

# Air Traffic Control Game

Shun Kito  
779991  
Tietotekniikka  
2019  
22-04-2020

## General description

I created an air traffic control game. The game contains multiple different difficulty levels, each having a different number of aircrafts appearing in the game, speed of the winds and frequency of appearing aircrafts, which differs from initial plan, but I think this is much better, where the game become more enjoyable. The runways are different from each other, but they aren't crosswise. Only one aircraft can be on the specific runway at the same time, which makes the game a bit more difficult. The aircraft could also be kept above the field at different height levels.

The game is over, when any aircraft crashes (more accurately later) or every aircraft's journey succeeds.

I was aiming at difficulty of at least moderate where a gui and multiple windows are required. And I implemented such a game, but I have more functionalities in the game, e.g. different options for airport, so I think I have achieved the difficulty of demanding.

## User Interface

The game can be started by choosing the file named ATCG\_GUI.scala and run as scala application.

First the program will ask the player to type his/her name.

Secondly, it will ask to choose the airport.

Thirdly, it will ask to choose difficulty level of the game (Easy, Normal or Hard).

In different levels, there are different amount of aircrafts appearing in the game and frequency of appearing aircrafts will also differ.

After choosing the level, the game begins.

In the game, player's target is to succeed every take-offs and landings by commanding aircrafts.

To both taking off and landing, the aircraft has to be stand-by (waiting).

After being stand-by, the aircraft can take off or land.

Only one aircraft is allowed to be waiting at the specific runway at the same time.

Flying aircrafts wait landing by making circle in the air. And commanding flying aircrafts don't need to include the name of the runway. Flying aircrafts can stand-by only when they are close enough to the destination airport.

Aircrafts on ground wait take-off by standing at the runway, given by the player.  
 To take-off, the player has to type in the runway from where the aircraft will take off.  
 To land, the player has to type in the runway to where the aircraft will land.  
 The player can also command flying aircrafts to change their angle in radian (between the line made by aircraft and airport and x-axis), speed and height.  
 The commands of these will be presented down below.

Continue commanding aircrafts until the final one succeeds.

The game will end, when every aircrafts have taken off and landed or any aircraft crashes.  
 In this game, to crash means to be out of the field, because then the aircraft can't be commanded anymore.

The player will gather points by succeeding in take-offs or landing. When any aircraft crashes, huge amount of points will be taken away.  
 Here's point list:

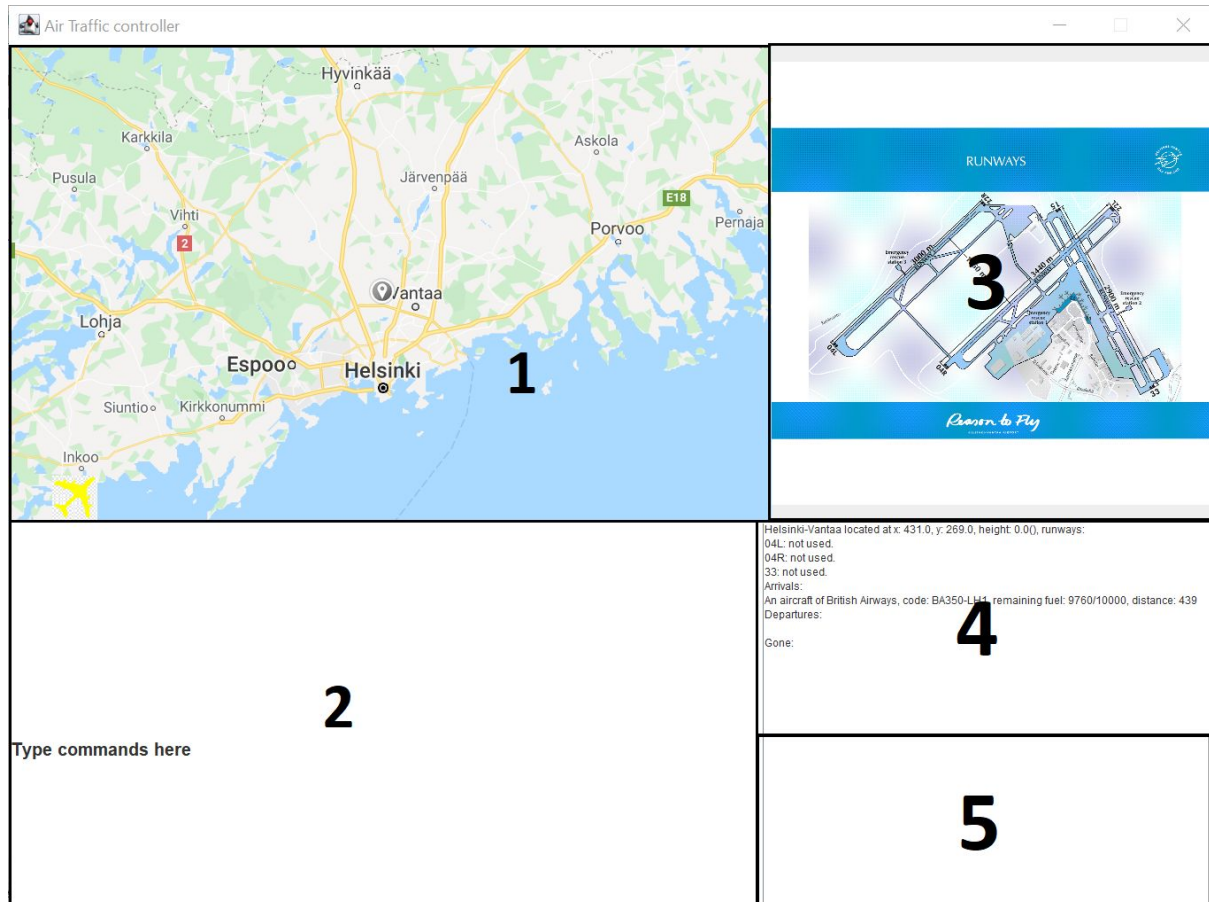
	Landing	Take-off	Crash
Easy	+1 p	+2 p	-10 p
Normal	+1 p	+4 p	-20 p
Hard	+1 p	+10 p	-50 p

The list of commands (every commands, codes and the names of runways can be typed in with capital letters or small letters):

- Landing:
  - LAND CODE RUNWAY
  - e.g. LAND AY350-NH1 33
- Take-off:
  - TAKE-OFF CODE RUNWAY
  - e.g. TAKE-OFF AY350-HT1 04R
- Stand-by (when flying)
  - STAND-BY CODE
  - e.g. STAND-BY JL350-TH1
- Stand-by (when on ground)
  - STAND BY CODE RUNWAY
  - e.g. STAND-BY AY350-HF1 04L
- Change height
  - CHANGEHEIGHT CODE HEIGHT
  - e.g. CHANGEHEIGHT CZ350-BH1 8000
- Change angle and speed
  - CHANGEAANDS CODE ANGLE SPEED
  - e.g. CHANGEAANDS AA350-NH1 2.5 5

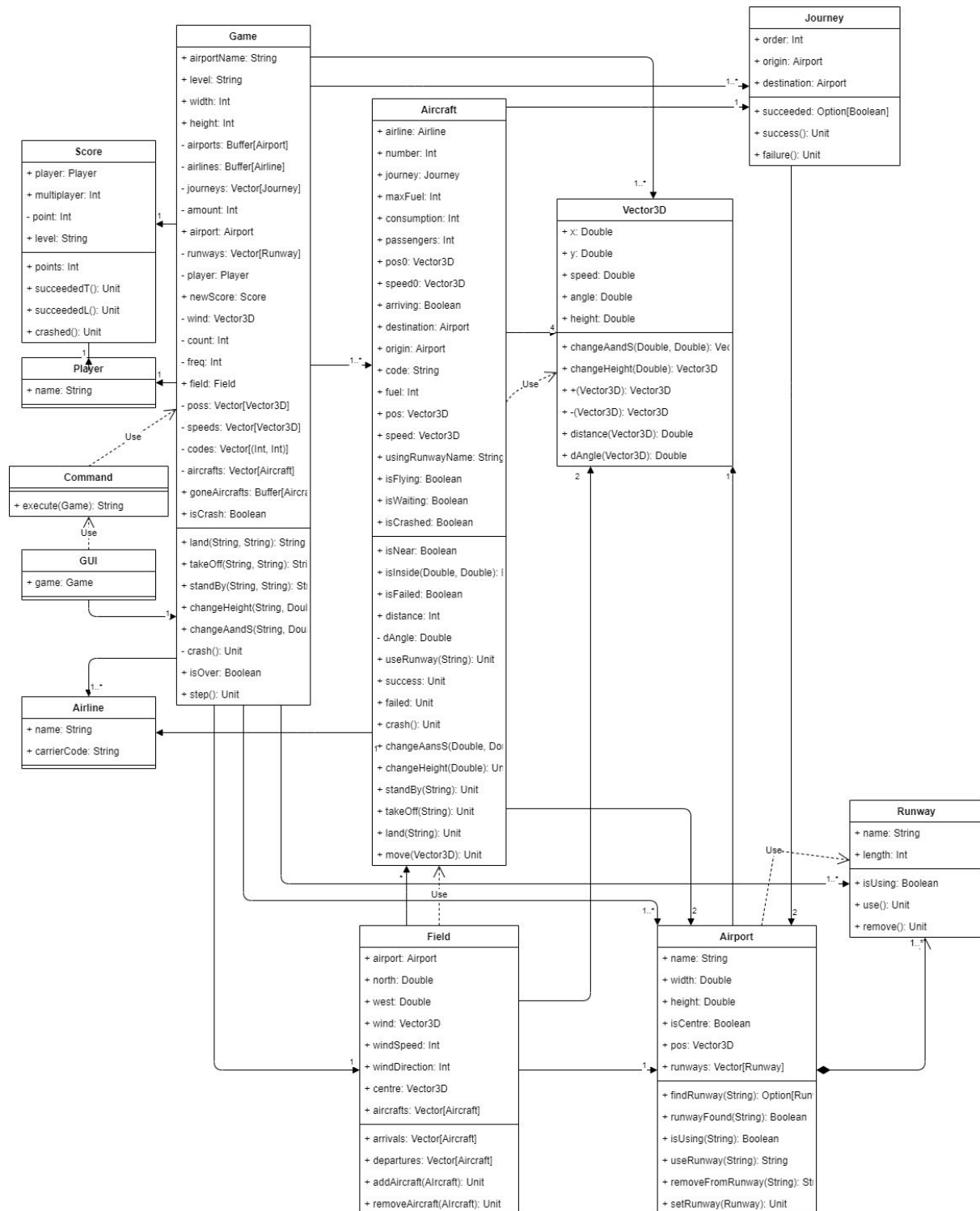
The informations of airport, runways and arriving, departing and gone aircrafts are shown in their own field. Usable runways can be noticed by text: not used. The information of an aircraft contains the name of airline, codename, remaining fuel and distance to its destination.

Also the information of typed command will be shown in its own field.



1. The field (map) where the aircrafts are drawn
2. The field where the player types the command in
3. The map of the runways in the airport
4. The informations of airport, runways and aircrafts
5. The information of typed command, also the score will be shown here after the game.

## Program structure



The classes and object will be splitted into main sub-parts like this:

- User Interface
  - ATCG\_GUI (Object)
- The part which puts together smaller parts or essential part as a part of a game
  - Game, gathers all the needed classes and uses them to maintain the game. Essential method are e.g. every kind of executing methods like land and take

off. Method step is one of the most important method in this class, which advances the game by adding turn by 1 and moving every aircrafts in the field.

- Command, receives a string containing command, code etc. from user interface and executes it by calling methods from the class game.
- Score, stores points gathered in the game. The class game uses methods in this class to update gathered points.
- Player, stores data of a player.
- Smaller parts representing things related to airtraffic
  - Aircraft, represents an aircraft. It has code, airline and journey. It also has a position on the map and a speed. When called by the game, executes the method in this class which are e.g. land and take off. Methods land, takeoff and standby have restrictions to do wanted things, e.g. land method works only when the aircraft is stand-by and given runway is not occupied.
  - Airline, represents an airline. It has name and carriercode.
  - Journey, represents a journey from an origin airport to a destination airport. It also has as a variable order, which indicates a order of the journey. The class also stores whether the journey has succeeded or not, or is it uncompleted. It has methods for updating state of variable succeeded, which are used when landing or takeoff succeeds or fails.
  - Field, represents a field size of a map in the game. In this game, the player can only command aircrafts which are in this field. In this class, methods addAircraft and removeAircraft are essential to manage which aircraft is currently in the field.
  - Airport, represents an airport. It has some runways and it can be in the centre of the map, that is a destination for arriving aircrafts in the game and an origin for departing aircrafts, or not. Methods useRunway and removeFromRunway are essential in this class, because these methods are used when some aircraft lands or takes off.
  - Runway, represents a runway. It has a name and length. Only one aircraft is allowed to be on one runway at the same time. Methods use and remove changes the state of the variable using.
- A tool needed in the game
  - Vector3D, represents a vector in this game. In this game this is used either as a position or a speed. It has coordinates x and y, height, speed and angle. All the method related to vectors are implemented here. In this class, + is one of the most important method, which adds two vectors and returns new one, because it will be used every turn.

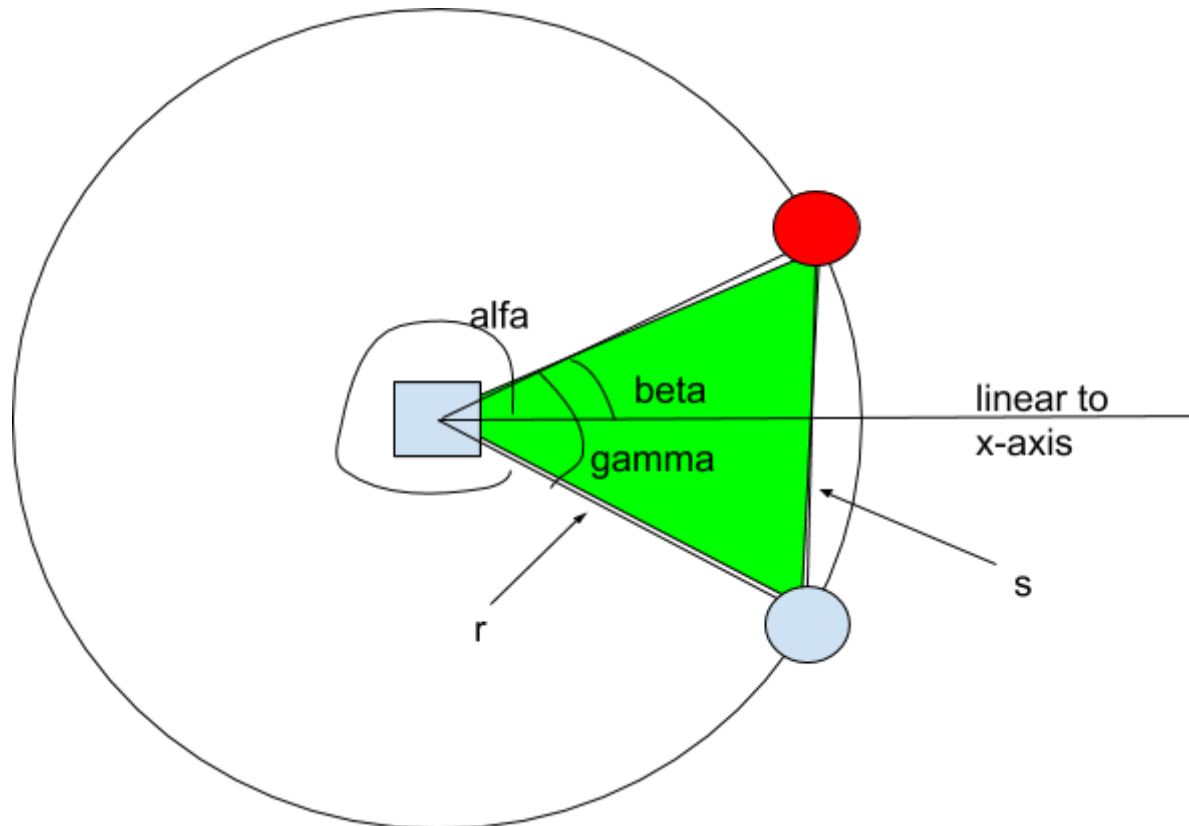
## Algorithms

In my game, the most important algorithms are related to vectors and I'm going to show few of them.

1. Vector representing position and speed.
  - a. Because flying and moving aircrafts are the essential part of the game, I had to represent aircraft's position and vector in some way. I took example from

project Asteroid given as a material in this course. There asteroid's position and speed was represented by vector.

- b. This works like this:
  - i. Both position and speed are vectors.
  - ii. When an aircraft moves, the program adds values of x and y of speed to x and y of position.
2. How to change the position of an aircraft, so it makes a circle around an airport.
  - a. This was one of the most demanding problem in my project.
  - b. I wanted aircrafts to have same speed as before starting make circle, so I used variable speed of vector representing speed.



- c.
- d. In this picture:
  - i. Blue box in the centre is the airport
  - ii. Blue circle is the aircraft in time t
  - iii. Red circle is the aircraft in time t+1
  - iv. r is radius, which is distance between the airport and the aircraft
  - v. s is speed of the aircraft
  - vi. alfa is the angle between the aircraft and the airport in time t
  - vii. beta is the angle between the aircraft and the airport in time t+1
  - viii. gamma is the difference of angle alfa and beta
- e. Firstly, we know r and s. Because we know the position of the aircraft and the airport we can calculate the angle alfa, using atan2. Now we know the angle alfa, r and s.
  - i. If (x, y) is the coordinate of the airport and (a, b) the coordinate of aircraft, then:
  - ii.  $\alpha = \text{atan2}(y - b, a - x)$

- f. Secondly, because we want the angle beta, we first calculate the angle gamma. It can be done using law of cosines. It looks like this:
  - i. Because it's a circle, the sides of the green triangle are both r, thus the green triangle is isosceles triangle. So we now know the length of base = s, and length of sides = r = distance. Thus the law of cosines looks like this:
  - ii.  $s^2 = r^2 + r^2 - 2r^2 \cos(\delta)$  and in the end:  $\delta = \arccos(1 - \frac{s^2}{2r^2})$ .
- g. Now we now both alfa and gamma, so we can calculate beta
  - i.  $\beta = \alpha + \delta$
- h. And we can now calculate new position of the aircraft:
  - i.  $a = r * \cos(\beta) + x$
  - ii.  $b = -r * \sin(\beta) + y$ , NB: minus is there, because in swing the y-coordinates increases opposite direction compared to "real" y-coordinates

## Data Structures

I used both mutable and immutable collections in my code. Mainly vectors and buffers. Buffers are used when some value had to be able to remove from the collection. Otherwise I mainly used vectors. In some cases I might use also buffer, where it wasn't necessary, because it's easier to write code using buffer, for me.

## Files and Internet access

In this game, only picture files are used. Object ATCG\_GUI handles it using java's imageio. These are not files, but the game uses the input from the player. The inputs like player's name, airport's name and level are received using scala swing's Dialog. Commands will be taken using swing's event Keypressed.

## Testing

I tested my game both by using unit tests and simply by playing the game. While playing the game, I mainly checked whether the graphical user interface worked correctly and also checked whether aircrafts moved correctly. For smaller things, I used unit tests for testing these.

Almost everything related to tests mentioned in the initial plan, is implemented and tested. And my code passed every and each tests. But I actually made more unit tests than I've expected initially. Unit Tests were made for the most of classes. Examples of existing unit tests are:

- Vector3D works correctly, by checking numbers
- Methods land and takeoff works correctly, by checking variables
- The command input by an user has correct form
- The score is updated after successful / unsuccessful landing and takeoff, and also possible crashes

And I tested e.g. these things by playing the game myself:

- The program warns user of using wrong command, when used one
- The status of aircrafts and airport is changed after using correct commands
- The status of aircrafts and airport is NOT changed after using wrong commands
- The informations of aircrafts are shown correctly on windows
- The information of aircraft that just came into or went out of the area is updated

## Known bugs and missing features

When the program is started to run, the program asks player to type his/her name. If the player doesn't type the name and shuts the window, program still continues to ask airport and although the player closes the window again, the program ask level. After that an error occurs, because the name of the player, the name of airport and the level is not given, the program is not able to create the game.

Also ending the game is not perfect. The game finishes making text field not to be able to type, but the program is still running behind. To end the program perfectly, the player has to close the window.

## 3 best sides and 3 weaknesses

3 best sides:

- The GUI: As a person, who likes travelling and aircrafts, the map used in the game and the picture of runways are really good.
- The functionality of the game: To be honest, the game works perfectly. I have tried to play same kind of game, online, and my game worked quite well compared to the online one.
- Math behind the code: I don't know if this is noticed very much, but I have used really much time to find the way to move aircraft to make circle. I felt so refreshed when the idea flashed into my mind.

3 weaknesses:

- The complexity of the code: Somewhere in the code, specially in the class game, there the code is quite complex, which leads to readability of the code. The expression of gone aircrafts is not good. This could be improved.
- The command is a bit difficult to type: They are so long. But I couldn't do anything for that, because the game needs much informations. There was one possible improvement for the length of the command. To remove the codename. But I just didn't want to do that, because it could lead to lack of being realistic. And I think it's much enjoyable, if there is codename. At least I enjoyed to type in AY350.
- The direction of the picture of aircraft is always same: It's a bit boring and looks strange, because every aircrafts have same picture and looks same. But I thought it isn't the essential part of the game, so I didn't implement that.

## Deviations from the plan, realised process and schedule

The process deviated quite a lot in the end. Initial plan was to make the GUI quickly and make testing easier, but this took really much time. Making GUI took almost a month, because of the problems in GUI and also other studies. In March I couldn't advance my



project. But after the exam week, I could do more, because one subject finished in that week.

I made basis of the every classes first and then GUI. After that I started to implement the essential methods like land, take-off and stand-by.

## Final evaluation

The game works as I wanted, in the end. So I would say, that the quality of whole game is very good. Although, there are some weaknesses, like occuring errors when the input is empty, but there are much more good points in the game compared to weaknesses.

The game could be implemented in the future, by making better GUI, making codes clearer, adding more airports etc. If the airports and aircrafts will be much more, it is better to make code, which can read files. Also saving the highest scores will be nice implementation. In some cases, my choice of collection could be improved e.g. changing buffer to vector.

If I could start the project from the beginning, I'll first say, that the time for making plan was too short. It could have been better to have more than week to think the plan. And then I would make more precise plan.

## References

[www.scala-lang.org](http://www.scala-lang.org)

Google maps

Stackoverflow

docs.oracle.com

[https://en.wikipedia.org/wiki/Law\\_of\\_cosines](https://en.wikipedia.org/wiki/Law_of_cosines)

<https://en.wikipedia.org/wiki/Atan2>

<https://www.mathopenref.com/coordparamcircle.html>

Picture of runways:

<https://www.privatejets.com/privatejets/images/airports/91899.gif>

[https://acukwik.com/runwaydiagram.ashx?Procedure\\_ID=23717](https://acukwik.com/runwaydiagram.ashx?Procedure_ID=23717)

[https://upload.wikimedia.org/wikipedia/commons/thumb/7/78/Heathrow\\_Airport\\_map\\_with\\_third\\_runway.svg/1200px-Heathrow\\_Airport\\_map\\_with\\_third\\_runway.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/7/78/Heathrow_Airport_map_with_third_runway.svg/1200px-Heathrow_Airport_map_with_third_runway.svg.png)

<https://images.fineartamerica.com/images/artworkimages/mediumlarge/1/5-pek-beijing-capital-international-airport-in-beijing-china-runway-jurq-studio.jpg>

[frankfurt](#)

<https://simpleflying.com/wp-content/uploads/2018/09/http-mashable.com-wp-content-uploads-2015-04-jfk-700x598.jpg>

[https://simpleflying.com/wp-content/uploads/2020/03/CDG-Runway\\_Layout-700x420.png](https://simpleflying.com/wp-content/uploads/2020/03/CDG-Runway_Layout-700x420.png)

Picture of aircraft:

<https://www.epicentrofestival.com/wp-content/uploads/2019/11/Airplane-Aircraft-Computer-Icons-Silhouette-Clip-A-Yellow-Airplane-Cliparts-720x720.jpg>

# Appendixes

Air Traffic controller


RUNWAYS

Return to Play


Helsinki-Vantaa located at x: 431.0, y: 269.0, height: 0.0(), runways:  
04L: not used.  
04R: not used.  
33: not used.  
Arrivals:  
  
Departures:  
An aircraft of Finnair, code: AY350-HN1, remaining fuel: 10000/10000, distance: 508  
An aircraft of Finnair, code: AY350-HP1, remaining fuel: 10000/10000, distance: 508  
Gone.

STAND-BY AY350-HN1 33

Air Traffic controller



RUNWAYS



Reason to Fly

Helsinki-Vantaa located at x: 431.0, y: 269.0, height: 0.0(), runways:  
04L: not used.  
04R: not used.  
33: not used.


Arrivals:  
An aircraft of British Airways, code: BA350-LH1, remaining fuel: 9400/10000, distance: 144  
An aircraft of Lufthansa, code: LH350-FH1, remaining fuel: 9900/10000, distance: 476

Departures:  
An aircraft of Finnair, code: AY350-HN1, remaining fuel: 9980/10000, distance: 577  
An aircraft of Finnair, code: AY350-HP1, remaining fuel: 10000/10000, distance: 508  
An aircraft of Finnair, code: AY350-HF1, remaining fuel: 10000/10000, distance: 508


Gone:

The take-off succeeded.

Air Traffic controller



RUNWAYS



Reason to Fly

Helsinki-Vantaa located at x: 431.0, y: 269.0, height: 0.0(), runways:  
04L: not used.  
04R: not used.  
33: not used.

Arrivals:  
An aircraft of Lufthansa, code: LH350-FH1, remaining fuel: 9300/10000, distance: 320  
An aircraft of China Southern Airlines, code: CZ350-BH1, remaining fuel: 9800/10000, distance: 379

Departures:  
An aircraft of Finnair, code: AY350-HP1, remaining fuel: 10000/10000, distance: 508  
An aircraft of Finnair, code: AY350-HF1, remaining fuel: 10000/10000, distance: 508

Gone:  
An aircraft of Finnair, code: AY350-HN1, remaining fuel: 9880/10000, distance: 926  
An aircraft of British Airways, code: BA350-LH1, remaining fuel: 8800/10000, distance: 312

GAME OVER!!!  
Player Shun's score:  
Level: Easy  
Points: -9