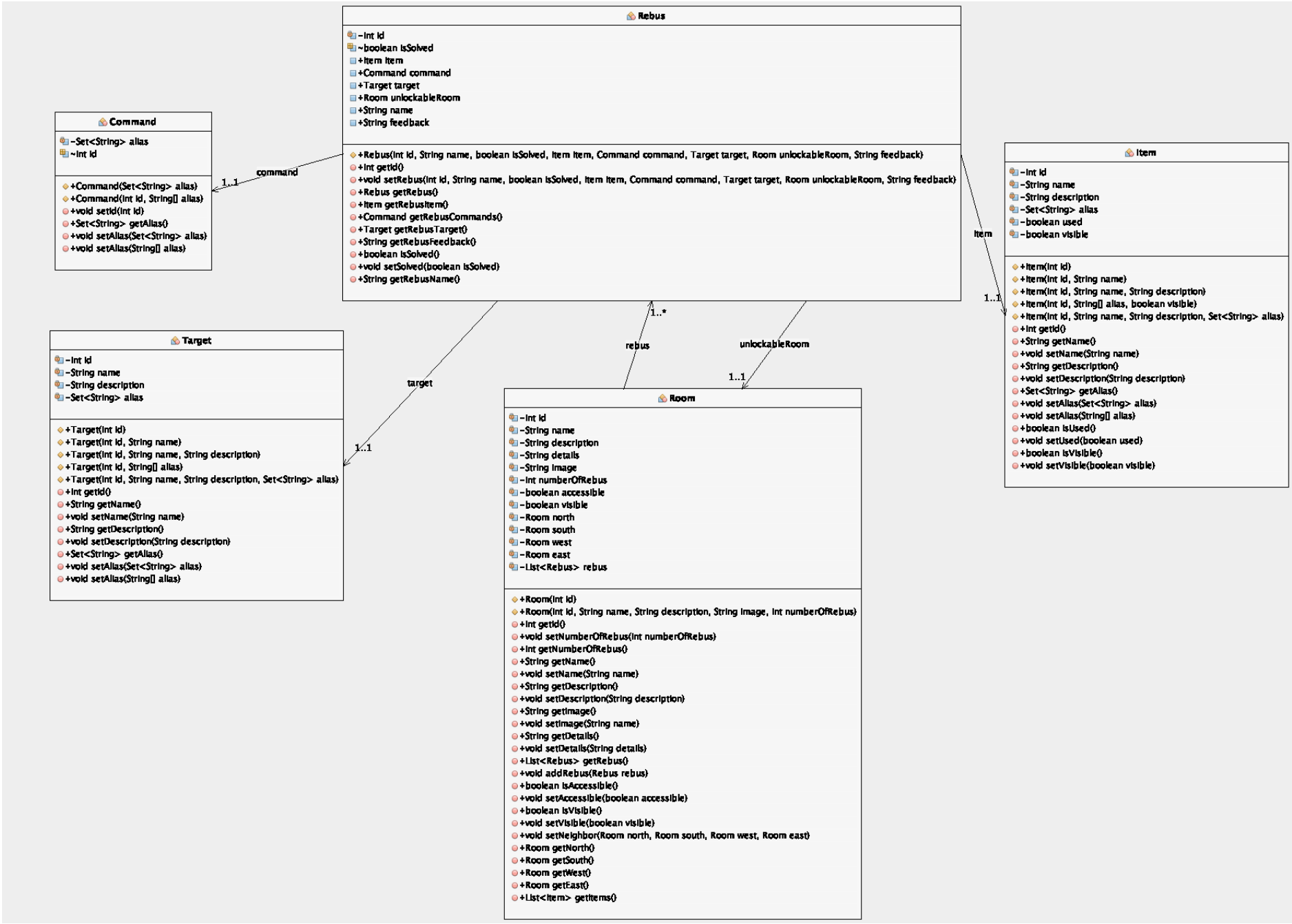
Collector: la classe contiene gli attributi ed i metodi necessari per centralizzare l’accesso ai dati utilizzati dal gioco ed continui aggiornamenti a cui gli stessi sono sottoposti.

Crono: la classe contiene gli attributi ed i metodi necessari per cronometrare il tempo impiegato dal giocatore per completare il gioco.

Package controller

Hacker: contiene il main del gioco ed avvia l’interfaccia grafica
Game: inizializza tutti i dati e gli oggetti necessari per il corretto funzionamento del gioco.
Parser: è la classe responsabile del parsing dei comandi inseriti in input dal giocatore.
Sound: è la classe necessaria per la gestione dello stream audio (colonna sonora) opzionalmente abilitabile da parte del giocatore.
Db: è la classe per accedere al database di tipo H2 e memorizzare/leggere i dati relativa alla classifica dei TOP5 (migliori 5 giocatori)

Diagramma UML delle principali classi



Dettagli implementativi e tecnologie

La fase di gioco prevede, essenzialmente, il superamento di un percorso predefinito risolvendo enigmi di varia natura.
Ad esempio, accade spesso che il player si trovi nella stanza A ed intenda accedere alla stanza B (quest'ultima temporaneamente non accessibile). Per poter procedere dalla stanza A alla stanza B, l'utente dovrà risolvere un "enigma" digitando la frase corretta nella TextBox "action".
Tale frase potrà essere composta da un "comando" + un "target" (es. "apri la porta") o da un "comando" + un "target" + un "item" (es. sfonda la porta con un calcio).

Sulla base di queste riflessioni, è stata creata una classe "Rebus" che usa oggetti di tipo Item, Target, Command e Room.
Ogni oggetto di tipo Room conterrà una lista di oggetti di tipo Rebus. La completa risoluzione dei Rebus previsti per ogni Room consentirà all'utente di accedere alla successiva stanza che risulterà "sbloccata".

Costruttore di "Rebus" ed esempio di instanziazione dell'oggetto:

public Rebus(int id, String name, boolean isSolved, Item item, Command command, Target target, Room unlockableRoom, String feedback)

Rebus rebus00 = new Rebus(0,"rebus00",false,null,jump,gate,boulevard,"Perfetto, potresti essere un campione di salto in alto. Procediamo.");

La risoluzione del rebus00 consentirà all'utente di accedere alla stanza boulevard e ricevere il feedback "Perfetto, potresti essere un campione di salto in alto. Procediamo."

Le classi Db e Sound sono state introdotte per dimostrare le conoscenze acquisite per quanto concerne la gestione delle basi di dati e la gestion dei thread. Inoltre, è stato introdotto anche l'uso della Serializzazione degli oggetti per consentire il salvataggio ed il ricaricamento della partite. Per l'interfaccia grafica è stato usato il framewrok SWING.

A titolo di esempio, è stata inserita una espressione lamda all'interno della classe Hacker.java contenente il main. In tale occasione si è voluto dimostrare come le espressioni lambda (introdotte a partire dalla versione 8 di Java) possano compattare il codice e semplificare la programmazione.

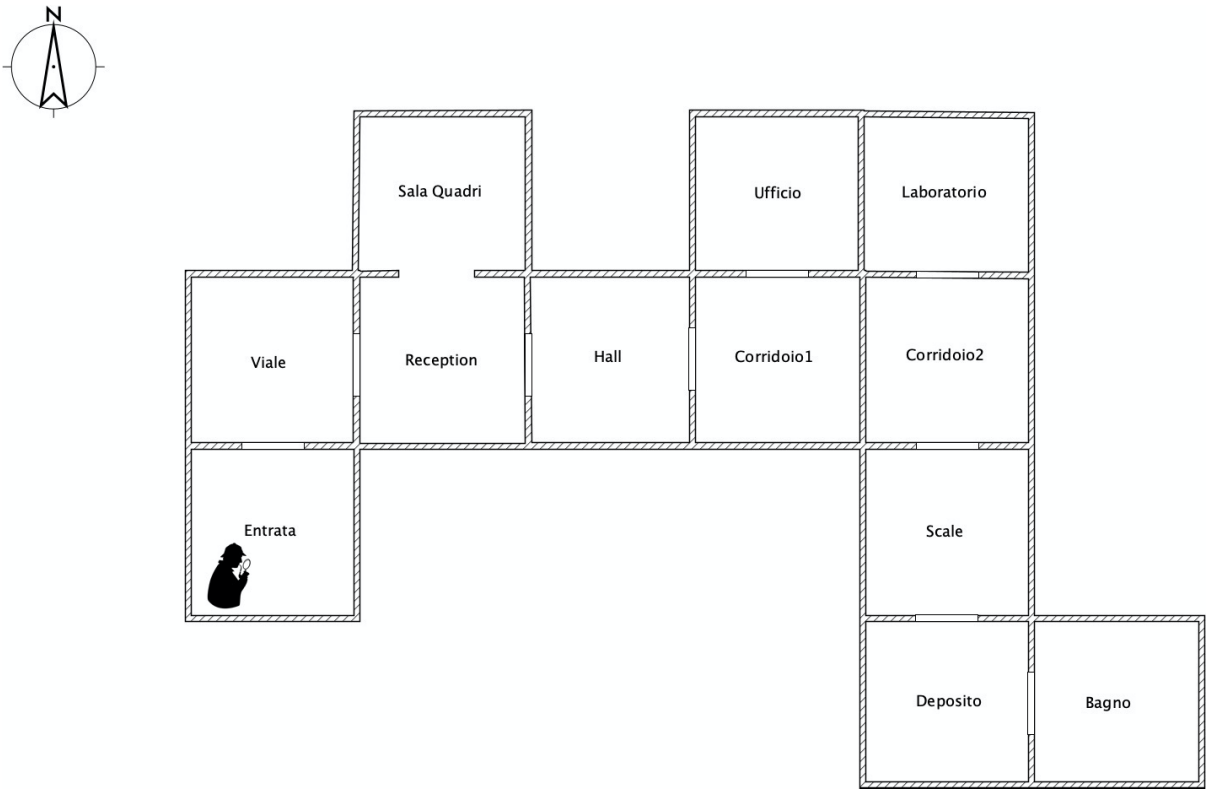
Il codice originale, riportato di seguito e prodotto in automatico dall'IDE utilizzato (NetBeans)

```
   java.awt.EventQueue.invokeLater(new Runnable() {
       @Override
       public void run() {
           GameGui Gui = new GameGui();
           Gui.setLocationRelativeTo(null);
           Gui.setVisible(true);
       }
});
```

È stato compattato utilizzando le lamdba expression:

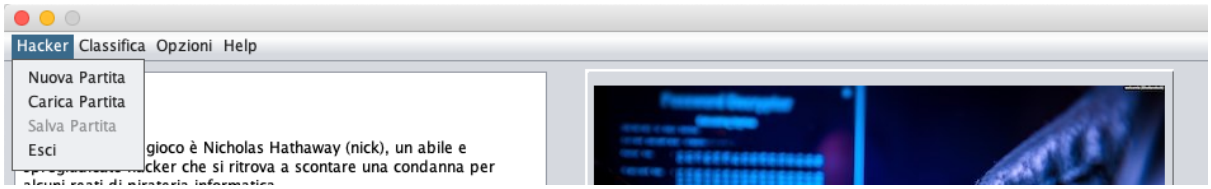
```
java.awt.EventQueue.invokeLater(() -> {
 GameGui Gui = new GameGui();
         Gui.setLocationRelativeTo(null);
         Gui.setVisible(true);
     }
 );
```

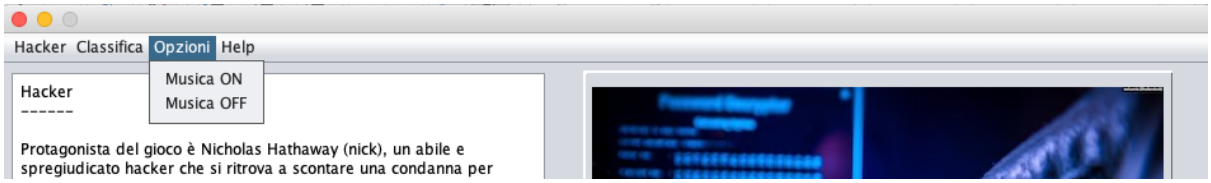
Mappa del gioco



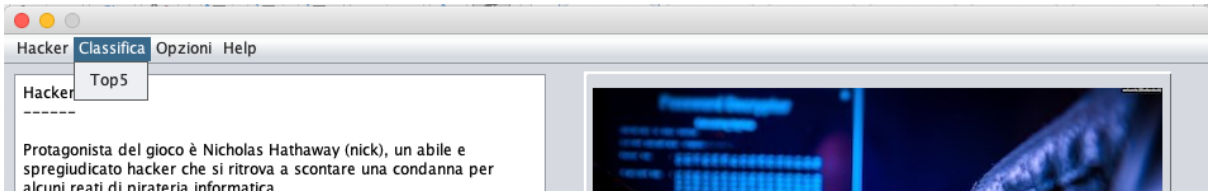
Come giocare

L'applicazione realizzata consente, tramite l'utilizzo di comodi menu grafici, di avviare una nuova partita, salvare la partita corrente o ricaricarne una precedentemente sospesa. Tramite il menu "Opzioni" è possibile attivare o disattivare uno stream audio che, in loop, potrà accompagnare il giocatore per tutta la durata del gioco.

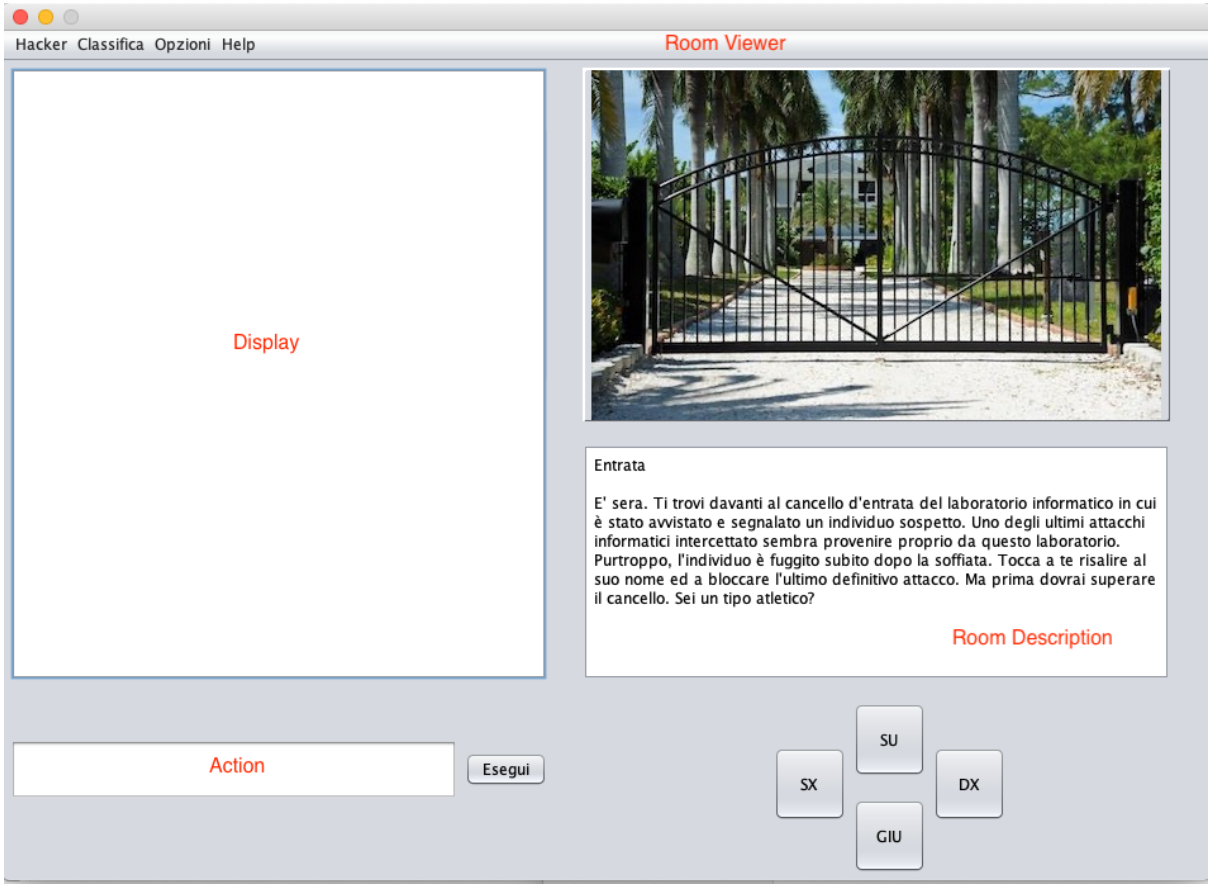


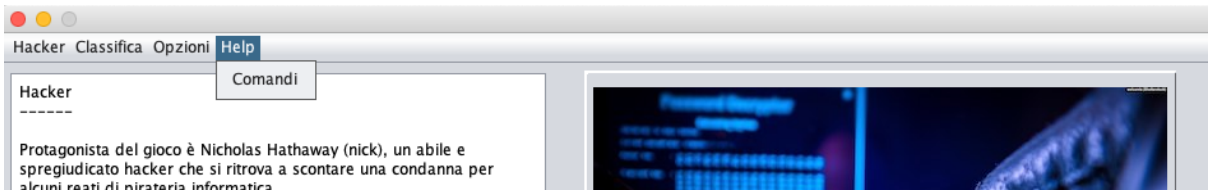


Inoltre, è possibile consultare la classifica dei migliori 5 giocatori (il giocatore più bravo è colui che riesce a completare il gioco nel minor tempo possibile…il tempo è espresso in secondi).



Dopo aver avviato una nuova partita ed aver inserito il nome del giocatore, sarà possibile muoversi usando i tasti direzionali rappresentati graficamente sulla UI.
Tuttavia non sempre è possibile passare da una stanza all'altra senza prima dover risolvere un rebus. Se il giocatore si trova davanti ad un rebus, può dare uno sguardo all'immagine contenuta all'interno del "Viewer" ed agli eventuali oggetti presenti in essa.
Il menu "help" contiene la lista dei comandi accettati dal gioco e fornisce un ulteriore aiuto per andare avanti.





Soluzione del gioco

Per risolvere il gioco è sufficiente rispettare la sequenza di comandi di seguito riportata o, in alternativa, caricare la partita game03.dat e digitare il comando “inserisci codice 666”.

scavalca il cancello + freccia su;
sfonda la porta con il vaso + freccia a destra;
freccia su + attiva interruttori;
freccia giù + freccia a destra;
apri la porta;
freccia su + leggi agenda + inserisci codice 38475;
freccia a destra + freccia giù;
apri il portafoglio;
apri la porta con il badge;
freccia giù + apri la porta;
freccia giù + sfonda la porta con un calcio;
accendi il notebook;
digita codice 666