CSU11013 Programming Project

Group 27 – Report

Scott McNally
Alexander Judge
Grigoris Ioannis Manitaras
Emmelia Klefti

April 13th 2023

Overview - Features:

Given a dataset of United States flights in January 2022, we have developed a program that can present the data in a user-friendly manner while performing calculations to present additional information to the user.

Upon launch the user is presented with a map of the U.S populated with pins of the most popular cities/airports based on number of flights, while a search bar and list are also present so they can search for any desired city present in the dataset. After selecting a city it is added to a list. Up to 6 cities can be added in this way. A "Help" button is also present that displays a pop-up with instructions. (Screenshot 1)

After forming their list of cities, the user can select one of three buttons: "Clear" removes all cities from the list, "Compare Selected" brings them to the Dashboard page and "View CO2 emission" brings them to the environmental report page.

Dashboard: this page contains three charts: (Screenshot 2)

- A pie chart of the sum of flights that were diverted, cancelled or arrived normally.
- A line graph of the sum of late flights (more than 50 minutes) for each day of the selected week. Users can select the desired week of January (1-4) through a slider below the graph.
- A bar graph with the total arrivals to each of the selected airports. Users can zoom in or out using a slider below the graph as some airport combinations can yield drastically different numbers.

CO2 Emissions: this page contains the following elements: (Screenshot 3)

- A list of the airports selected by the user accompanied with the number of trees (millions) needed to offset the carbon emission of flights departing from the airports.
- A pictograph visualizing the numbers included in the list above using different numbers of a tree graphic.
- A bar graph with the estimated megatonnes of CO2 emissions per airport.

The user can select to switch between these two pages before heading back to the main screen containing the map to re-select different cities/airports to compare.

Assessment of required components:

Component	Implementation
- code to read in the data from a file and place it in classes	Upon launch, the program reads through the provided csv file of flights and creates a variable of our custom class Flight for each row (flight). An ArrayList of flights is then used to store all flights in memory while the program is running.
- code to select a subset of this data	In the main Map screen the user is able to search for cities in the provided search bar and select from a list of available cities that correlate to their search. In addition,

	they can select cities from the map pins and see all cities on the list if they have not searched through the search bar. All data presented in the Dashboard and CO2 Emissions pages are the result of this initial selection.
- code to draw the data to the screen	Both the Dashboard and CO2 Emissions pages use a combination of giCentre Utilities and controlP5 libraries to create pie, bar and line charts visualising data from the selection of cities from the user. We have developed the pictograph of trees required to offset the airports' CO2 emissions. The graphs can have their data changed during runtime and most can be manipulated using sliders (zooming/selecting data to focus on such us specific weeks).
- code to handle user commands.	The search bar handles user input from the keyboard and sends it to the searching algorithm. The mouse can be pressed to select cities from the list or pins and select buttons to navigate through the screens. It can also be held to alter sliders and change the appearance of values in graphs.
- code to put everything together	The program functions through screens which can be accessed by clicking on different buttons. Once cities are selected, users can navigate back and forth between the Dashboard and CO2 Emissions pages before heading back to the main Map screen that will clear their list of selected cities so they can select new ones. Additionally, a gif is played on launch until all data is loaded into memory and a notification is given to the user once the program is ready to be used.
- additional functionality	We perform calculations to estimate the amount of CO2 being emitted due to all departures and their distances from the list of selected airports. We then provide an estimate of how many trees need to be planted to offset January 2022's emissions for each.

Individual Contribution:

Name	**Work Done**
Scott McNally	Created the loading screen which allowed us to display something while the 500 thousand flight entries were being loaded in. This used a background thread so that the data could be loaded in asynchronously to the draw loop. Other features were also initialized within this such as the maps and graphs. Created the search bar. This was an implementation of the text field from the CP5 library which allows the user to specify which airports they want to compare by searching for their name. Each airport that contains the text that the user enters is put into a list displayed below the search bar.

	This creates an autocomplete feature as the user is not required to enter the full name of the airport they are searching for. Implemented UI elements related to the search bar such as the autocomplete list, selected airports list and clear button. The autocomplete list utilizes controlP5's list box which allowed me to create a dynamic scrollable list from which the user can select flights from. I created the selected airports list box which was later reused in other sections of the program like the dashboard. Created a cohesive design for the team to work from by selecting logos and color patterns which suited the look we wanted for our program.
Alexander Judge	Created main map class. The map's focus was to add a simple but useful quick method to compare different states' airports. Used a screenshot of an open-source map found online as the actual map image itself. Made it so the map was on the initial screen on startup. Later, I gave the map a border to give a more cohesive feel to the map being drawn on the main screen. Map screen also contains all 32 custom pins, and their respective airports. Created pin class to interact with the map. Users can click on pins, adding them to an ArrayList for comparing. Added hover and clicking input handling with pins to give feedback to user on what exact pin they would be adding and the last pin added to avoid confusion. Added 32 custom pins to map, all using discrete coordinates which were gotten through a simple mouse press debug log of location on screen. Created Index button dummy class for further integration to allow user to index between different screens.
Grigoris Ioannis Manitaras	Implemented UI elements such as sliders using controlP5, pop-ups and notifications using UiBooster, a Button Class used to add buttons throughout the program and custom window titles per screen. Implemented the switching of screens so that multiple interfaces could be presented on one window. Created the "Late Flights" line graph on the Dashboard page – a function of class Flight isLate determines whether a flight has arrived late or not by comparing CRS arrival and actual arrival times and adding late flights to the line graph dataset. Linked with a slider so that the user can select which week of January to see data for their selected cities. Structured and written part of this report including the Overview and Assessment of required components sections.
Emmelia Klefti	Created the status of arrival flights pie chart and the number of arrivals per airport bar chart on the Dashboard page – using the getData method the program checks whether the flight is diverted or cancelled and updated the status array accordingly and if the flight was carried out as expected the arrival number of the airport is incremented. Created the expected carbon emission bar chart of the departures from the specified airports using the fact that on average a plane uses 3.2 kg fuel per second and that every 1kg of fuel produces 3.1kg of CO2. This is implemented in the Flight class as a method getCO2emission which uses

	the arrival and departure times to calculate the duration of the flight and its CO2 emission. All bar charts have a slider that can be used to zoom the y-axis and therefore get a closer reading of the value shown. Using the expected CO2 emission, I created a pictograph of the trees needed to offset the CO2 emitted using the fact that on average each tree offsets 26.6kg of CO2 on average. Each small tree on the graph represents 5 million trees. A method in the Flight class (getTreesNeeded) was implemented to calculate the trees.

Problems Encountered:

- Since loading the data provided from the disk all the time is not efficient, classes Flight representing a single flight and Flights representing a dataset of Flights were created and all data is loaded at launch. A loading screen was added until this operation completes and a notification is sent to the user once it does.
- Comparing/calculating and displaying data from all individual airports at once would result in frame drops and overall poor performance – mitigated by setting a maximum of 6 airports users can select to compare data from.
- Some graphs would not render correctly after switching back and forth between screens. To mitigate this, we only set graph data when a user decides to switch screens. All appropriate graphs for that screen are populated before being drawn.
- We clear the list of airports selected after the user returns to the main Map screen due to an issue caused when data is retained and the user selects more by searching.
- Adding all airport pins on the map would cause it to be rather cluttered and not very friendly as the user would not be able to make out the map underneath clearly.

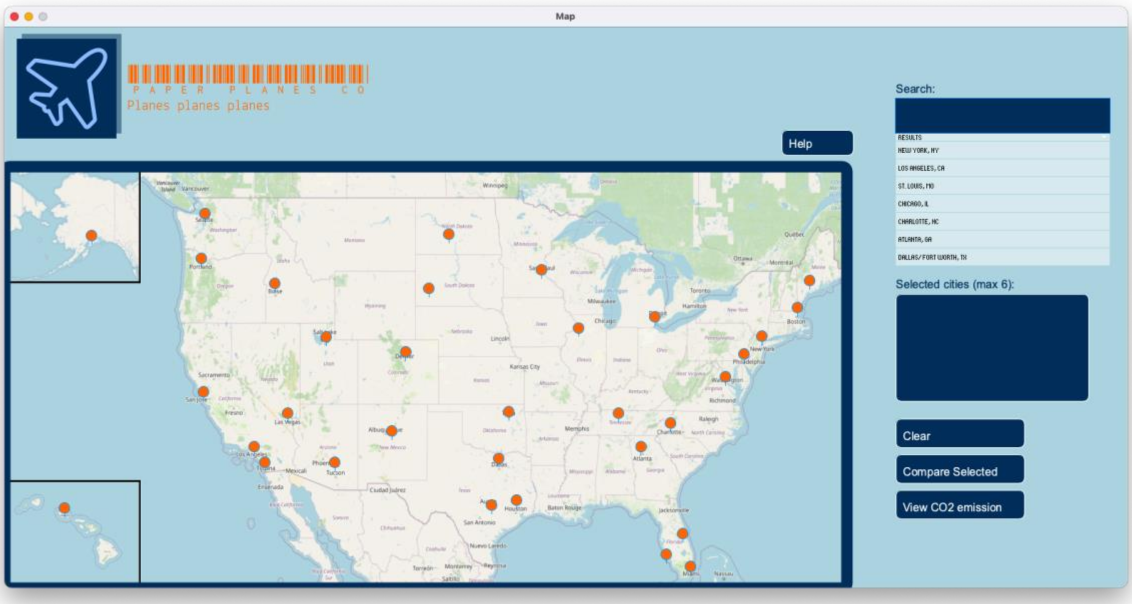
Sources:

Method of calculating CO2 offsetting - encon.eu.

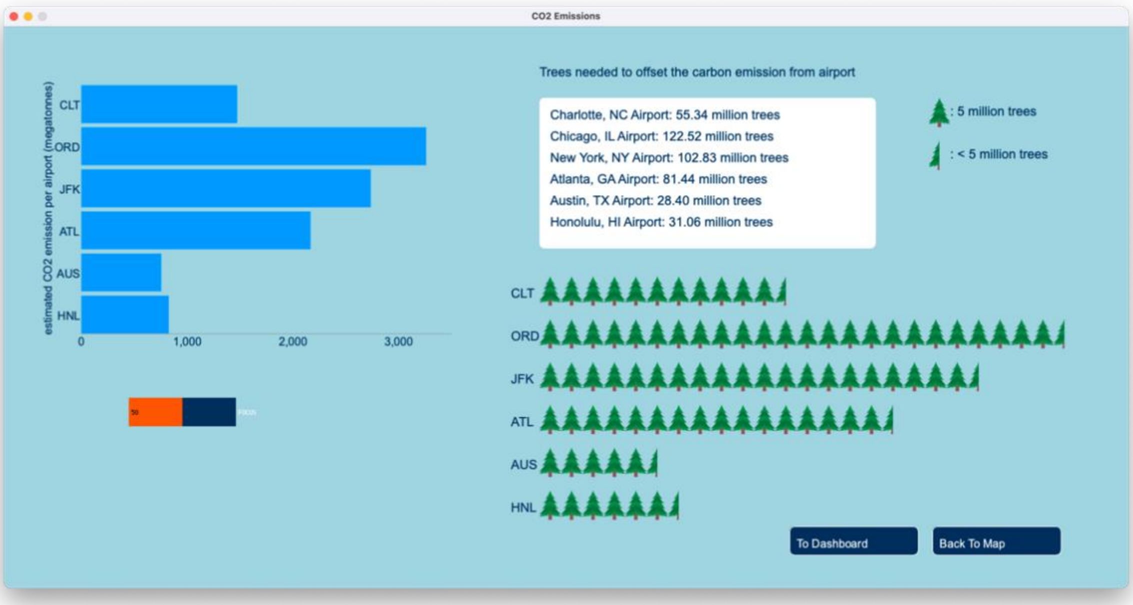
Assets:
-Pine Tree Clip Art - vecteezy.com
- U.S.A Map (Screenshot) – https://www.openstreetmap.org/
- Airplane GIF by Valentin Kirilov - dribbble.com
- “Paper Planes” logo created on logo.com

Libraries:
- controlP5 by Andreas Schlegel – UI Elements
- giCentre Utilities by Jo Wood and Aidan Slingsby – UI Elements
- gifAnimation by Patrick Meister, Jerome Saint-Clair – UI Elements
- UiBooster by Nick 'Milchreis' Müller – UI Elements

Screenshots:

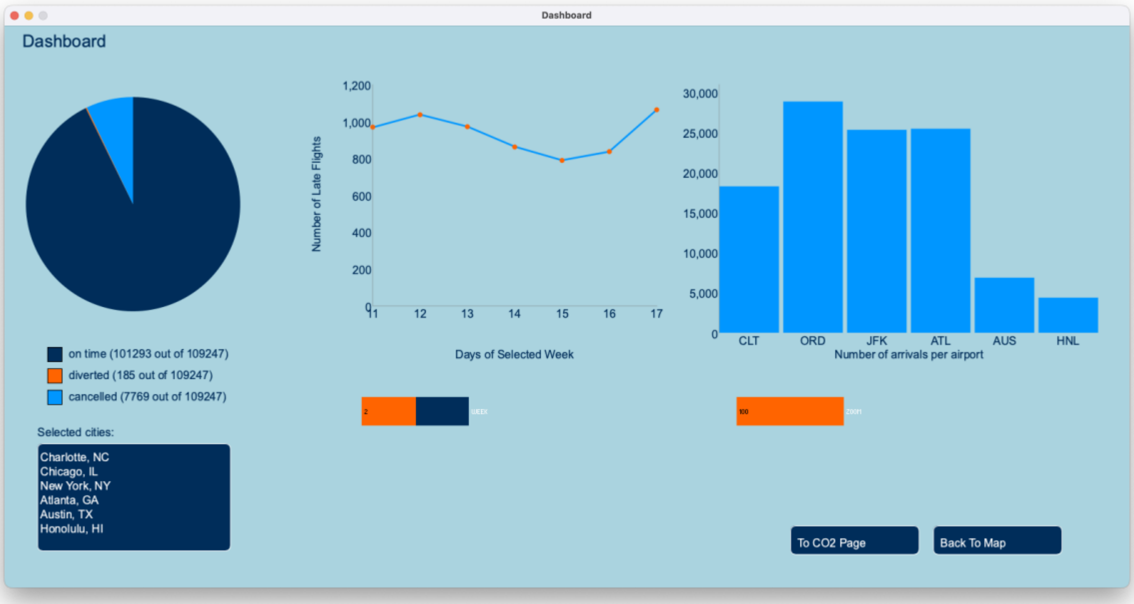


Screenshot 1



Charlotte, NC Airport: 55.34 million trees
Chicago, IL Airport: 122.52 million trees
New York, NY Airport: 102.83 million trees
Atlanta, GA Airport: 81.44 million trees
Austin, TX Airport: 28.40 million trees
Honolulu, HI Airport: 31.06 million trees

Screenshot 2



- ■ on time (101293 out of 109247)
- ■ diverted (185 out of 109247)
- ■ cancelled (7769 out of 109247)

Selected cities:

Charlotte, NC
Chicago, IL
New York, NY
Atlanta, GA
Austin, TX
Honolulu, HI

Screenshot 3