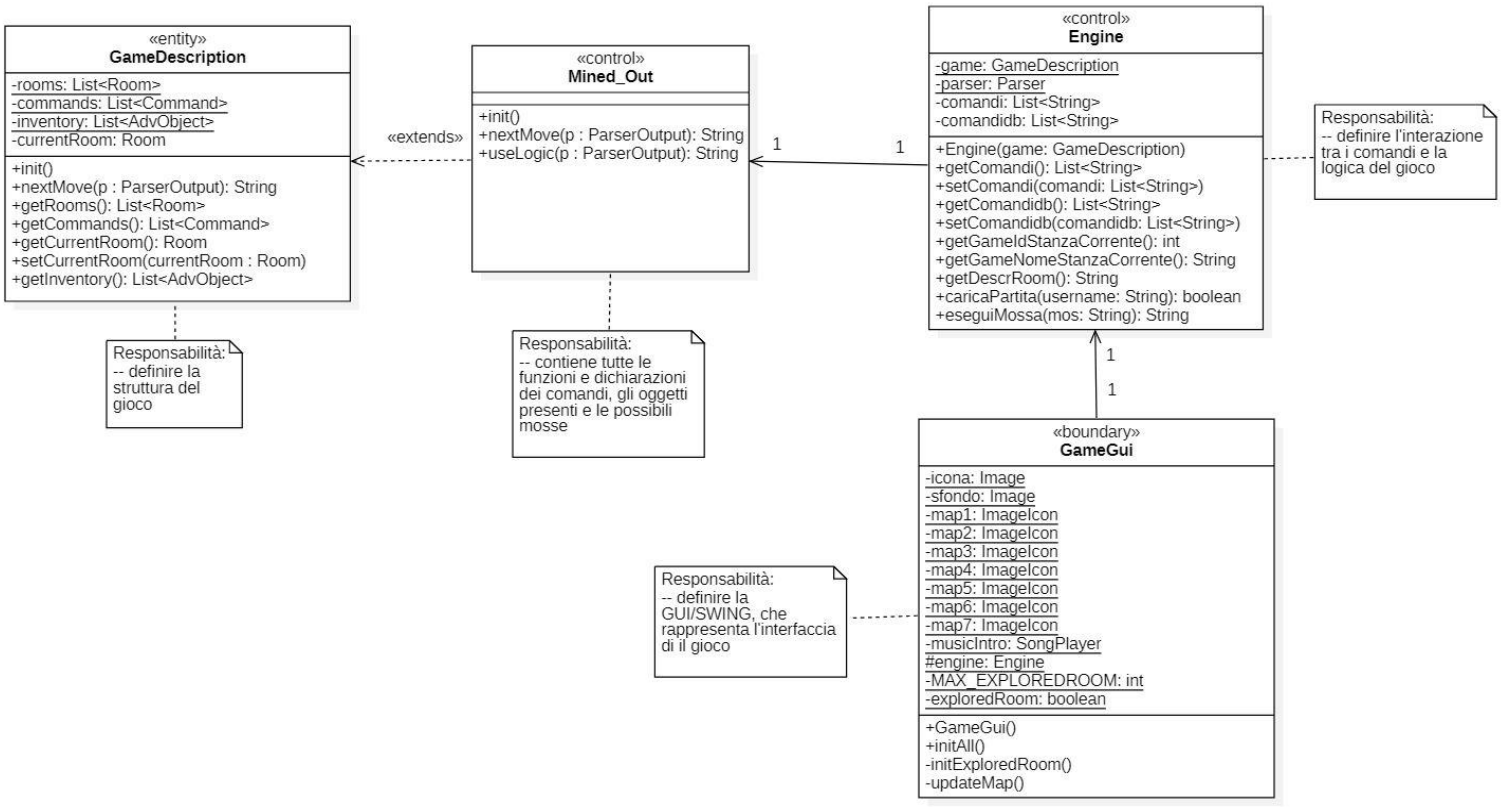
RELAZIONE Mined_Out

Realizzato da: Giuseppe Monitillo, Angelo Sorangelo, Vito Vicenti

Introduzione: idea alla base del gioco ed introduzione sul progetto realizzato

Il gioco è di genere survival. Il protagonista precipita col proprio aereo a causa di un malfunzionamento in un bosco sperduto.
Egli riesce a sopravvive allo schianto, ma sembra essere l'unico sopravvissuto. In quel momento inizia a calare il sole, diventa sempre più buio ed aumenta la forza del vento.
Dopo aver vagato per ore in cerca di un rifugio nel bosco, riesce a ripararsi all'interno di una botola.
L'avventura inizia appena il vento chiude in modo violento la botola costringendo il protagonista a vagare per le vie intrinseche di una miniera abbandonata, cercando una via di uscita.
Il protagonista dovrà di volta in volta riuscire a trovare il modo per passare alla stanza successiva attraverso enigmi e soluzioni, fino ad arrivare all'uscita della miniera.

Architettura del sistema e digramma UML delle classi: dettagli architetturali del gioco realizzato e diagramma delle classi



Eventuali dettagli implementativi e tecnologie utilizzate:

Sono stati utilizzati i Database per dare la possibilità di salvare/caricare una partita.
Quando è aperta l'applicazione, è mostrato un dialog che consente di scegliere se caricare una partita, oppure iniziarne una nuova.
Il riconoscimento degli utenti avviene mediante l'username, difatti nella tabella user del DB sono salvati tutti gli username.

Il salvataggio avviene caricando nella tabella command del DB tutti i comandi che sono stati impartiti dall'utente.
I comandi durante la partita sono salvati temporaneamente in una coda doppia (deque).
Il caricamento avviene eseguendo, e successivamente cancellando dalla tabella, i comandi dell'utente di cui si vuole caricare la partita.

I Thread sono stati utilizzati per la riproduzione di file musicali, avviati all'apertura del gioco, con la possibilità di essere fermata e ripresa tramite l'opportuno comando presente nel menu del gioco.
Selezionando dal menu la sezione tempo si apre un dialog che mostra il tempo di gioco trascorso.
Tale sezione è utile per tracciare i traguardi nel gioco e per avere una visione totale del tempo da quando ha inizio la partita.
La funzione del tempo è stata realizzata utilizzando un thread, il quale si avvia quando viene inizializzato GameGui.

I File sono stati utilizzati per la musica, le immagini delle mappe di gioco, le icone del programma e per le stopwords.

Le espressioni lambda sono state usate per scorrere le liste degli oggetti\stanze e modificare lo stato delle stesse.

Il gioco è basato sulle swing/gui, tramite le quali è possibile visualizzare l'interfaccia di gioco utilizzata durante tutta la partita.
In particolare è mostrato un menu che consente di chiudere la partita, salvando o meno i progressi di gioco, ed un altro menu che consente di fermare o riprendere la riproduzione della musica di sottofondo.

La programmazione Object Oriented è stata la base dell'intero progetto, ed il rispetto dei principi è stato uno degli aspetti più importanti riguardo le scelte di progetto. Sono state utilizzate classi astratte, sottoclassi e classi che implementano interfacce.
Sono state utilizzate strutture dati come liste, code doppie e insiemi per gestire i dati del programma, in particolare per le stanze e per gli oggetti.
Sono stati utilizzati gli Iterator per scorrere le Collection.

Per facilitare il percorso al giocatore è stata creata una mappa, mostrata nell'interfaccia di gioco, che viene aggiornata nel momento in cui l'utente avanza durante il percorso, in modo da avere una visione totale dell'ambiente in cui si svolge il gioco.

Le librerie particolari utilizzate sono inerenti:

-La musica,

```
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;
```

-I Database,

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatemen
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import java.sql.Statement;
```

STRUTTURA DATI DEQUE

Specifica algebrica di una struttura dati utilizzata:

	COSTRUTTORE DI d'		
OSS.	newd	addFirst(d,i)	addLast(d,i)
isnew(d')	true	false	false
removeFirst(d')	error	If !(isnew(d)) then newd else addFirst(removeFirst(d),i)	If !(isnew(d)) then newd else addLast(removeFirst(d),i)
removeLast(d')	error	If !(isnew(d)) then newd else addFirst(removeLast(d),i)	If !(isnew(d)) then newd else addLast(removeLast(d),i)
getFirst(d')	error	If isnew(d) then i else getFirst(d)	If isnew(d) then i else addLast(getFirst(d),i)
getLast(d')	error	If isnew(d) then i else addFirst(getLast(d),i)	If isnew(d) then i else getFirst(d)
deleteDe-que(d')	true	If isnew(d) then newd else addFirst(deleteDeque(d),i)	If isnew(d) then newd else addLast(deleteDeque(d),i)
isEmpty(d')	true	false	false
length(d')	0	length(d)+1	length(d)+1

LEGENDA:

addFirst per inserire un oggetto all'inizio della deque

addLast per inserire un oggetto alla fine della deque

removeFirst per eliminare il primo oggetto della deque

removeLast per eliminare l'ultimo oggetto della deque

getFirst per esaminare il primo oggetto

getLast per esaminare l'ultimo oggetto

deleteDeque per eliminare la deque

isEmpty per la deque vuota

length per la lunghezza della deque

SPECIFICA SINTATTICA

Sorts: boolean, item, Deque, Int;

Operations:

newd() -> Deque

addFirst(Deque,item) -> Deque

addLast(Deque,item) -> Deque

isnew(Deque) -> boolean

removeFirst(Deque) ->Deque

removeLast(Deque) -> Deque

getFirst(Deque) -> item

getLast(Deque) -> item

deleteDeque(Deque) -> Deque

isEmpty(Deque) -> boolean

length (Deque) -> integer

SPECIFICA SEMANTICA

Declare d:Deque, i:item;

Isnew(newd) = true

deleteDeque(newd) = true

removeFirst(addFirst(d,i)) = If !(isnew(d)) then newd else addFirst(removeFirst(d),i)

removeLast(addFirst(d,i)) = If !(isnew(d)) then newd else addFirst(removeLast(d),i)

getFirst(addFirst(d,i)) = If isnew(d) then i else getFirst(d)

getLast(addFirst(d,i)) = If isnew(d) then i else addFirst(getLast(d),i)

deleteDeque(addFirst(d,i)) = If isnew(d) then newd else addFirst(deleteDeque(d),i)

isnew(addFirst(d,i)) = false

removeFirst(addLast(d,i)) = If !(isnew(d)) then newd else addLast(removeFirst(d),i)

removeLast(addLast(d,i)) = If isnew(d) then newd else addLast(removeLast(d),i)

getFirst(addFirst(d,i)) = If isnew(d) then i else addLast(getFirst(d),i)

getLast(addLast(d,i)) = If isnew(d) then i else getFirst(d)

deleteDeque(addLast(d,i)) = If isnew(d) then newd else addLast(deleteDeque(d),i)

isEmpty(newd)= true

isEmpty(addFirst(d,i))= false

isEmpty(addLast(d,i))= false

length(newd)= 0

length(addFirst(d,i))= length(d)+1

length(addLast(d,i))= length(d)+1

SPECIFICA DI RESTRIZIONE

Restrictions:

removeFirst(newd) = error

removeLast(newd) = error

getFirst(newd) = error

getLast(newd) = error

Soluzione del gioco:

guarda
prendi piccone
apri cassetta
prendi chiave
usa chiave
nord
guarda
usa piccone
nord
guarda
ovest
guarda
prendi martello
usa martello
nord
guarda
est
guarda
silenzio
nord
<qualsiasi comando>

Dettagli e accorgimenti:

Il programma nella fase di avvio richiede l'inserimento di un username, il quale non deve essere già presente nel Database (se si sta avviando una nuova partita).
Altrimenti è possibile caricare una partita giocata precedentemente, selezionando l'username dell'utente.

Ogni volta che il giocatore si muove da una stanza all'altra viene comunicato all'entrata nella stanza il percorso effettuato fino a quel momento iniziando dalla prima stanza che è il punto di partenza.

Nella finestra di gioco, tramite il menu in alto, è possibile decidere di fermare la musica o farla ripartire, e mostrare il tempo di gioco.

Nello stesso menu, nell'opportuna sezione, è possibile eseguire il salvataggio dei dati di gioco, in modo da continuare la partita in un secondo momento, oppure abbandonare la partita senza salvare.