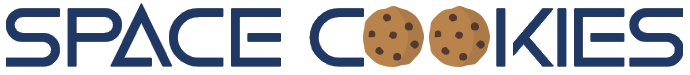
****

**Membri del team**

**- D**onatello Scigliuto, matricola 707052  
**- L**uigi Vulcano, matricola 698958  
**- C**ristiana Sorrenti, matricola 682718

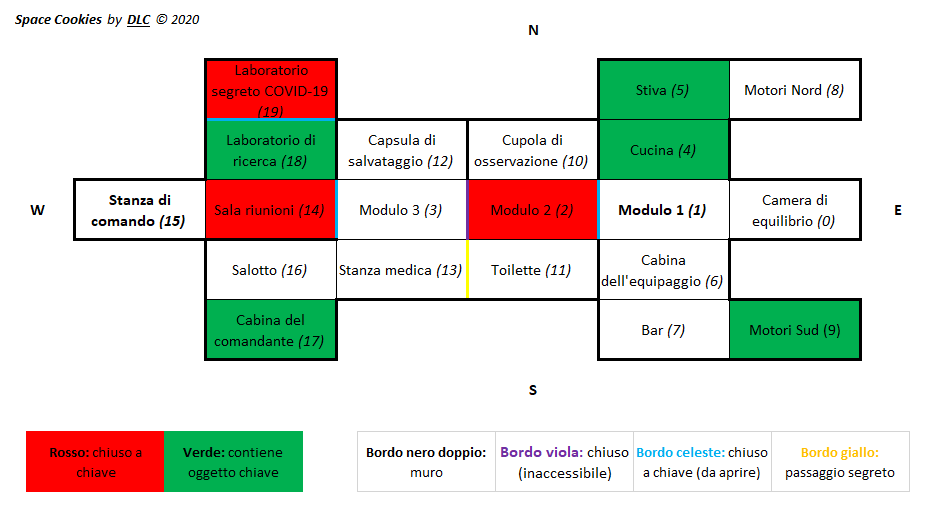
**Descrizione generale del gioco**

Il gioco realizzato è un’avventura testuale che tratta di un personaggio rinchiuso in un’astronave spaziale.  
Il protagonista per raggiungere il suo obiettivo finale dovrà risolvere determinati enigmi presentati all’interno del gioco.

Per orientare meglio il giocatore, la mappa è stata divisa in più step. Superato ogni step non sarà più possibile tornare indietro, in modo tale che il giocatore non continui a girare inutilmente in stanze dove l’interazione con gli oggetti non porta all’avanzamento dell’avventura.

All’avvio del gioco viene data all’utente la possibilità di attivare o meno una musica di sottofondo. La canzone è stata composta appositamente per l’occasione. È stato scelto uno stile vintage per richiamare le avventure dell’epoca.

Subito dopo l’introduzione viene mostrato l’elenco dei comandi principali, per rendere il gioco accessibile anche a chi non conosce questa tipologia di giochi.

**Mappa**

Come si può evincere dalla legenda, ci sono:  
- Stanze che contengono oggetti chiave, necessari per il proseguimento dell’avventura.  
- Stanze chiuse a chiave, da sbloccare tramite la risoluzione di enigmi.  
- Stanze di passaggio, con le quali è possibile interagire tramite dei comandi bonus da scoprire, inseriti per divertire il giocatore.

I numeri in figura rappresentano gli ID associati alle stanze.

**Salvataggio e Caricamento**

Durante il gioco sono presenti alcuni pericoli che portano al Game Over, per cui è stata inserita la possibilità di salvare i progressi ottenuti per poterli caricare successivamente.

Per poter salvare e caricare la partita sono state serializzate tutte le classi inerenti alla partita in corso, attraverso i metodi “save” e “load”.

**Interfacce grafiche**

Nel gioco sono state inserite alcune interfacce grafiche con cui il giocatore può interagire per risolvere determinati enigmi, quali un tastierino numerico per aprire una specifica porta e due diverse immagini che racchiudono degli indizi per il proseguimento dell’avventura.

Questi sono stati inseriti per cercare di dare più immersività al giocatore, in modo da farlo sentire stimolato visivamente.

**PRINCIPALI CLASSI**

**Engine**

La classe Engine contiene il main del programma.  
Essa inizializza il gioco, istanziando tutti gli elementi necessari all’esecuzione del gioco.  
Lo Scanner prende in input tutto ciò che viene digitato nella console e ne passa il contenuto al Parser.  
Se il Parser accetta ciò che è stato digitato, esso verrà passato nella funzione “nextMove”.

**Parser**

Il Parser è in grado di riconoscere il seguente tipo di frasi:  
• <comando>  
• <comando> <articolo/preposizione> <oggetto>  
• <comando> <articolo/preposizione> <oggetto inventario>  
• <comando> <articolo/preposizione> <oggetto> <preposizione> <articolo> <oggetto inventario>  
Dove l’articolo è opzionale. Nel caso venga inserito o meno, la frase verrà riconosciuta lo stesso.

Esempi:  
- “INVENTARIO”  
- “APRI (L’) ARMADIO”  
- “USA (LA) TORCIA”  
- “APRI (LA) PORTA CON (LA) CHIAVE”

La frase digitata nella console viene divisa in token per ogni spazio o apostrofo rilevato.  
Ogni token verrà comparato uno per uno con i possibili comandi/oggetti/articoli.

*1. Comando 🡪 2. Articolo/Preposizione (opzionale) 🡪 3. Oggetto/Oggetto inventario  
🡪 4. Preposizione (frase composta) 🡪 5. Articolo (opzionale) 🡪 6. Oggetto inventario*

**1.** Il contenuto del primo token viene cercato all’interno della lista dei **comandi**.  
Se il comando viene trovato, verifica se c’è un token successivo, altrimenti ritornerà *il comando* al gioco.  
**2.** Rilevato il secondo token, il Parser verifica se si tratta di un **articolo** o di una **preposizione**, cercandoli all’interno di un array di stringhe che li contiene.  
Se è un articolo o una preposizione, passa al token successivo, altrimenti rimarrà sul suddetto token.  
**3.** Il contenuto del token corrente viene cercato sia nella lista degli **oggetti presenti nella stanza attuale** che nella lista degli **oggetti presenti nell’inventario**.  
Se si tratta di un oggetto inventario, ritornerà al gioco *il comando + l’oggetto inventario* trovato.  
Se invece si tratta di un oggetto della stanza, verifica se è presente un token successivo, in caso contrario ritornerà al gioco *il comando + l’oggetto* trovato.  
**4.** Il Parser passa alla ricerca di una **preposizione** tra tutti i token successivi.  
Se nella frase non è presente una preposizione, ritornerà al gioco *il comando + l’oggetto* trovati precedentemente. Altrimenti passa al token successivo per un’eventuale frase più composta.  
**5.** Se è stata trovata la preposizione, il Parser torna alla ricerca di un possibile **articolo**.  
Se lo è, passa al token successivo, altrimenti rimarrà sul suddetto token.  
**6.** Il token successivo viene confrontato con la lista degli **oggetti presenti nell’inventario**. Questo è l’ultimo ad essere controllato.  
Se viene trovato, restituirà al gioco *il comando + l’oggetto + l’oggetto inventario*.  
In caso negativo, la frase non verrà riconosciuta.

Per non rendere il Parser troppo restrittivo, le parole inserite dopo il nome di un oggetto o di un oggetto inventario non vengono prese in considerazione.

**Game**

La classe Game comprende le funzioni di init, nextGame, save e load.

La funzione “init” contiene:  
- L’elenco dei **comandi** con i loro alias.  
- L’elenco delle **stanze**, con il loro ID, nome, descrizione e i loro attributi (se è accessibile, perché potrebbe essere chiusa a chiave, e se è visibile, perché potrebbe avere la luce spenta).  
- La **mappa** del gioco, con i confini di tutte le stanze.  
- L’elenco degli **oggetti**, con il loro ID, nome, descrizione, gli alias, le stanze in cui si trova quell’oggetto e i suoi vari possibili attributi (se è apribile, si può prendere, utilizzare...).

La funzione “nextMove” contiene le azioni che esegue ogni comando, con le loro peculiarità.  
Nel caso di comandi che necessitano di un oggetto, controlla se il suddetto oggetto è presente all’interno della stanza corrente e/o nell’inventario.

Le funzioni “save” e “load” permettono il salvataggio e il caricamento della partita.

**Organizzazione**

A causa dell’emergenza Covid-19 è stato necessario lavorare a distanza. Nonostante ciò, due membri del team sono riusciti a vedersi di persona per un certo periodo, col terzo in collegamento tramite Microsoft Teams.

GitHub si è rivelato uno strumento fondamentale per mantenere il codice aggiornato, senza doversi scambiare continuamente i file che venivano man mano modificati.

Gran parte del codice è stata scritta tramite videochiamata su Microsoft Teams, con tutti e 3 i membri del gruppo collegati.

L’utilizzo dello pseudocodice si è rivelato di grandissima utilità al fine di cercare le migliori soluzioni da adottare e semplificare la creazione del codice finale in Java.

L’organizzazione per il progetto si è svolta tramite gruppo Telegram.