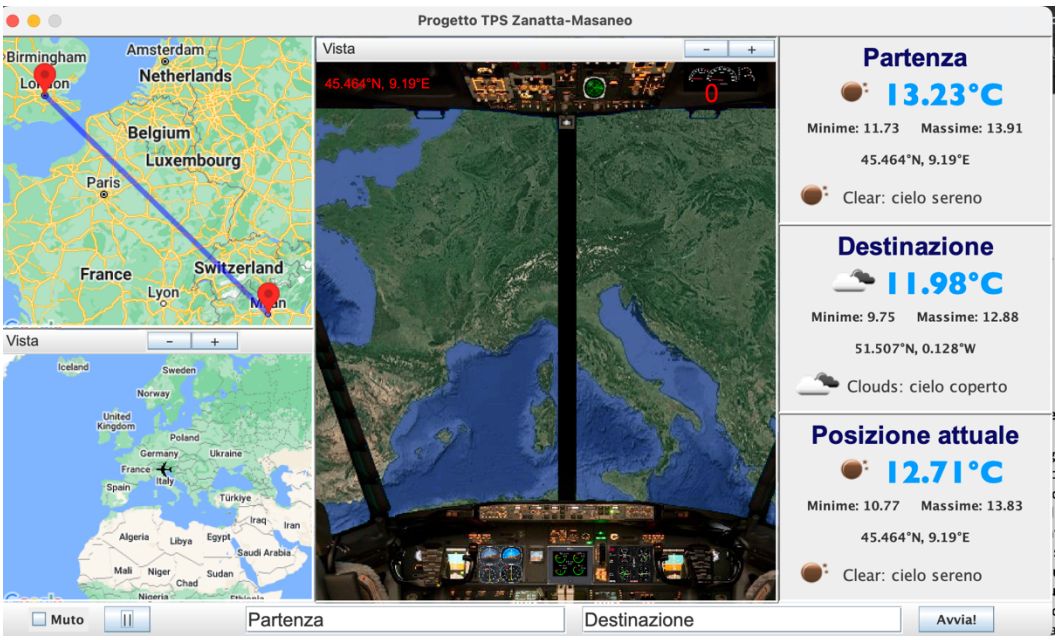
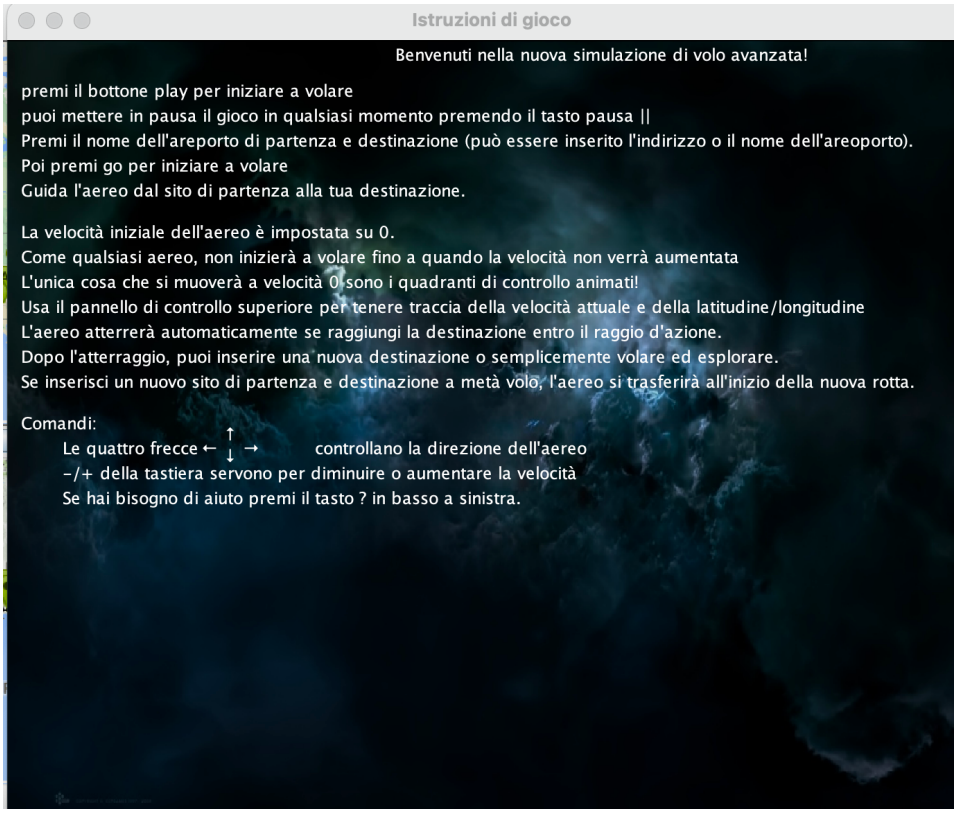
Progetto TPS Zanatta-Masaneo AirSimulation

Indice

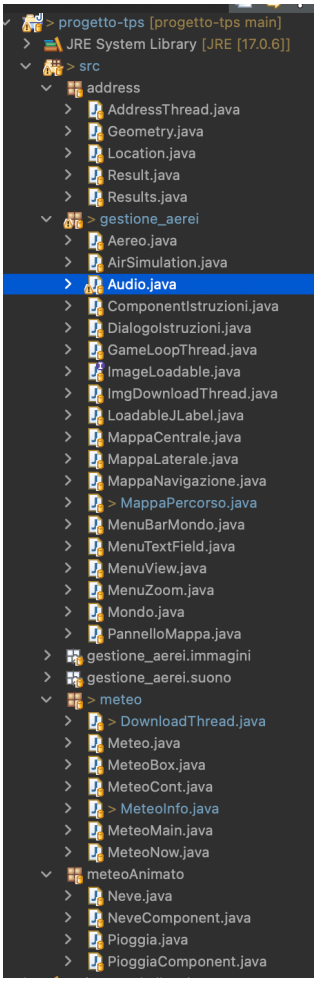
1. Descrizione del progetto

Il progetto che abbiamo realizzato è un simulatore di aerei iterativo. Esso permette all'utente una volta avviata l'applicazione di poter scegliere un aeroporto o città di partenza e d'arrivo. Una volta scelti e avviata la simulazione, nella cartina in alto a sinistra apparirà l'itinerario di volo da seguire mentre in basso a questa la posizione dell'areo in tempo reale.
Al centro viene visualizzata la cabina di comando in cui il pilota ovvero l'utente può avere una visione dettagliata e realista di dove si trova ed il suo spostamento.
Per muovere l'aero una volta acceso, si utilizzano le 4 frecce per direzionarsi ed i tasti + e – per aumentare o diminuire la velocità.
Una volta arrivati a destinazione apparirà un messaggio e sì potrà far partire una nuova simulazione come in qualsiasi momento della precedente.
Infine a destra sono visualizzate varie condizioni meteo ovvero: la nostra posizione attuale, l'aeroporto di arrivo e quello di destinazione, esse sono in costante aggiornamento per dare piena consapevolezza all'utente delle condizioni in cui l'aereo sta viaggiando.
Ad inizio gioco vengono visualizzate le istruzioni di gioco per spiegare ai nuovi utenti come funziona l'interfaccia ed il funzionamento.





2. Sviluppo del progetto



Il progetto è stato sviluppato tramite la libreria swing con cui sono stati fatti i vari pannelli, input e bottoni del programma.
La voce del copilota è invece un insieme di suoni che vengono inseriti all'interno tramite java.applet.
Per fari si che tutte le finestre comunicassero in modo corretto fra di loro sono stati utilizzati più thread, uno per ogni pannello per far si che ogni aggiornamento potesse venire in modo simultaneo e non sequenziale provocando rallentamenti e poca fluidità all'applicazione.
Non è stato utilizzato uno specifico design pattern ma invece vari package in cui ognuno contiene una macroarea del progetto essi sono:

- Address: contiene la gestione degli indirizzi per poterli aggiornare nella mappa durante la partita, sfrutta le api di Google Maps e salva i dati all'interno di classi autogenerate
- Gestione Aerei: è il pacchetto più grande, contiene le varie interfacce di gioco in particolare la classe mondo, mappa i vari comandi i suoni e racchiude i pannelli fra loro è quindi il main component. Le classi che iniziano con il nome Mappa sono invece quelle che fanno uso delle api di maps per caricare le varie cartine nei pannelli, ogni secondo viene fatto un update di tutti i componenti.

- Immagini: contiene tutte le immagini di gioco statiche ed anche un'easter egg per quando si tocca il polo nord
- Suono: contiene tutti i suoni del copilota e dell'aereo che poi vengono presi dal package gestione_aerei
- Meteo: è il pacchetto che si occupa della parte destra del programma, si occupa di fare le chiamate alle api del meteo, prendere le immagini che rappresentano la situazione meteorologica per le varie località che vengono indicate poi nella partita. Alcune classi sono generate per fare il mapping dei dati ricevute, Metobox è invece l'interfaccia grafica di ogni rappresentazione meteo mentre Metromain le racchiude tutte.

3. Utilizzo delle API

Le API utilizzate per la creazione del progetto sono state 2:

- http://api.openweathermap.org
  - Quest'Api viene utilizzata per reperire le informazioni meteo dei vari luoghi, quando viene effettuata una get request oltre ai vari parametri di cui si vogliono ottenere i dati, esempio si specifica latitudine e longitudine della posizione esatta di cui conoscere i valori, bisogna aggiungere l'APPID che permette di utilizzare un token di chiamata
  - Esempio di risposta: {"coord":{"lon":73.7793,"lat":40.6338},"weather":[{"id":803,"main":"Clouds","description":"nubi sparse","icon":"04d"}],"base":"stations","main":{"temp":17.22,"feels_like":17.06,"temp_min":13.32,"temp_max":20.77,"pressure":1008,"humidity":79},"visibility":10000,"wind":{"speed":5.66,"deg":50},"clouds":{"all":75},"dt":1681682861,"sys":{"type":1,"id":4582,"country":"US","sunrise":1681640107,"sunset":1681688062},"timezone":14400,"id":5122279,"name":"Inwood","cod":20}
  - La risposta avviene come si può notare in formato json, successivamente vengono quindi convertite in oggetto java tramite la librerira GSON

```java
public class DownloadThread extends Thread {
    private final MeteoInfo info;
    private final URL url;

    public DownloadThread(MeteoInfo info, double lat, double lon) throws MalformedURLException {
        this.info = info;
        url = new URL("http://api.openweathermap.org/data/2.5/weather?&lat=" + lat + "&lon=" + lon +
                "&units=metric&APPID=6498c9bbba6235eba0bfb10829f70f48&lang=it");
    }

    @Override
    public void run() {
        try {
            URLConnection connection = url.openConnection();
            InputStream in = connection.getInputStream();
            String json = IOUtils.toString(in);
            System.out.println(json);
            MeteoNow now = new Gson().fromJson(json, MeteoNow.class);
            info.displayWeather(now);
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

- https://maps.googleapis.com/maps/api/staticmap
    - Ritorna le immagini delle varie cartine che vengono inserite a sinistra dell'applicazione
    - Tramite la classe ImageIcon si pasa direttamente l'url con la get request ed essa ritorna l'immagine che viene poi inserita all'interno del componente
    - Come la precedente api anche questa necessità di una specifica chiave personale per effettuare richieste

4. Le classi generate Java

Le classi che sono state generate si trovano specialmente nel package meteo, queste sono state fatte grazie a tool online che permettono di generare una classe dal json da cui poi verrà fatto il mapping attraverso la libreria GSON