**Air Traffic Control System**

Team SegFault

Lindsey Wingate, Sadie Sorenson

Csci 363 – User Interface Design

December 8, 2016

**Abstract**

Team SegFault was contracted to produce an Air Traffic Control System model by the US Government Federal Aviation Administration by the Contractor Dr. Emanuel S. Grant. The contract included specifications for the design. Several specifications were adjusted through time to meet the requirements of the GFAA. The main specifications were met on time, including a circular airspace, an icon for airplanes with a Collision Avoidance algorithm, and various scenario buttons to demonstrate how the model functions. Time restraints and resource restraints played a role in the development of this model, as well as the experience of the developers in the team.

1. Introducton

The Air Traffic Control Design Project was assigned to a group of developers in September of 2016. The goal of the project was to simulate the 50-mile airspace of an active airport. The display was required to mimic the radar viewing scope mounted on top of the airport tower. Additional lines at 10 mile intervals concentric with the airspace boundary line were included. At least two enter and exit points were required around the outer-most circle for controlled landings and take-offs. In the middle of the airspace, two runways for landing and takeoff were represented by doubly thick lines.

Aircrafts were to be represented by an icon of choice. Each aircraft was to have a visible ellipsoid area surrounding it at a 3-mile radius. At all times the aircraft’s ID, destination airport ID, control, altitude, speed, and heading are to be represented next to the aircraft.

A Collision Avoidance algorithm was required to prevent aircraft collisions. If any aircraft enters the collision avoidance circle of another aircraft, warnings are set off. If there is an aircraft collision, the user must be notified. If an airplane enters the airspace, it is then under the control of the Air Traffic Controller. The ATC may take control of the aircraft by clicking on its icon and typing in altitude, speed, and heading. This allows the ATC to control the flow of traffic as if he were speaking with a pilot.

Departing aircrafts are randomly created by clicking on a point within the air space circle. If there is a queue of airplanes leaving, the airplane must wait until it is granted take-off clearance. Arriving aircrafts are randomly created by clicking on a location outside the airspace circle and immediately head towards the nearest entrance to the airspace.

1. Background

When developing the air space design, we used the Agile Development model. Several key principles of Agile Development include completing each feature before moving on to the next, a fixed timescale, and requirements evolving with active team communication and decision making.

The project started on a fixed timescale with no negotiations. The project report deadline was always December 9; because of this deadline, the team worked in phases to complete the system. The first phase was the foundation. In order to add actions to the interface, a static design was required. The development of the circular radar airspace and its corresponding circles was completed first. The second phase included design aspects and creating the logo for the airplanes. The third phase included adding movement to the airplanes. Finally, the last phase was the implementation of a collision algorithm. As the team moved through these phases, assumptions were made about the visual effects and color choices. The key motion elements that were added to the interface became the main focus, which led us to make some changes in color.

1. The Project

As mentioned above, there were four phases of work done on this project. In the first phase, a circle class was developed with some difficult to show the ATC airspace. The 2D Graphics library was avoided because it did not allow manipulation of the shapes in the way that was required. Once the circle was established, inner-circles were added. An additional class was written to define the runways.

Once the basic airspace was created, design elements such as a background color and the sidebar for directions were added. Manipulating the colors of the objects was difficult at first; many sources had to be referenced in order to determine the correct syntax. Black was decided to be the best color for the background with white lines for the runways and airspace identifiers. These two high contrast colors provide a clear visual for any Air Traffic Controller using the interface with color blindness. Once the airplane logos and their individual airspaces were created, they were designated a deep orange color. This mantained the contrast between the airplane logos and the background.

After the static elements were set, motion was integrated into the interface. There were a few issues with the development of randomly creating planes when clicked outside or inside the airspace. The main problem was recognizing where the click occurred. The airspace circle was draw inside a box; if a user clicked within that box but outside the circle, it still registered as a click within the box. To solve this issue, a formula was calculated based on the coordinates of the entire screen. A click from the user passes coordinate information to the program which then compares those coordinates to the circle formula. If the coordinates were within the circle, then an airplane would be created departing. Otherwise, an airplane arriving would randomly appear. In this phase, the font sizes were also increased for better visibility. A timer was used to simulate the motion of the planes every few seconds. Each time the screen updated, the planes show in their new locations.

The final phase of development required a collision algorithm. In order to accomplish this, coordinates of planes were again used. Code was written to compare the airplanes locations any time the timer updated. If the planes are within a certain distance from each other, a warning in red, bold, capital font blinks on the sidebar. At this point an additional text box was added to the side panel for warnings. Additionally, if planes collide (their coordinates are equal and the ATC did not change their information soon enough) the planes stop and a notification in red, bold, capital font fills the warning textbox. Ideally, the planes would be replaced by a triangle instead of just stopping but time limitations forced us to stick with the sidebar notification.

1. Conclusion

This project was instrumental in learning JavaSwing. Development was made much easier by the NetBeans interface. Without this additional interface, we probably would not have completed the specifications on time. If time had permitted we would have written each line of code from scratch, allowing more specificity in the coordinate system. Similarly, a separate class for airplanes would have been developed. The airplanes were created within the circle class so they would show up in front of the airspace lines. If layering the foundation code and the objects that move was not an issue, the code could have been written in a much cleaner, precise format. Despite these difficulties and restrictions, the interface deliverable fills the specifications required per the US Government Federal Aviation Administration.