Air Traffic Control Simulation

Team Flow:

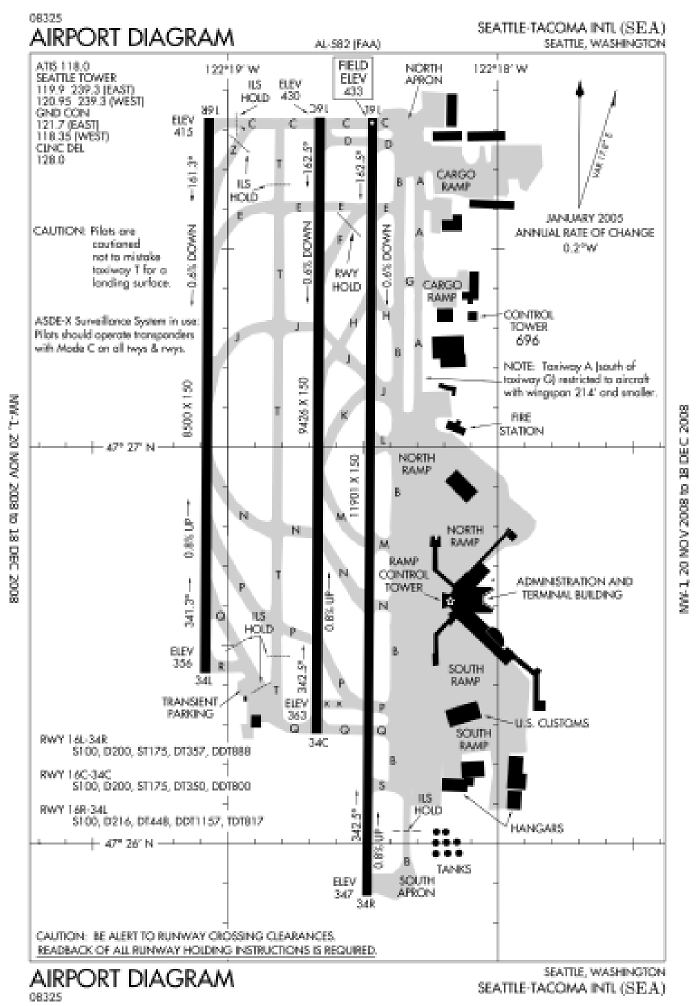
Members: Daniel Bragaru, Drew Loukusa, Ruslan Shabura, Sean Chen
Professor: Johnny Lin
Class: CSS 458
Date: May 12th, 2018

Project Description And Development Plan

Introduction:

In 2017 Sea-Tac was rated the 9th busiest airport in the United States, and has more than 16,000,000 enplanements each year. According to the data from the Federal Aviation Association, in 2017 the airport served over 46.7 million passengers, had grown 0.96% up on aircraft movements, and 16.22% up on air cargo in metric tons from 2016. It has been a concern that the increasing air traffic congestions slow down the take-off and landing coordinations, worsen passengers' experience, and decrease the efficiency of the airport. The current protocols and 3-runway airport is no longer able to maintain a satisfactory air traffic control performance level.

Figure 1



08325
AIRPORT DIAGRAM
AL-582 (FAA)
SEATTLE-TACOMA INTL (SEA)
SEATTLE, WASHINGTON
ATIS 118.0
SEATTLE TOWER
119.9 239.3 (EAST)
120.95 239.3 (WEST)
GND CON
121.7 (EAST)
118.35 (WEST)
CLNC DEL
128.0
FIELD ELEV 433
NORTH APRON
CARGO RAMP
JANUARY 2005 ANNUAL RATE OF CHANGE 0.2°W
CAUTION: Pilots are cautioned not to mistake taxiway T for a landing surface.
ASDE-X Surveillance System in use. Pilots should operate transponders with Mode C on all twys & rwys.
CONTROL TOWER 696
NOTE: Taxiway A (south of taxiway G) restricted to aircraft with wingspan 214' and smaller.
FIRE STATION
NORTH RAMP
RAMP CONTROL TOWER
ADMINISTRATION AND TERMINAL BUILDING
SOUTH RAMP
U.S. CUSTOMS
TRANSIENT PARKING
HANGARS
TANKS
SOUTH APRON
RWY 16L-34R S100, D200, ST175, DT357, DDT888
RWY 16C-34C S100, D200, ST175, DT350, DDT800
RWY 16R-34L S100, D216, DT448, DDT1157, TDT817
8500 X 150
9426 X 150
11901 X 150
47° 27' N
47° 26' N
122°19' W
122°18' W
CAUTION: BE ALERT TO RUNWAY CROSSING CLEARANCES. READBACK OF ALL RUNWAY HOLDING INSTRUCTIONS IS REQUIRED.
NW-1, 20 NOV 2008 to 18 DEC 2008

The Air Traffic Control Simulation, also often referred as the ATC Simulation, is a common practice to simulate all the possible parameters involved in the air traffic control performance at an airport. It has been ubiquitously used in air crew training, control navigation, and traffic control modeling. The goal of the project is to develop a 2-dimensional air traffic control simulation model to provide a simulated visual environment for the researchers to investigate the key factors that impact air traffic control, to simulate real-time control performance, and to explore possible solutions to improve the current control protocols and procedures for Sea-Tac.

In this project, we aim to model the flow of traffic (specifically commercial jets) at and around an airport. We will simulate most aspects of the airport from above in 2D. These aspects will include: Jets flying to the airport, Jets taking off, jets flying holding patterns, jets landing, jets taxiing to and from runway to terminals, jets being loaded and unloaded, serviced and fueled. Aspects to consider are congestion, delays due to weather, emergency landings.

In order to accomplish the project goal, we will study the current ATC simulations and procedures, learn about necessary aviation terminologies, research mathematical theories, and design a quantitative model. The project includes six phases: 1) project design, 2) model and theory consolidation, 3) prototype and development, 4) testing, 5) verification and validation, and 6) final analysis and presentation. The system will be run in Python, and it should be able to provide a visualized simulation of air traffic, and allow user adjustable variables, and provide instantaneous flight data and information.

Model Description:

<u>Plane Object:</u> (Variables):
Each plane/jet object will have its own variables to track the following: Fuel; Number of passengers; Weight; Time spent in each phase of the airport process; Location (long,lat); Its' airport process status; Emergency status (engine out or low fuel for example); Current heading; Speed; Altitude; Type (passenger or cargo); Fuel burn rate and History. History will log each variable at each time step.

Altitude will be represented by shading the airplane: Light shades are higher, darker are lower. Emergency status will be represented by changing the color of the plane: Yellow for low fuel, Red for mechanical problems or medical emergency on board.

Planes will burn fuel at rates based on altitude, weight, speed (cruising, descending) and can change status from normal to low fuel to emergency (10 min of fuel left).

There is no set limit on how many plane objects there can be at one time.

<u>Airport Process Statuses:</u>
This status is to tell us where in the process of using an airport each plane is in.
Planes can be in multiple statuses at once, can be initialized in statuses: A, TE.
Status A: Plane is in the air. Plane enters this phase once it leaves the runway
Status L: Plane is landing. Plane enters this phase once it receives clearance to land
Status TX: Plane is taxiing
Status G: Plane is on the ground. Plane enters this phase once it touches the ground
Status TE: Plane is at a terminal. Plane enters this phase once at a terminal
Status TA: Plane is taking off. Plane enters this phase once it leaves the taxiway to the runway.

Air Traffic Control Simulation

<u>ATC Object:</u>
The ATC object will hold the following data variables: A list of all planes in the airspace of the airport and on the ground at the airport. This list will also assign each plane a landing priority based on a number of factors: Fuel left in plane, time already spent waiting in holding pattern, potential emergency statuses (engine out) and altitude/ distance from airport.

The ATC object will issue commands to plane objects telling them what to do. The ATC object will essentially be the main logic of the simulation. There will only be ONE ATC object.

<u>Terminal Object:</u>
The terminal object will hold a few variables: Bool for if a plane is at the terminal, variable for holding a plane object. Mostly this object will have methods for doing the following actions ON planes: Fueling a plane, loading a plane with passengers, loading a plane with cargo if the plane is a cargo plane.

There will only be a set number of terminal objects, (to match the terminals that SEA-TAC has). There will be a separate “terminal” for cargo planes which won’t be a terminal but rather a cargo loading area.

<u>Simulation:</u>
The simulation will be made up of N time steps and at each time step the following will happen:

- For each plane in the sim:
    - For planes in the air:
        - Location will be updated based on heading and speed
        - Fuel will be updated based on weight, speed and altitude, and burn rate
        - Heading will be updated based on path needed to align with runway or holding pattern or avoid other jets
        - Weight will be updated based on remaining fuel, cargo on board or num passengers,
    - If a plane in the air is in range to request landing, it will send a landing request ATC
    - If the plane is at a terminal and is ready to taxi it will send a taxi request to ATC
    - If a plane is on the taxiway and is ready to takeoff it will send a takeoff request to ATC
    - If a plane is on the taxiway and is ready to go to terminal it will send a go-to terminal request to ATC
- The ATC will:
    - Take in requests from planes in the sim
    - Issue the appropriate instructions for planes based on known factors
        - Instruct the plane or planes with the highest landing priority to land

        - Instruct planes requesting to land to enter holding pattern or land
        - Instruct planes requesting to taxi to runway to hold or begin taxiing
        - Instruct planes requesting to taxi to terminal to hold or begin taxiing

    Update landing priorities for each plane in sim
- The terminals will:
    - For planes that are at a terminal:
        - Refuel the plane: Planes fuel will be updated
        - Board passengers or cargo

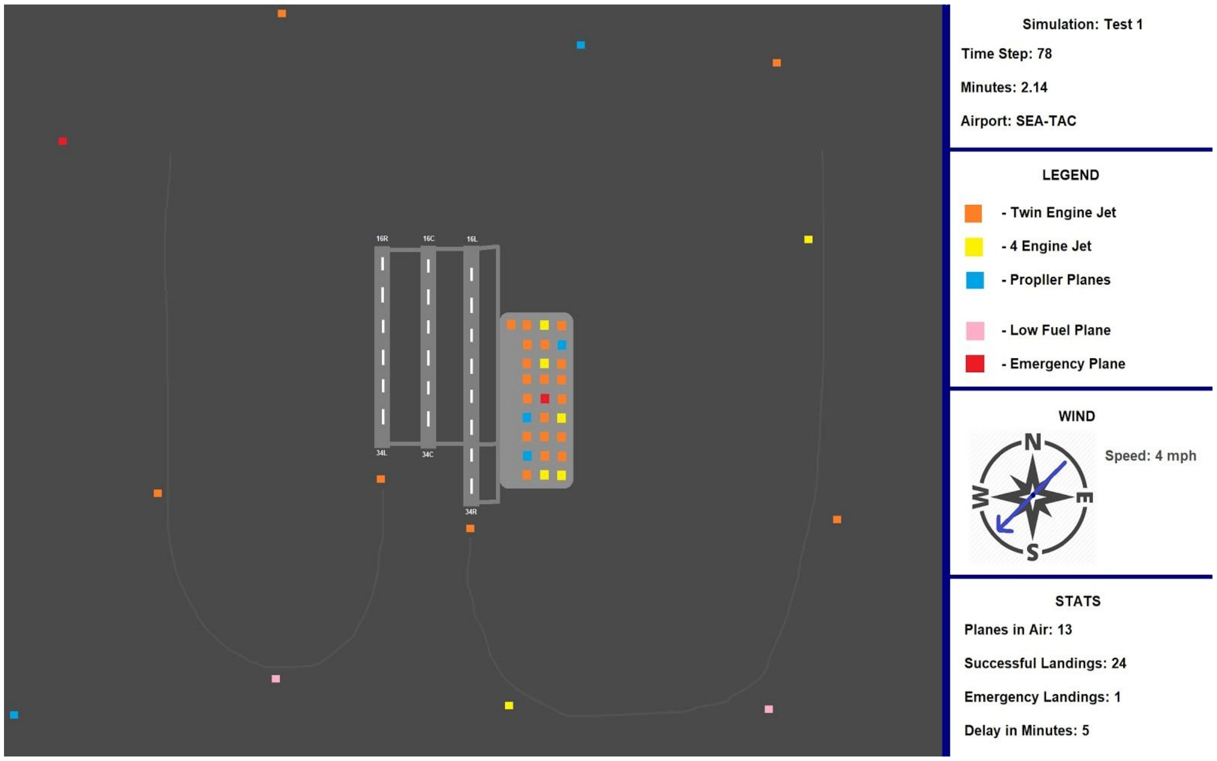
The planes will fall in a queue, and will maintain distance in between the plane in front. Emergency planes take priority, and therefore ruin the timing on the que. The planes in the queue will need to enter a loop situation (holding pattern), to realign with the runway, this will give us a sense of how air traffic is delayed and changed. The path to a runway is determined by the direction of the wind. The planes always land into the wind, or the best available angle between runway and wind: The wind direction will be represented by arrow on a compass, right around where the legend is for the simulation.

Simulation time, and step will be displayed, and the number of planes currently in the air as well. The model will have two runways to match the two runways that SEA-TAC has. One is slightly longer than the other and will be used by heaver planes that require more runway to takeoff and land.

Delays: There will be several types of delays. A few examples: Pilot is late (plane can’t take off): This will cause plane to be rescheduled into takeoff queue at a later time, upsets takeoff queue. Emergency landing: Upsets landing queue and takeoff queue. Weather Delay: Severe crosswinds can make it extremely difficult to land, this would upset the landing and takeoff queues.

Visuals: Will be composed of two primary visuals: 1. ATC overview: This will show planes heading towards the airport and will show an area of 100 x 100 miles. (See figure 2). (Should this be scalable?). 2. Airport overview: This will show how planes are moving around on the ground. *(See figure 3).*

Air Traffic Control Simulation



Simulation: Test 1

Time Step: 78

Minutes: 2.14

Airport: SEA-TAC

LEGEND

🟧	- Twin Engine Jet
🟨	- 4 Engine Jet
🟦	- Propller Planes
🟪	- Low Fuel Plane
🟥	- Emergency Plane

WIND

Speed: 4 mph

STATS

Planes in Air: 13

Successful Landings: 24

Emergency Landings: 1

Delay in Minutes: 5

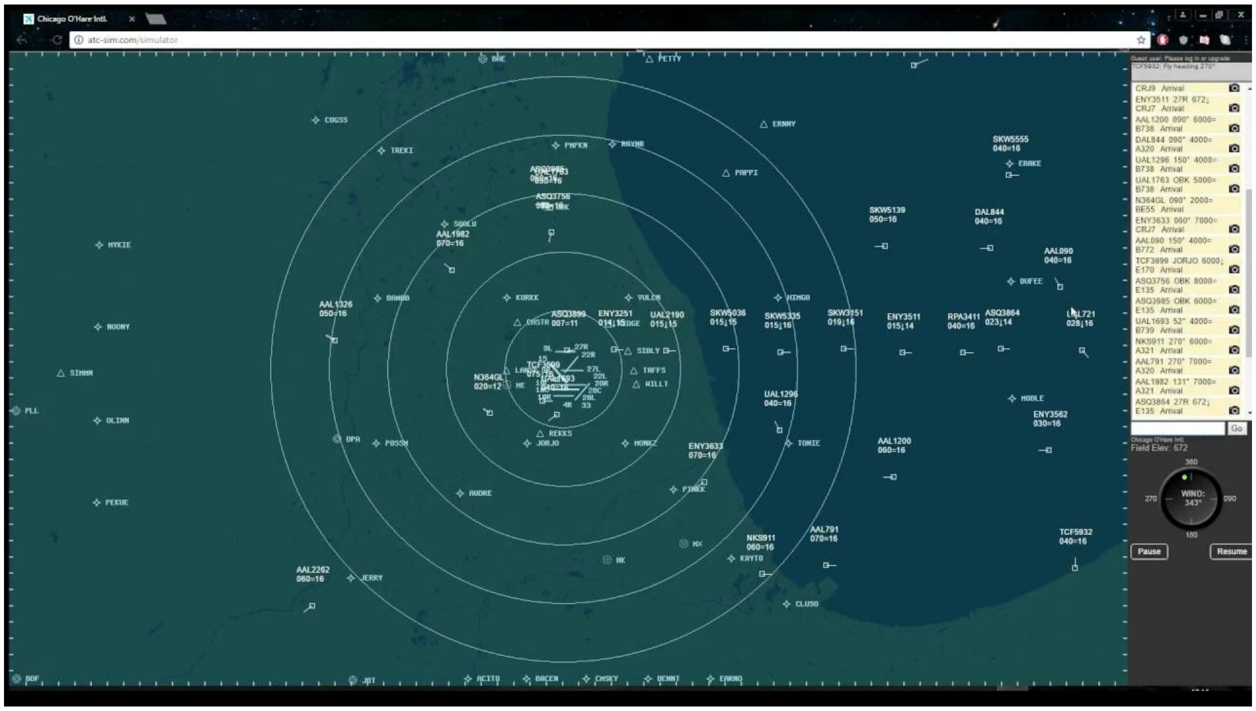


Figure 3:

Details that will not be modeled: Realistic shape (all planes will be rep. with a generic plane icon).

Air Traffic Control Simulation

Analysis:

Metrics and Quantities:
Amount of time that jets wait on the taxiway before taking off, the amount of time that jets spend in holding patterns waiting to land (clearance to land), the amount of time jets spend waiting on the taxiway waiting to go to a terminal, the amount of time jets spend at terminals, the amount of time that jets spend at airport from landing to takeoff,

Questions for Model:
How do delay events affect the wait times of planes on the ground and in the air? If a plane needs to make an emergency landing which will interrupt ATC scheduling, how will that affect the wait times of planes? How accurate are the delay times when compared to a real world example?

What the Model Can Tell Us:
This model can tell us how wind direction affects the landing path , and will help visualize that transition. The model can also help us understand what are the side effects of emergency landings on the normal air traffic, and it will help visualize the behavior which regular trafic has to follow in order to ensure a free-landing for the plane in emergency. In many cases, this behaviour requires a constant loop around the airport, at a given altitude, until the runways are clear, for our simulation it would be just a loop without the altitude aspect because we are on a 2 Dimensional space. Once the emergency is cleared, the regular trafic can proceed with the queue and continue the landings, but of course there has to be some kind of delay for the regular traffic, and this model will help calculate that. In the real-world, based on this delay, Airlines would have to adjust their schedules, or mobilize back-ups in order to ensure a timely execution of upcoming flights which rely on current planes in the air.

Verification / Validation:

- We will get informed about rules and flight patterns established by the FAA, and we will follow them to make our model as accurate as possible.
- We will also compare against available simulations and videos found online to get the best results possible.
- We will try to get a record of air traffic data from Sea-Tac, if possible, and analyze the flow of events during a regular-operational day.
- If data-records are not available, we will try to come up with our own data, and spend a few hours at the airport where we can conduct some analysis on the traffic landing and taking-off from the airport.
- Some key components we would look for are:
    - Average time between each landing.
    - Average time between each takeoff.
    - Number of planes coming into the airport over a specific amount of time,
    - And the relation the number of planes coming out of the airport over that same amount of time.
    - The size of the planes which are landing.

Testing:

Our project will use Unittest framework as it comes built in with Python, and has a lot of documentation to allow us to get our testing done throughout the stages of the project. We might end up trying Py.Test or nose just to see which one might work better with outrgroup. Meanwhile its going to be very important for everyone who is working on coding that they document thoroughly. This will help whoever is going to be writing the test cases/scenarios to have a good grasp on the intended functions/outcomes of said code, so they will be able to write fully functional test suite(a set of test cases ran together). One cool thing we found from reading the Unittest documentation is the setup() and teardown() modules that we could potentially use if our program would need them. setup() allows for us to literally setup the state of our program to be tested, without setup from our end. Meaning to create files, states, close files, etc. Also, teardown() would remove any files or erase content/close.

Overall, testing should help us verify the results are true while validating our system satisfies the project requirements. We will be using the second naming convention for our test modules. Ex: circle.py, circle_test.py to help group our files and keep everything close to each other as it should be. In terms of testing, we will make sure we test for upper/lower bound scenarios, make sure that our program accounts of outliers, and incorrect value types.

Personnel:

Drew is very familiar with Python programming and development. He has also been conducting lots of extensive research on project idea explorations, models and simulations discussions. He will be putting most of his efforts into developing Python codes and debugging.

Daniel’s driver character seamlessly fits into our team dynamics, and he is in charge of checking team progress, collecting ideas, and push through the development. He has built project framework, come up with the project design, and determined each team member’s role and tasks.

Sean is devoted to studying the mathematical theories and algorithms for the program. He will be working on developing the analytical models for the ATC simulations. He will also conduct the project analysis, collaborate with Drew and Russ to ensure the system aligns with the design, and is valid to mathematical models and theories.

Russ is interested in testing, developing testing manuals, and working on deliverable I/O API samples. He will be working with the other three team members to polish the design, test the system, and verify and validate the project.

Air Traffic Control Simulation

Technologies:

Communication: We mainly use discord to communicate as its super easy to set up, allows for pinned messages, and voice chat as well. So far we have used it almost every day to communicate with one another, have team meetings hosted in the voice chat we we vote and discuss the direction of the project. Plus it's nice since we have it on all of our laptops/computers/phones so we can use it anywhere we are.

File Sharing: We are using Google Drive for any documents and storage of files as it's a free platform that allows multiple users to work on the same documents. Makes work so much faster and it's all real time. Our group has also agreed to use GitHub for coding version control and to effectively share the same code.

Python:
We can use the Python module "Turtle" and "Paint" for drawing our airport and handling the animation of planes moving. Unittest for our testing framework, possible py.test or nose depending how we like it at the time of trying it. There is always the possibility of more modules we will use as we start working on the project and happen to run in to something that could prove to be very useful.

Benchmarks:

May 14th - Project Kickoff
The project development plan will be finished. The team has a clear vision about technologies, methodologies, and project design.

May 19th - Model and Theory Consolidation
The team should have consolidated the theory discussion and analytical research, and start developing the Python codes.

May 27th - Product Complete - Version 1.0
The team should complete the first draft of the system, and start testing, verification and validation.

May 29th - Product Complete - Version 2.0 (Milestone)
The team should complete the second draft of the system, and finish the testing, verification and validation.

May 30th - Final Presentation and Product Release
The team should have the final version of the project done, and get ready for the final presentation.

References

[1] Figure 1:
https://en.wikipedia.org/wiki/Seattle%E2%80%93Tacoma_International_Airport#/media/File:Seattle-Tacoma_International_Airport_diagram(2).svg

[2] Figure 2:
Daniel Bragaru, using Pain.

Google Doc: Method of Checking Member Investment

https://docs.google.com/document/d/11IbKRAUWuD0_J0cCp9FtjZug__kz3CkEtngIkVgHBeE/edit