AirLine Traffic Simulator

Relazione di progetto
Programmazione ad Oggetti

Severi Andrea
Foschi Andrea
Rodilosso Daniel

Agosto 2021

Indice

Capitolo 1

Analisi

Il gruppo si pone come obiettivo quello di realizzare un videogioco sulla gestione del traffico aereo, ispirandosi al gioco mobile "Flight Control".
L'obiettivo del giocatore sarà quello di far atterrare il maggior numero di aerei che compariranno progressivamente sulla mappa, evitando di farli collidere tra di loro (causando la fine del gioco).
Il giocatore gestirà personalmente la direzione di ogni velivolo, disegnandone il percorso che dovrà seguire ciascun aereo, con il mouse.
La difficoltà di gioco aumenterà con l'aumentare del numero di aerei che saranno fatti atterrare.

1.1 Requisiti

Funzionalità necessarie:

- Disegnare correttamente con il mouse il percorso che l'utente vuol far seguire al velivolo selezionato.
- Implementare un'intelligenza artificiale, che muoverà gli aerei quando non avranno una destinazione scelta dall'utente (l'aereo una volta entrato nella mappa non potrà uscire, ma dovrà continuare a volare all'interno di essa).
- Realizzare una gestione efficiente degli aerei in entrata sulla mappa e il relativo atterraggio quando raggiungeranno la pista di atterraggio.
- Salvataggio degli score dei vari utenti in un file apposito, che verrà reso disponibile ad ogni avvio.
- Gestione della difficoltà crescente durante la partita.

Funzionalità opzionali:

- Creazione e aggiunta di mappe dinamiche (es: implementazione di oggetti che causeranno la distruzione dell'aereo se sorvolati, animazioni dinamiche nella mappa).
- Implementazione suoni di gioco.
- Aerei speciali, con velocità diverse.
- Gestione del vento: questo causerà una maggiore o minore velocità degli aerei durante la partita.

Challenge principali:

- "Fluidità" delle animazioni (tracciamento della rotta, movimento degli aerei, atterraggio degli aerei, collisione tra gli aerei).
- Corretta implementazione del pattern MVC.
- Gestione della difficoltà (crescente durante la partita).

1.2 Analisi e modello del dominio

AirLine Traffic Simulator dovrà essere in grado di gestire lo spawn (Spawn) degli aerei (Plane) e la loro entrata in partita dopo un quantitativo di tempo prefissato.
È presente una **Mappa di gioco** (Seaside) a tema aeroporto.
All'inizio della **Partita** (Game) verranno creati e appariranno in maniera sequenziale gli **Aerei** sulla mappa, che seguiranno un **percorso casuale** (gli Aerei ancora non possiedono una rotta specifica da seguire ma semplicemente si muovono nella Mappa).
In questo caso quando un aereo non possiede una rotta specifica dovrà muovervi verso una direzione indicata dall'**algoritmo di I.A.**
Compito dell'utente è quello di far atterrare gli Aerei sulle **Piste di Atterraggio** (AirStrip) presenti sulla mappa.
Gli Aerei presenti nel gioco potranno essere di più tipologie (PlaneType), aventi velocità diverse.
Un Aereo potrà atterrare quando sarà in prossimità dell'area di atterraggio, resa disponibile dalla Pista stessa.
Sarà quest'ultima (AirStrip) a convalidare la possibilità o meno di fare atterrare il velivolo in caso di contatto con l'area di atterraggio.
Per ogni Aereo che riuscirà ad atterrare correttamente, il Giocatore (User) incrementerà il proprio **Punteggio** con un valore deciso a priori.
Il **Giocatore** potrà eseguire le seguenti operazioni sull'Aereo che vuole muovere:

- selezionare l'Aereo che vuole controllare.

- disegnare il percorso (Path) che poi l'Aereo dovrà correttamente seguire.

È stato pensato di implementare degli indicatori all'interno dell'area di gioco per segnalare gli aerei in arrivo da fuori dalla Mappa.
Questi segnali scompariranno non appena l'aereo entrerà in partita.

Il giocatore perde se:

- fa collidere gli aerei tra di loro in volo
- urta il bordo con un aereo presente in partita

In entrambi in casi, viene generata un'esplosione a video.
Nel tentativo di evitare ciò, allo scorrere del **Tempo di gioco**, altri aerei che dovranno essere fatti atterrare, compariranno sulla Mappa, con maggiore frequenza in base al punteggio.
É possibile mettere anche in **pausa** la partita tramite pulsante dall'interfaccia di gioco.
Alla fine della Partita, il Giocatore avrà ottenuto un punteggio finale, che sarà salvato su **file**. Se sarà sufficiente per entrare in **Classifica**, verrà inserito in quest'ultima.
Dal **Menù Principale**, oltre che iniziare una nuova partita, si potrà visualizzare tale classifica.

Capitolo 2

Design

2.1 Architettura

L'architettura di Airline Traffic Simulator segue l'architettura M.V.C.
Abbiamo deciso di utilizzare il design classico di questo pattern.
Utilizzando la libreria di JavaFX abbiamo un unione molto dipendente tra la parte di View e quella di Controller. Questo perché la View stessa viene disegnata e implementata tramite un file .fxml, il cui scheletro deve essere presente all'interno del Controller. Ciò genera da una parte un ottimizzazione per la realizzazione dei due componenti, dall'altra però va inevitabilmente a creare un forte legame di dipendenza. In fatti nel caso in cui successivamente si volesse cambiare libreria grafica sostituendo JavaFx, la parte di Controller, andrebbe interamente riscritta, rendendo meno semplice la sostituzione in blocco della View stessa.
AltSim implementa Seaside ed è il Controller centrale del sistema, ad esso sono infatti collegati i vari Controller degli oggetti chiave dell'applicazione (Plane, Map, Airstrip ***). All'interno di Seaside l'utente sarà in grado di disegnare e controllare la rotta dei vari Plane che entreranno in partita. Una particolare attenzione merita il GameEngine. Esso si occuperà di monitorare i vari oggetti durante la partita (come ad esempio per il caso di Plane, verificare se collide con un altro Plane, o essendo in prossimità della AirStrip far atterrare il Plane stesso) e intraprendere le azioni di aggiornamento e notifica adeguati. Abbiamo riscontrato che un vantaggio di utilizzare questa architettura è che è possibile inserire altre Entità o modifiche all'interno del nostro gioco senza impattare la complessità di progettazione.

2.2 Design dettagliato

2.2.1 Severi Andrea

Durante le fasi di analisi, è emersa l'esigenza di mantenere i dati relativi alle partite degli utenti in un file apposito e di estrapolarne una classifica contenente i migliori giocatori. Per cui, uno dei miei principali obiettivi era l'apportare un contributo per la creazione dell'utente e del salvataggio del suo punteggio.

User

Occupandomi personalmente dell'interfaccia grafica dell'applicazione, avevo subito in mente ogni singolo passo per la realizzazione dello `User`. Le sue caratteristiche sono l'univocità del nome utente da scegliere nel menù principale e il punteggio, attribuito successivamente in partita. Nonostante la scelta di soltanto due parametri, a mio vedere sufficienti per lo stato attuale dell'applicazione, ho deciso di utilizzare il pattern **Builder** per la creazione dell'utente. La motivazione è la predilezione per un pattern che fornisce chiarezza nella lettura del codice, rispetto a un costruttore, e per fornire una base solida di partenza nel caso di futura aggiunta di nuove caratteristiche da attribuire all'utente stesso.

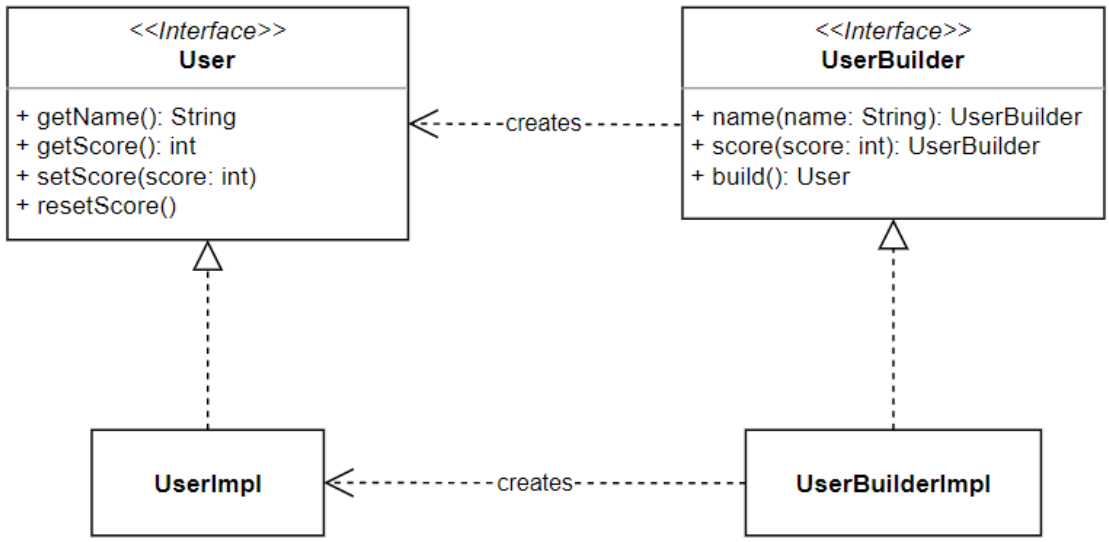


Figura 2.1: Creazione di `User`

UserRecords

Al fine di mantenere in memoria la creazione degli utenti e dell'attribuzione loro di un punteggio, è stato necessario creare una classe che potesse gestire la realizzazione di una *hidden directory* all' avvio dell'applicazione con al suo interno il file dei punteggi. Inizialmente, l'approccio scelto è stato la realizzazione della classe `RecordsValidation`, la quale controllava sia dell'esistenza di directory e file sia della creazione di essi. Tale metodo non era di mio gradimento, per cui ho deciso di separare le due parti di logica utilizzando il pattern **Adapter**, in modo tale da poter riutilizzare i metodi necessari anche per `UserRecordsImpl`, evitando ripetizioni di codice e violazioni del principio DRY. Per quanto riguarda l'interfaccia `UserRecords` ho optato per il pattern **Strategy**, in quanto contiene algoritmi utilizzabili per qualsiasi tipo di file si voglia usare. In prima battuta, la scelta è ricaduta su due estensioni: *.csv* e *.json*. La decisione di usare *.json* è stata dettata dal fatto che è più facilmente manipolabile, grazie alla libreria Gson, rispetto alla controparte, di cui è possibile trovare librerie di terze parti, ma che sono meno convincenti, data la scarsa accuratezza nell'attribuzione ai nomi dei metodi e il minor sviluppo da parte del team di creazione[1]. Per cui `UserRecordsImpl` contiene l'implementazione json degli algoritmi di inserimento, attribuzione del punteggio e controllo riguardo la presenza dell'utente nel file.

[1]In particolare si fa riferimento alla libreria Tablesaw

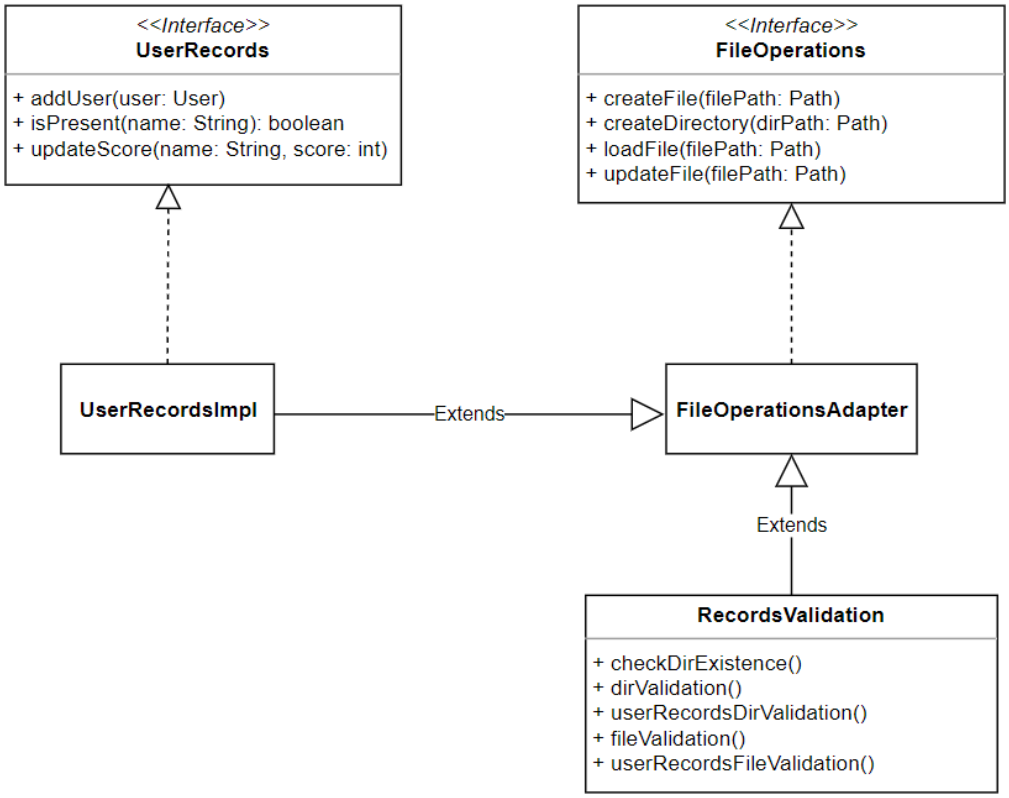


Figura 2.2: Gestione del file

Leaderboard

Per quanto riguarda la realizzazione della classifica ho pensato a un semplice algoritmo di comparazione fra gli utenti, basato ovviamente sul punteggio, volto all'utilizzo degli *Stream*, introduzione nella versione 8 di Java, nonchè uno dei pilastri del corso.

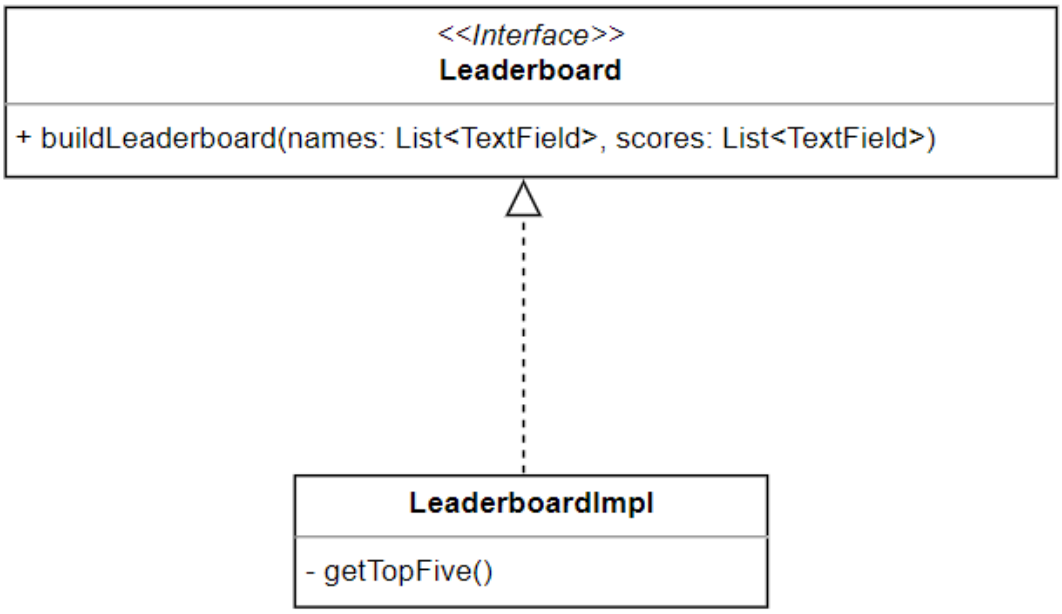


Figura 2.3: Gestione della classifica

NameQuality

La classe `NameQuality` consente di controllare il nome inserito dall'utente nel menù principale e di valutarne la qualità, facendo affidamento all'enum `NameResult`. Al fine di ottenere nomi semplici e di lunghezza finita, ho deciso di implementare un pattern regex, il quale restringe la scelta del nome a un lunghezza massima di 12 lettere e l'utilizzo di soli numeri e lettere. In tal modo, l'input viene verificato tramite espressione regolare e viene ottenuto uno dei seguenti valori:

- "`CORRECT`": il nome segue i canoni imposti
- "`EMPTY`": il nome è vuoto o contiene soli spazi
- "`TOO_LONG`": il nome è più lungo di 12 caratteri
- "`WRONG`": il nome contiene caratteri non validi
- "`TAKEN`": il nome è già in uso

In base al valore ottenuto, successivamente si agisce in maniera diversa in interfaccia grafica. Tale scelta mi ha permesso di utilizzare spesso l'`Enum` nella View per inviare informazioni all'utente riguardo il nome selezionato.

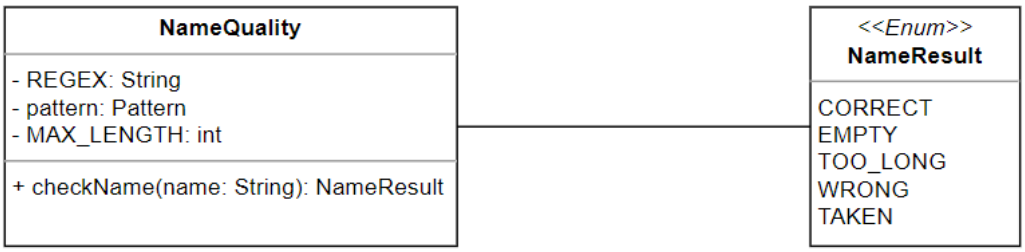


Figura 2.4: Controllo sul nome

2.2.2 Foschi Andrea

Piste di atterraggio

Mi sono personalmente occupato di una delle due componenti fondamentali del gioco: le piste di atterraggio. Tale componente si suddivide in due macrocategorie: quelle che gestiscono l'atterraggio degli aerei standard, e quelle che gestiscono l'atterraggio degli elicotteri (in realtà per quest'ultima categoria vi è nel codice una sola predisposizione a tale tipo di velivolo per implementazioni future, a causa della mancata implementazione degli elicotteri).

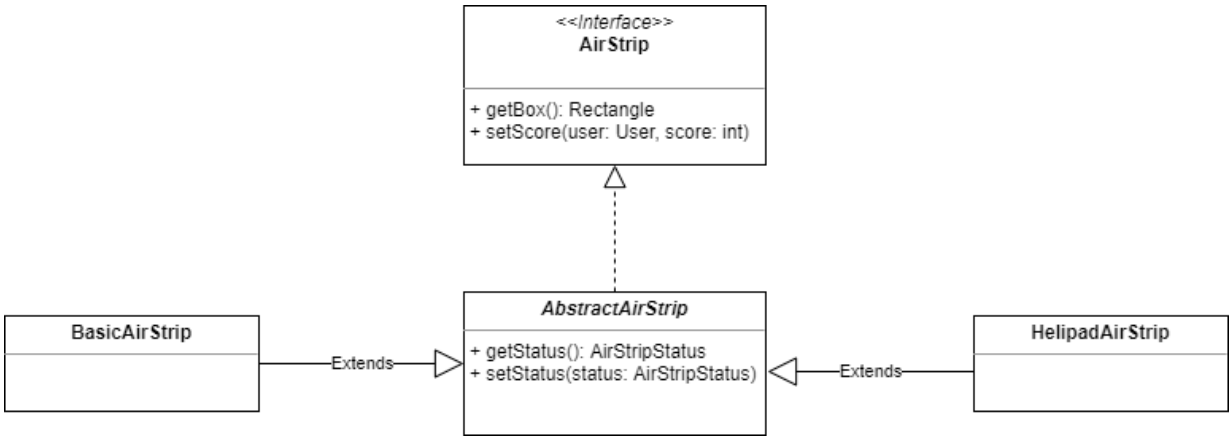
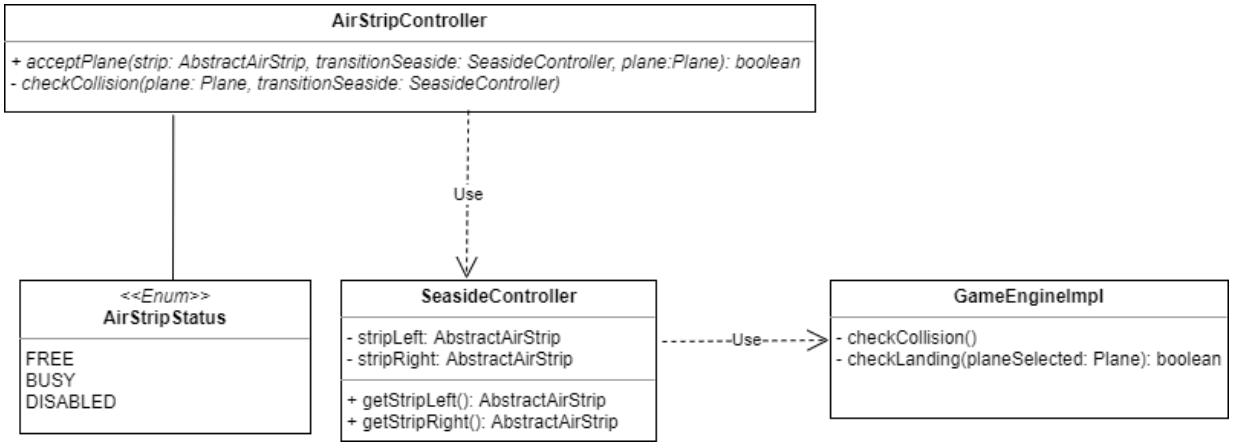


Figura 2.5: Pista di atterraggio

Atterraggio degli aerei e verifica delle collisioni

Per fare in modo che la pista di atterraggio prenda in ingresso un aereo da far atterrare è stato necessario definire un controller del sistema. In questo modo la pista di atterraggio, tramite il metodo intersects() fornito da JavaFX, può accertare la collisione avvenuta e notificare l'evento alle altre componenti dell'applicazione. Si denota l'utilizzo dell'Enum AirStripStatus, che può assumere i seguenti valori:

- "`FREE`": la pista è in grado di ricevere un aereo da far atterrare;
- "`BUSY`": la pista sta facendo atterrare un aereo e non è in grado di riceverne altri;
- "`DISABLED`": la pista è momentaneamente non disponibile.



2.2.3 Rodilosso Daniel

Capitolo 3

Sviluppo

3.1 Testing automatizzato

3.1.1 Severi Andrea

3.1.2 Foschi Andrea

3.1.3 Rodilosso Daniel

3.2 Metodologia di lavoro

3.2.1 Severi Andrea

3.2.2 Foschi Andrea

3.2.3 Rodilosso Daniel

3.3 Note di sviluppo

3.3.1 Severi Andrea

3.3.2 Foschi Andrea

3.3.3 Rodilosso Daniel

Capitolo 4

Commenti finali

4.1 Autovalutazione e lavori futuri

4.1.1 Severi Andrea

Questo progetto mi ha sicuramente aiutato a crescere come sviluppatore e mi ha fatto appassionare maggiormente a un linguaggio che inizialmente reputavo difficile e temibile, tanto dall'aver acquistato volumi quali "*Effective Java*" e "*Design Patterns*". Sono consapevole del fatto che ho ancora molto da migliorare, ma cerco sempre di documentarmi al meglio per trovare tecniche e approcci migliori alla scrittura del codice. Ho sempre cercato di rispettare regole (lingua inglese, stile) e convenzioni, affinchè il codice sia leggibile e comprensibile da tutti. Per il futuro voglio sicuramente concentrarmi su uno studio più approfondito dei Design Patterns e su una scrittura più professionale del codice, magari creando nuovi progetti da zero o andando a rimodellare quello appena creato. Inoltre conoscere sempre di più la programmazione ad oggetti che tanto mi ha appassionato, con lo studio possibilmente anche di altri linguaggi.

4.1.2 Foschi Andrea

4.1.3 Rodilosso Daniel

4.2 Difficoltà incontrate e commenti per i docenti

4.2.1 Severi Andrea

Come gruppo, nonostante la comunicazione costante, è sicuramente mancata un po' sintonia, conoscendosi poco e sarebbe servita una migliore organizzazione, in modo tale da non riscrivere parti già fatte e concentrarsi sulle singole parti. Forse poteva servire per evitare di fallire la deadline precedente. Il corso mi ha aiutato a ragionare molto di più sul problema da risolvere, piuttosto che buttarsi subito sulla scrittura di codice. Nonostante sia un corso difficile, ho apportato il massimo impegno e penso ciò verrà ripagato. Una cosa che non mi è piaciuta a pieno è stata la spiegazione su JavaFX, poichè non ero bene a conoscenza di ciò che realmente offre e che pensavo si dovesse implementare da zero. Nonostante questo, corso fatto molto bene.

4.2.2 Foschi Andrea

4.2.3 Rodilosso Daniel

Appendice A

Guida utente

Home Page

All'avvio dell'applicazione, sono presenti 4 pulsanti: `START`, `LEADERBOARD`, `CREDITS`, `EXIT`, per avviare una nuova partita, visualizzare la classifica dei migliori giocatori, visionare i riconoscimenti per ognuno dei membri del gruppo e uscire dall'applicazione.

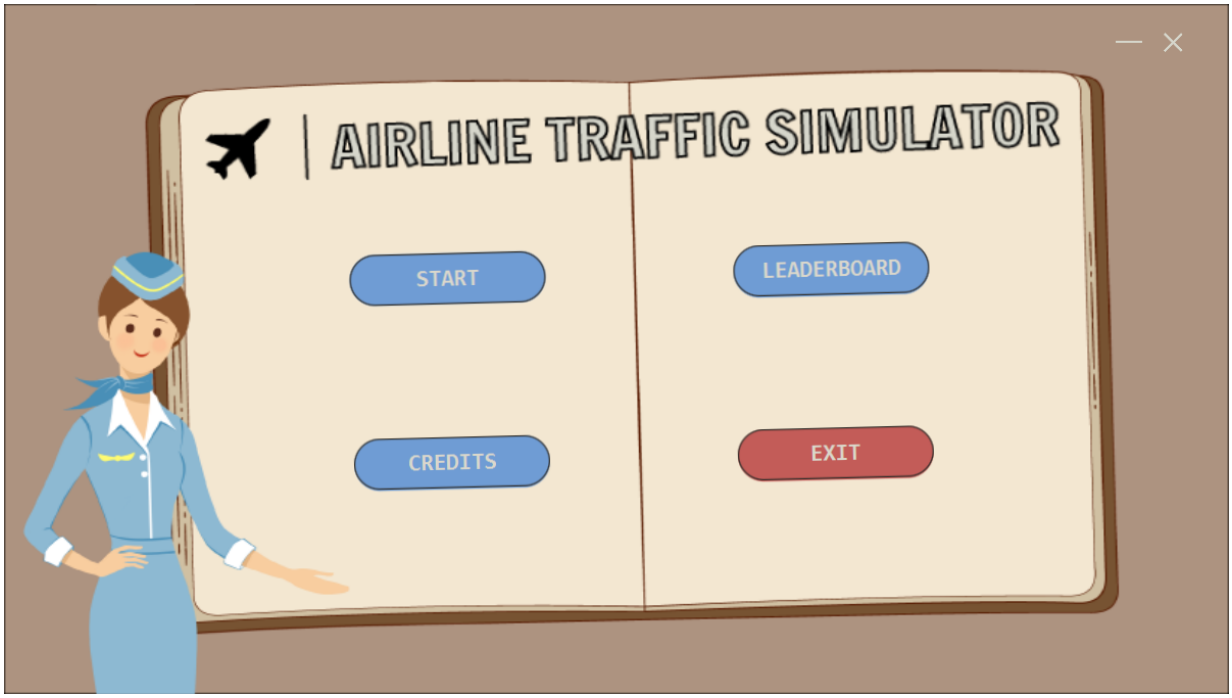


Figura A.1: Home Page

MapChoice

Tramite questa schermata è possibile scegliere una delle mappe disponibili, al momento solo una delle quattro. Inoltre è obbligatorio scegliere un nome, come suggerisce l'info per iniziare una nuova partita.

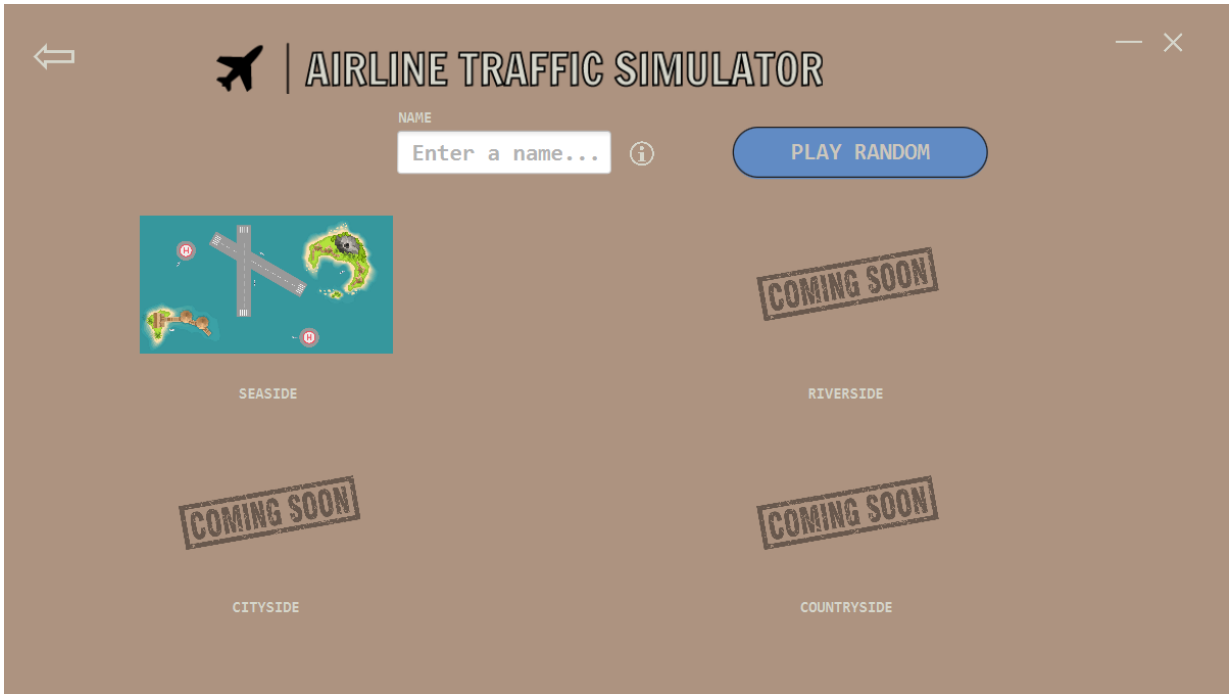


Figura A.2: Scelta della mappa

Classifica

In questa schermata è possibile vedere la top 5 dei migliori giocatori di AirLine Traffic Simulator con relativo nome e punteggio.

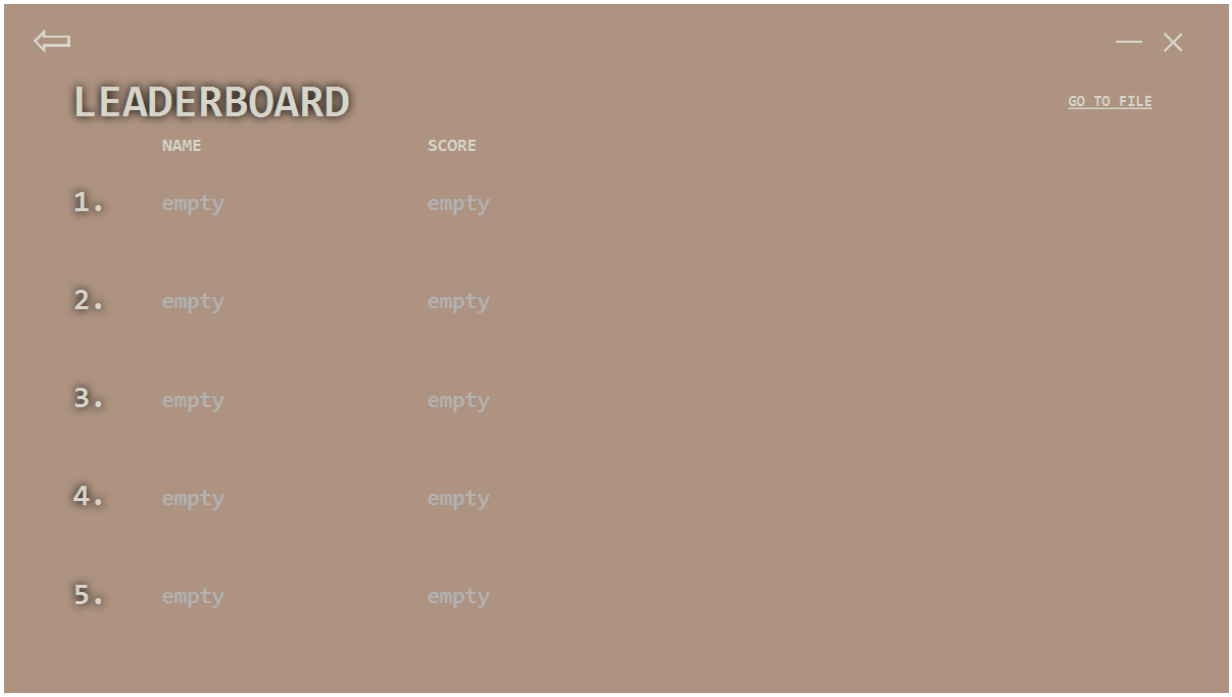


Figura A.3: Classifica

Credits

Schermata per una veloce visione del lavoro svolto dai singoli partecipanti al gruppo.

Figura A.4: Riconoscimenti

Partita

All'avvio della partita, la mappa Seaside si presenterà così con l'opzione di poter mettere in pausa il gioco e uscire o riprendere successivamente.

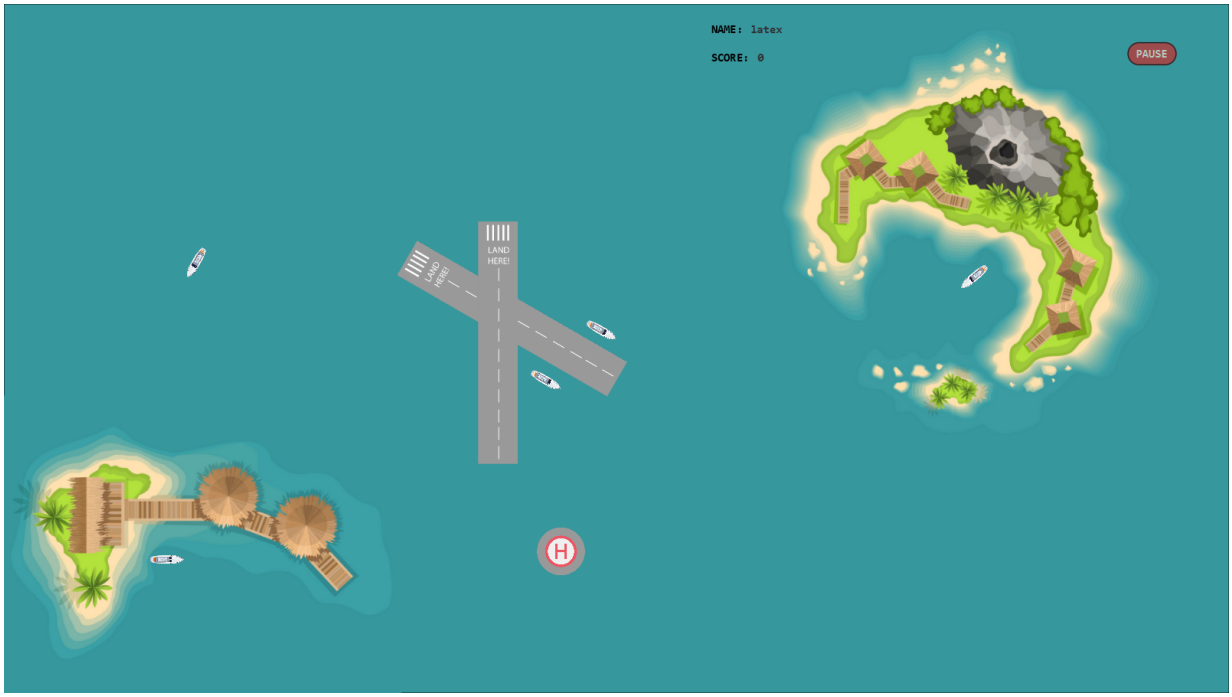


Figura A.5: Mappa di gioco