DOCUMENTAZIONE CASO DI STUDIO METODI AVANZATI DI PROGRAMMAZIONE

THE PHILIP MASSE INFORMATIC CRISIS

Creato Da:

Fabio Spiriticchio, matricola 736518

Sergio Mari, matricola 741336

Indice del documento:

1
AVVENTURA GRAFICA

1.1. DESCRIZIONE DEL GIOCO

THE PHILIP MASSE INFORMATIC CRISIS (Abbreviato T.P.M.I.C) prende spunto dal videogioco "Tom Clancy's Splinter Cell Chaos Theory", un videogioco stealth sviluppato per la prima Xbox, Playstation 2, PC e piattaforme secondarie (Gamecube, Nintendo DS, 3DS, Nokia phone)

T.P.M.I.C è un gioco "Text-adventure 3D Interattivo", che segue la storia di Sam Fisher, una splinter cell reclutata da Third Echelon dal programma sperimentale dell'NSA.

Al suo fianco ci sono Irving Lambert, amico di vecchia data di Sam, che gli fornirà istruzioni sul campo, e Anna Grimsdottir che si occupa di raccolta dati.

In questa avventura dovranno ritrovare un ingegnere che ha informazioni vitali sugli algoritmi kernel di Philip Masse, e evitare che cadano nelle mani sbagliate.

1.2. TRAMA DEL GIOCO

E' l'anno 2006, un ingegnere americano di nome Bruce Morghenholt, è stato rapito da una forza separatista indonesiana, chiamata "La voce del popoplo" guidato da Hugo Lacerda, tentando di estorcergli informazioni riguardo gli algoritmi di Masse. Morghenholt aveva lavorato sul "Progetto Watson" per lo sviluppo della nave da guerra "USS Clarence E. Walsh", utilizzando le informazioni sui Kernel di Masse. Questi algoritmi se in mani sbagliate, possono provocare migliaia di morti, mandando in tilt i vari sistemi di comunicazione e di supporto partendo dagli stati uniti, e divulgandosi in tutto il mondo. Devi scoprire dove hanno portato L'ingegnere e verificare se è ancora vivo. In caso contrario lascia li il corpo come prova.

Qualunque mezzo è accettabile durante la missione. Quinta Libertà.

1.3) OBIETTIVI, SISTEMA DI MOVIMENTO, AUDIO, MENU

OBIETTIVO

Completare gli obiettivi primari predisposti dalla missione e arrivare al punto di estrazione.

DIALOGHI

I dialoghi tra i personaggi principali appariranno nel videogioco stesso, mentre i dialoghi degli NPC (Non Playable Character) saranno udibili mediante Audio.

SISTEMA DI MOVIMENTO

Il gioco prosegue in maniera lineare, predisposto con delle sequenze video (frame) e dei pulsanti dove per ogni pulsante premuto, corrisponde alla sequenza video successiva. In alcuni casi è possibile effettuare scelte diverse, che caricheranno l'azione corrispondente e il conseguente dialogo, quando previsto. Il giocatore può interagire (laddove è prevista l'interazione) nelle sequenze aprendo porte, scassinando serrature, disattivando luci, interrogare e scegliere se stordire o uccidere alcuni NPC. Tali interazioni permettono alle sequenze video di proseguire il videogioco, ma il movimento durante esse sarà automatizzato. Una volta completate le sequenze si procede al punto di estrazione, che segna la fine del videogioco.

SISTEMA AUDIO

Il videogioco disponde di un sistema audio per i dialoghi e l'ambiente. Passando da menu è possibile scegliere l'opzione per regolare il volume, o mutarlo del tutto. L’applicazione della modifica sonora richiede che venga prima salvata (che avverrà automaticamente una volta tornati alla schermata principale). Le impostazioni utente verranno salvate a mantenute tra avvii del gioco.

1.3.1) TASTI DI SISTEMA, SALVATAGGIO E CARICAMENTO

Il gioco utilizza dei bottoni predefiniti per eseguire varie operazioni generiche:

-Nuova partita: si occuperà di caricare il gioco dal frame iniziale

-Salva Partita: Può essere usato ad ogni checkpoint del gioco, che si verifica quando bisogna effettuare un'azione. Il sistema prende l'ultimo frame caricato messo in pausa, e salva il nome del frame all'interno di un txt, con la data di sistema.

Inoltre, per la sezione sezione carica partita, verrà creato uno slot con l'immagine del frame attuale, con il nome del file di testo e il numero del frame, per far comprendere meglio quale salvataggio si stia caricando.

NOTA: La classe effettua un controllo anche sui caratteri ammessi dal filesystem attuale per il salvataggio del nome del file. Ad esempio, su Windows non è concesso salvare il nome del file come "con" perché esso è riservato al sistema. Inoltre è stata fornita una lista di caratteri vietati per i seguenti filesystem (Ext (usato su Linux), APFS (macOS), NTFS (Windows))

-Carica partita: Crea una lista all'interno del menu con i salvataggi attualmente esistenti (Generati da Salva Partita), permettendo di caricarne o eliminarne.

-Opzioni: Permette di regolare l'audio del gioco, la scala del volume è in formato Db (Decibel).

-Fine Partita: Torna al menù iniziale.

-Esci: chiude il gioco.

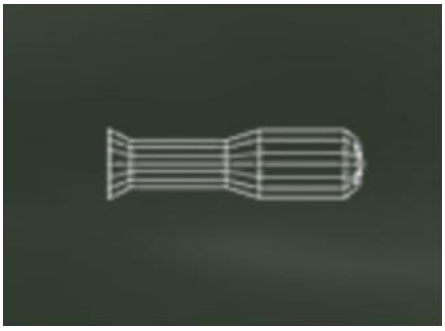
Il menù principale è composto dai tasti Nuova Partita, Carica Partita, Opzioni ed Esci.
La schermata di gioco è composta da Salva Partita, Carica Partita e Fine Partita (oltre ai tasti azione delle scene).

COMANDI:

Ogni scena del gioco comprende una o più possibili azioni, che verranno mostrato nella parte inferiore a sinistra dello schermo.
Eseguire una azione avvierà una nuova scena, che avrà a sua volta altre azioni.

PROTOTIPO DI PISTOLA SC

Proiettili camiciati da 5.7X28mm di caricatori da 20 colpo con silenziatore.



PROIETTILE ELETRICO

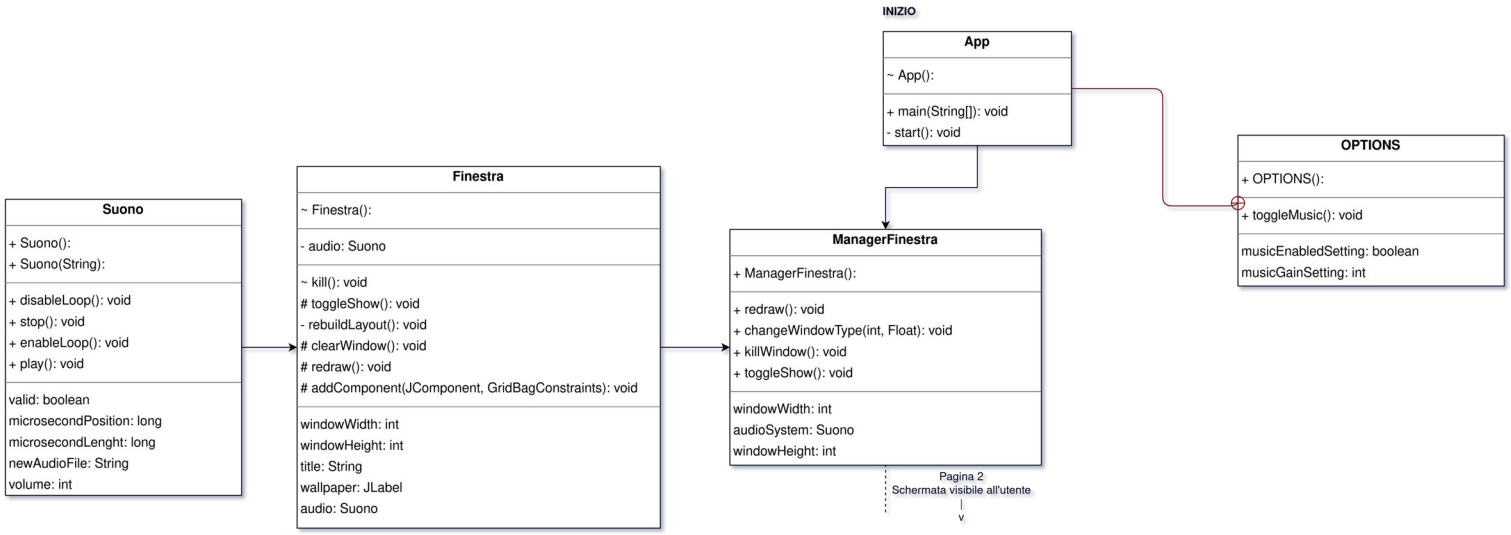
Proiettile adesivo lanciabile da 40mm ad alto voltaggio e basso amperaggio.
La scarica elettrica mette il bersaglio fuori combattimento al contatto.

PROTOTIPO MODULARE DA ASSALTO SC-20K

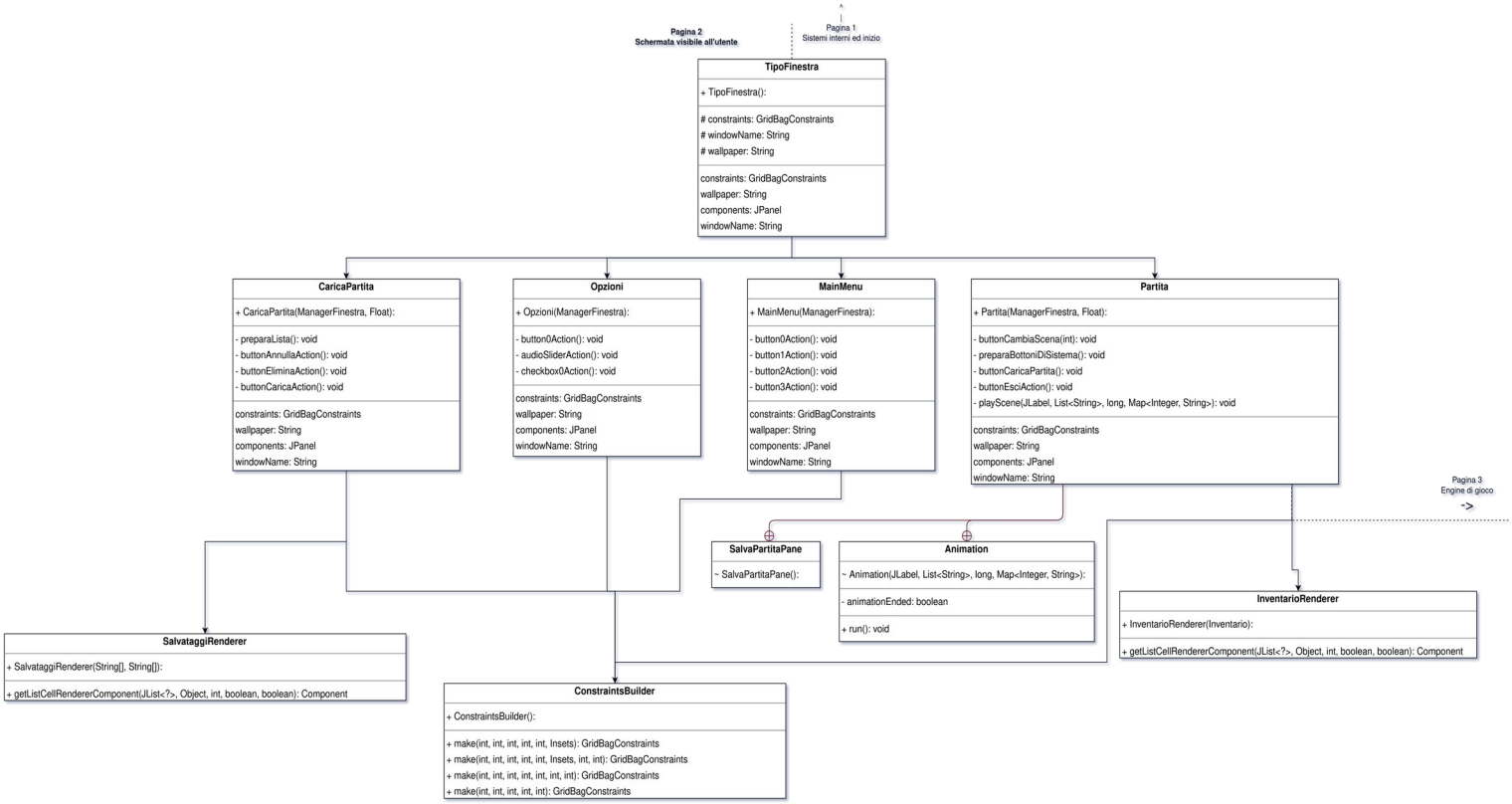
Proiettili camiciati da 5.56X45mm in caricatori da 30 colpi in configurazione Bull-PUP.
Con mirino Reflex 1.5X e silenziatore.

2
ASPETTI TECNICI

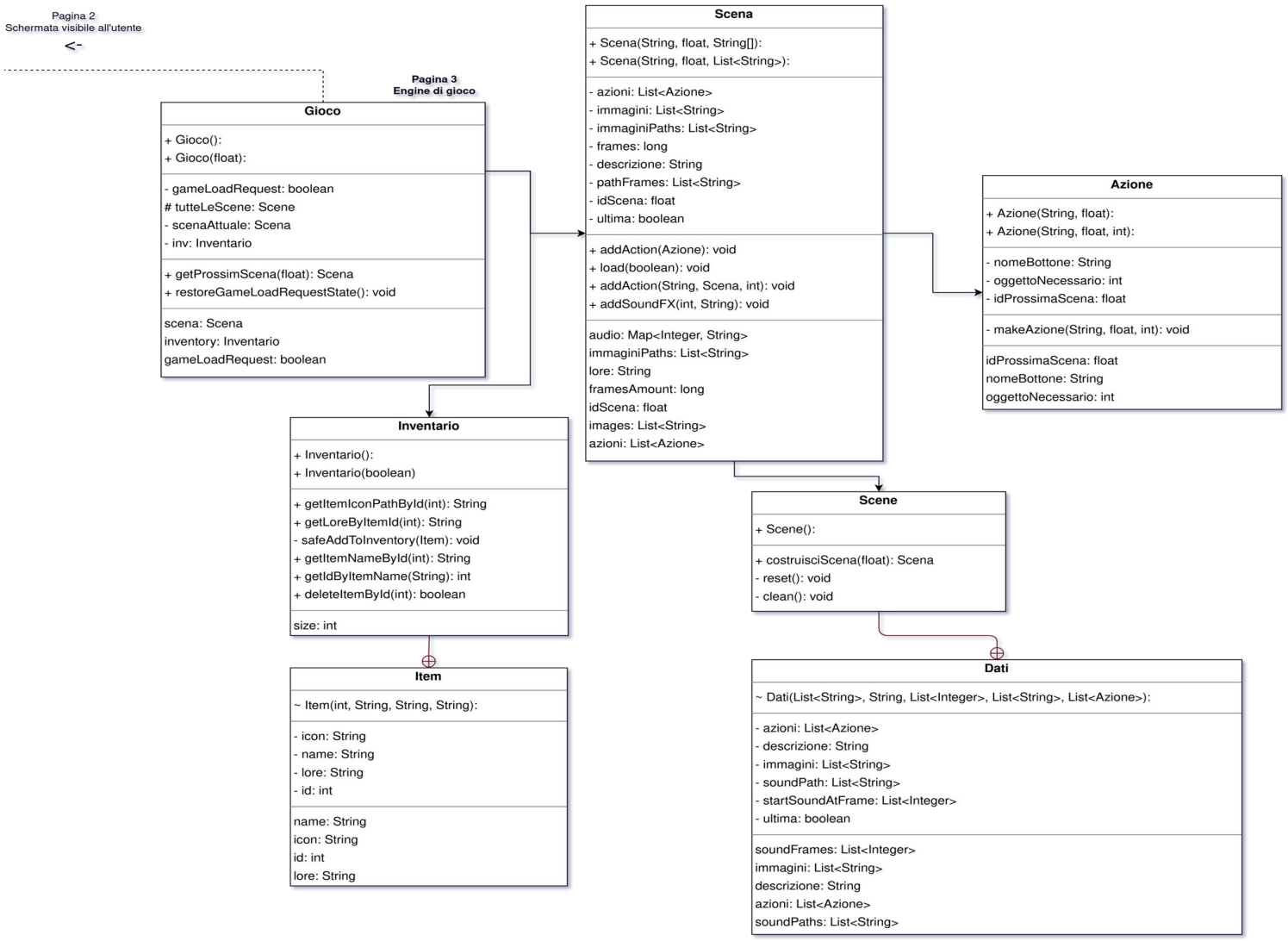
2.1) Diagramma delle classi



Questo è l'inizio del programma, che comprende l'inizializzazione dei i sottosistemi sonori e grafici. Comprende anche le impostazioni utente.
A partire da ManagerFinestra si può scendere verso la componente visibile su schermo dall'utente, di cui la classe ne gestisce la comunicazione.



Basandosi sul tipo generico TipoFinestra, questa è la effettiva componente grafica della applicazione. Comprende tutta la comunicazione input/output con l’utente. La classe Partita è l’unica a comunicare con il motore di gioco, da cui riceve tutti le informazioni riguardanti le scene interattive da mostrare.



La componente motrice contiene esclusivamente tutti i dati riguardo la storia, la grafica, il sonoro e l'interazione del gioco. La classe Scene si occupa di preparare tutti i dati necessari al funzionamento del gioco, dati che verranno consumati dalla precedente classe Partita.

2.2) Architettura del sistema

L'avventura grafica "T.P.M.I.C" è stata scritta interamente in Java.
A livello di programmazione, il gioco è stato sviluppato in *5* package
Elencati di seguito:

scct: Contiene il package Types e due classi java:

- **Finestra**: è la classe contenente lo scheletro della finestra grafica.
- **ManagerFinestra**: Si occupa della gestione della finestra, offrendo metodi per gestire lo stato della finestra come cambiarla, ridisegnarla, chiuderla o prenderne l'audio.

Types: Contiene i package Scenes e utils e 5 classi java:

- **CaricaPartita**: Si occupa di leggere la lista dei salvataggi presenti all'interno del sistema, ricavare il numero del frame salvato, e aggiornare la lista frame per riprendere dal punto salvato.
- **MainMenu**: Offre una interfaccia iniziale con i comandi di scelta base come iniziare una nuova partita, caricarne, modificare le opzioni o uscire.
- **Opzioni**: Predispone gli elementi per la gestione del volume audio tramite il layout precedentemente usato
- **Partita**: Disegna i frame da renderizzare su schermo, con l'inventario laterale, e la descrizione della scena sottostante, gestendo le varie textbox. Ii occupa di verificare che il giocatore non possa salvare la partita prima di aver raggiunto up checkpoint. Quando un checkpoint è raggiunto, mostra anche le azioni possibili

  La sottoclasse **SalvaPartitaPane** si occupa di gestire il salvataggio della partita, verificando tramite regex (regular expressions) che il nome del file sia accettato dal sistema.

  La sottoclasse **Animation** si occupa del renderind delle scene scrivendo sullo schermo i fotogrammi uno ad uno. Inoltre, avvia gli effetti sonori se ce ne sono.

- **TipoFinestra**: Fornisce i costruttori per ottenere nome della finestra, sfondo, componenti e coordinate del layout usato per i componenti. E' la classe padre su cui sono basate le schermate principali dell'applicazione.

Scenes: Contiene il package Base e 1 classe java:

- **Scene**: Inizializza la lista dei frame da renderizzare, con i dati da scrivere su schermo per scena (descrizione, pulsanti).

Base: Contiene 2 classi java:

Azione: Fornisce il pulsante da premere con il nome dell'azione da effettuare e l'oggetto necessario per comperla.

Scena: Inizializza il thread del caricamento delle scene, prendendo il nome dei file da caricare in ordine attraverso un indice ricavato dal nome della cartella.

Utils: Contiene sei classi java:

ConstrainsBuilder: Consente una più facile e meno ridondante costruzione del layout grafico.

Gioco: Contiene la descrizione della partita in corso (l'inventario, lo stato attuale delle scene).

Inventario: Contiene gli oggetti usati nel gioco, offrendo anche una descrizione ed una immagine.

InventarioRenderer: Aiuta nella visualizzazione completa degli oggetti nell'inventario in una lista laterale.

SalvataggiRenderer: Aiuta nella visualizzazione completa dei dati di salvataggio In una lista a quasi schermo intero.

Suono: Si occupa di caricare i file audio e di gestirli. La gestione comprende la regolazione del volume, l'avvio, il fermo e la ripetizione del suono.

2.3) Dettagli implementativi e problemi tecnici

Il progetto è compatibile con l'ultima versione del JRE/JDK 11 o superiore

Può essere compilato ed avviato attraverso il file .jar nella cartella del progetto:

```
mvn compile assembly:single
java -jar ./target/scct*.jar
```

(Lo stesso comando è scritto in comp_and_run.sh)
Oppure compilato ed avviato da un IDE come NetBeans.

Threads

Si sono utilizzati due thread, il renderer e il caricatore di frame.
Per evitare errori, Il thread renderer parte solo DOPO che il caricatore ha caricato il 15% dei frame all'interno di una lista, riducendo il tempo di attesa di inizio di ogni scena, e di poter comunque continuare a caricare i frame rimanenti.

Swing

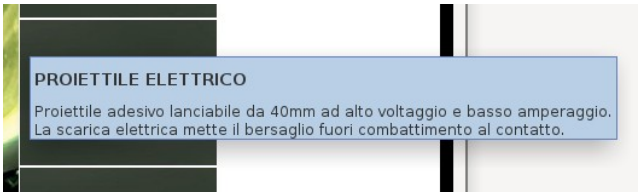
La grafica del progetto è stata scritta utilizzando la libreria Swing di Java. La finestra ed i suoi componenti son gestiti dalla stessa libreria.
La risoluzione minima supportata della finestra è 1280x768.
Il gioco si avvierà in una schermata massimizzata. Se de-massimizzata, la finestra sarà nella risoluzione precedentemente scritta e ridimensionabile.
Risoluzioni minori potrebbero causare la sovrapposizione o il restringimento di alcuni elementi sell'interfaccia.

Componenti come il testo della descrizione delle scene del gioco scalano in base alla risoluzione dello schermo.

Ogni schermata dell'applicazione viene costruita su richiesta verso un ManagerFinestra, che offre un facile spostamento trai vari elementi del programma.

Files

Il programma ha un grande utilizzo di file di vario genere.
I tipi di file utilizzati da programma sono immagini, audio e testo. Di seguito sono i loro utilizzi:
- **Immagini**: Utilizzate per mostrare le animazioni a schermo. Ogni scena è composta da un numero di immagini che vengono sequenzialmente renderizzate a schermo.
- **Audio**: Utilizzati per musiche nei menù, o per effetti sonori e dialoghi durante le scene.
- **Testo**: Utilizzati dal programma per la lettura e scrittura dei salvataggi partita o le impostazioni audio.
File testuali HTML sono anche usati per mostrare un ToolTip descrittivo degli oggetti nell'inventario.



Gioco, scene e azioni

Quando viene iniziata una nuova partita o caricata una precedente, viene preparato (dal costruttore Scene) uno schema che racchiude tutte le scene presenti nel gioco.

Ogni Scena ha un valore identificativo (utile alle azioni per identificare la nuova scena da eseguire) che lo rappresenta nello schema generato precedentemente, oltre ad Azioni che conterranno gli identificatori delle scene successive.

Ogni Azione è composta dall'identificatore della scena da avviare, un nome ed un eventuale oggetto necessario per compiere la azione. Quando un oggetto è necessario, bisogna selezionare l'oggetto corretto dal proprio inventario.

Al termine del video di ogni scena, le azioni inerenti saranno mostrate a schermo. Viene anche assegnato un identificativo alle azioni per permettere la selezione anche nelle scene con scelte. Dalla azione scelta viene ricavato l'identificativo della scena che crea, identificativo che verrà usato per cercare la prossima scena nello schema costruito all'inizio. Quando una scena viene raccolta in questo modo, viene avviata.

Ogni file di salvataggio contiene un identificativo della scena in cui il gioco è stato salvato. Quando il gioco viene caricato, viene raccolto dallo schema la scena corretta ma mostrata solo l'ultimo fotogramma.
In questo modo, le scene caricate mostreranno le azioni immediatamente e sarà nello stesso stato in cui è stato inizialmente salvato.

Mostrare una anteprima per l'inventario ed i salvataggi

Per permettere di mostrare una immagine degli oggetti dell'inventario e una anteprima dei salvataggi, dei renderer personalizzati sono stati creati (Rispettivamente InventarioRenderer e SalvataggiRenderer). Internamente l'inventario e la lista salvataggi sono due liste JList. I renderer permettono di modificare il Comportamento standard del componente scrivendo un JLabel all'interno di ogni cella permettendoci di Impostare una anteprima direttamente nella lista stessa.

Opzioni utente

Attraverso la schermata Opzioni, accedendoci dal MainMenu, è possibile aumentare o ridurre il volume di ogni audio nel gioco. E' anche possibile disattivare il suono completamente.
Queste impostazioni audio vengono salvate in un file di opzioni che viene caricato all'avvio del programma. Se questo file non dovesse esistere, viene generato con delle impostazioni preimpostate.

Suoni

La componente Suono gestisce la musica e gli effetti sonori del gioco. Permette una più facile e diretta gestione dell'audio offrendo metodi diretti come play(), stop(), enableLoop(), setVolume(int) etc.
La classe può tenere attivo una musica o effetto sonoro per volta.
E' presente un oggetto Suono in Finestra, in cui ci si può interfacciare attraverso ManagerFinestra.
L'oggetto Suono è utilizzato in tutto il programma, cambiando il suono da avviare richiamando setNewAudioFile(String).
La classe Suono si occupa anche della gestione dei flussi audio di output.

2.4) Struttura dati utilizzata

Inventario è una struttura dati generata e pre-costruita all'avvio di una nuova partita.
Integra una lista per i vari elementi dell'inventario (Oggetti di tipo Item). Quindi, inventario fa da interfaccia per l'utilizzo di una lista.

Una lista è una sequenza di elementi di un singolo tipo definito (omogenei). E' una struttura dati a dimensione variabile ad indici sequenziali con operazioni Add(Elemento) o Remove(Indice).

La classe Inventario espande sul concetto di lista permettendo di interagire con tutti i dati dagli elementi Item. Ogni elemento Item è composto da un identificatore interno, un nome, un path a file descrittivo ed un path a file immagine.
Perchè è anche possibile eliminare, viene usato questo indice così da mantenere Il funzionamento delle azioni che richiedono un oggetto.

Questa espansione è stata scelta per fornire operazioni dirette verso gli oggetti dell'inventario e nascondere l'implementazione di Item.

Specifica Sintattica (di Lista)

- creaLista() → Lista → crea una lista vuota [In Java come new List<?>()]
- get(int, Lista) → Elemento → Ottiene l'elemento al punto specificato
- add(Elemento, Lista) → Lista → Aggiungi un elemento alla fine
- write(Elemento, int, Lista) → Lista → Modifica un elemento
- remove(int, Lista) → Lista → Rimuove un elemento, eventualmente modificando gli indici degli altri elementi per mantenere la continuità.
- vuota(Lista) → boolean → Ritorna vero o falso se una lista è vuota o meno.

Specifica Sintattica (di Inventario)

- Inventario() → Inventario → Equivale ad un creaLista()
- Inventario(boolean) → Inventario → Se dato vero, equivale a creaLista() e scrive degli elementi predefiniti
- getItemNameById(int)→ String →
- getIdByItemName(String)→ int → Equivalgono al get(int), per poi accedere ad elementi specifici di Item
- getItemIconPathById(int)→ Str. →
- getLoreByItemId(int)→ String →
- getSize() → int → equivale a contare tutti gli elementi della lista, 0 se vuota
- safeAddToInventory(Item) → Inventario → Equivale a add(Elemento), ma tenendo conto che l'elemento Id dell'oggetto rimanga univoco (essendo non legato alla posizione nella lista).
- deleteItemByid(int) → boolean → Equivale a remove(int), ottenendo l'oggetto da eliminare basandosi sull'id di Item.

2.4.1)Osservazioni

Osservazioni (Lista)	creaLista()	add(v)
get(int, Lista)	Errore	if int = 1 then v
add(Elemento, Lista)	Lista[v]	Lista[v,E]
remove(int, Lista)	Errore	If int = 1 then Lista[]
write(Elemento, int, Lista)	Errore	If int = 1 then Lista[E]
vuota(Lista)	true	false

Definizioni:
(String) name, (String) desc, (String) icon, (int) id

Osservazioni (Inventario)	Inventario()	Inventario(true)
getItemNameById(int)	null	If int >= 0 or <= 2 Then name
getIdByItemName(String)	-1	If String € Inventario Then id
getItemIconPathById(int)	null	If int >= 0 and <= 2 Then desc
getSize()	0	3
safeAddToInventory(item)	Inventario[item]	Inventario[Item0, Item1, Item2, item]
deleteItemByid(int)	false	If int >= 0 and <= 2 Then true (Inventario[*] - Item{int})

2.5) Strumenti utilizzati per lo sviluppo

L’intero progetto è stato sviluppato in Linux, utilizzando i seguenti strumenti:
Maven (Project management e compilazione)
Microsoft VS Code (Scrittura codice)
Apache NetBeans (Finalizzazione progetto)
LibreOffice (Scrittura documento)
Ffmpeg e Shotcut (Conversione video e audio)
OBS (Registrazione scene)