A Framework for Virtualizing Joystick Controls in a Flight Simulator Training Environment

Alex Redei*
Central Michigan University, Mt. Pleasant, MI 48859, USA

Sergiu Dascalu† and Frederick C. Harris, Jr.†
University of Nevada, Reno, NV 89557, USA

Abstract

In aviation there can be little room for error. This paper explores software arbitration of two joysticks controlled by two pilots, where each joystick is independent of the other and each pilot's actions are potentially equally valid. In such scenarios, it can be difficult to know which commands are valid, and which commands should be ignored. Inspired by historic conflict resolution scenarios in commercial aviation history, we developed a framework for virtualizing joystick commands from two multi-axis joysticks. The framework has been utilized in a two-person flight simulator, where different joystick conflict resolution techniques were modeled and evaluated. In this paper, we detail our framework for arbitrating conflicts in a multi-axis joystick system, thereby increasing the responsiveness of control input in a potentially conflicted state. Both the framework's hardware prototype and software system are described and the results of implementing and evaluating three joystick conflict resolution techniques are presented and discussed.

Key Words: Human-computer interaction, joystick, input mapping, flight simulator, pilot training.

1 Introduction

In a training environment, the learning curve can be greatly reduced by offering immediate and focused feedback to the person being trained. In the realm of pilot training, this feedback is especially crucial, as good training is of the utmost importance. The correct maneuvering and operation of an aircraft can be a life or death situation.

Motion flight simulators are an innovative training tool that can help with this problem. They help pilots learn the ropes of flying an aircraft, and train for component failures, without the risk of death.

The closer the simulation is to real life, the more meaningful the experience will become. This means that any simulation system must provide a constant feedback loop to its pilots. Such feedback can come in many forms, from audible alerts, to visual feedback, to the tactile responses of a control mechanism.

*Dept. of Computer Science. E-mail: redei1a@cmich.edu.
†Dept. of Computer Science and Engineering. E-mail: {dascalus@cse.unr.edu, fred.harris@cse.unr.edu}.

In the real world, planes are not just piloted by one person – there is always at least two pilots on any commercial airliner in the United States (per FAA's two-person cockpit rule [6]). European airlines, while not all strictly two-pilot planes, seem to be following this trend [13]. In order to keep the training consistent with the real world, we have to support two people in the cockpit as well: a pilot and a co-pilot.

But what happens when both are at the controls? Who should the simulator respond to? Who should get the feedback? Moreover, if everyone has a duplicate set of the same controls, whose commands should be listened to?

Clearly, a system is needed to consolidate and prioritize the various inputs being provided to the simulation software.

In this article, we discuss a methodology for handling the intricacies of a multi-person, multi-axis motion simulator using an approach developed in our lab. The rest of this paper is organized as follows: Section 2 is devoted to the aviation history of Air France Flight 447, from which much of this framework's inspiration has been drawn. Related works are presented in Section 3, hardware is described in Section 4, and software architecture and design presented in Section 5. The methodology of our conflict resolution software is detailed in Section 6. Section 7 addresses the experimental results we acquired from our work with the newly developed framework, and Section 8 wraps up our findings with several concluding remarks and directions of future work.

2 Air France Flight 447

Air France flight 447 is the tragic story of how poor software design and a miscommunication between the pilots can lead to catastrophe [3, 19]. Air France flight 447 departed Rio de Janeiro's Galeão airport on May 31st 2009, and was expected to arrive in Paris' Charles De Gaulle airport the following day. Flight 447 is interesting because geographically it crosses both the Atlantic Ocean and the equator, as depicted in the route map shown in Figure 1.

En-route to Paris flight 447 crossed the intertropical convergence zone, also known as the ITCZ, a band of powerful storms situated around the equator [14]. Storms in this region can reach 50,000 ft in altitude, well above the altitude commercial airplanes can fly at [10]. Violent thunderstorms form as air masses from the two hemispheres interact in this region. On the particular day of Air France flight 447's tragedy,

ISCA Copyright© 2019

Figure 1: Flight plan of Air France Flight 447 [4]

Figure 2: The Intertropical convergence zone [4, 8]

they had no choice but to fly through the storm. A picture of the ITCZ from July 12th, 2000 can be seen in Figure 2.

The decision to fly through the storm led to tragedy. Upon flying through the storm, devices called pitot tubes located at the nose of the aircraft iced over. A pitot tube is a critical device in aviation because it measures the airspeed as air moves over the body of the aircraft. For a period of 47 seconds, two of the three pitot tubes froze over and malfunctioned, reporting invalid airspeed information [3]. The onboard flight computer, being unable to reconcile the differing and invalid airspeed measurements, errored out, turning off the autopilot and returning control to the pilots.

Co-pilot Pierre Bonin took control of the aircraft upon discovering that the autopilot had turned off. He was unfamiliar with flying through the ITCZ, and his anxiety in this situation was apparent in the voice recorder. Co-pilot Bonin, feeling that the aircraft was losing altitude, made the fatal decision that day to command the plane to climb under manual control using his joystick, without communicating to his co-pilot David Robert. On an Airbus A330, which is the aircraft involved, there are two joysticks that control the pitch and roll of the aircraft, symmetrically located on the right and left side of the cockpit, that is called a "side-stick." An image of the Airbus A330

Figure 3: The Airbus A330 side stick [3]

side stick is shown in Figure 3.

David Robert, suspecting that the aircraft may have been in a stall condition, attempted to correct the situation by commanding the aircraft to pitch down. However, unbeknownst to co-pilot Robert, co-pilot Bonin was still commanding the aircraft to climb using his joystick, so the on-board computer was receiving conflicting information from the two side-sticks. A diagram showing how the side-stick is used to manipulate the control surfaces on the aircraft is shown in Figure 4. The software was programed to override the joystick inputs in such scomes, and in this case that meant the pilot's conflicting commands were cancelling each other out.

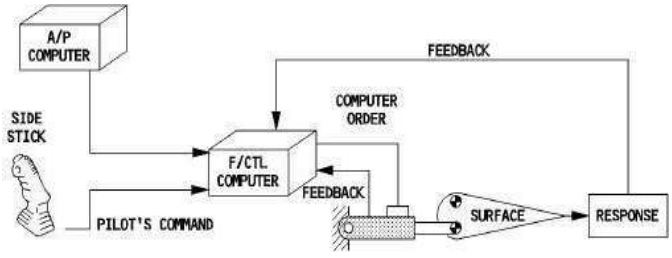

Figure 4: Details of the Airbus A330 side stick mechanism [9]

In the end, the tragedy of Air France flight 447 was completely preventable. Poor software design and a miscommunication between the co-pilots doomed the flight. The decision to command the aircraft to climb at such a high altitude, and the confusion over who was in control of the aircraft, led to a stall from which Air France flight 447 never recovered. Co-pilot David Robert's final words were recorded

at 2:14 am where he exclaimed "Dang it, we're going to crash. This can't be happening!" [19]. The impact into the Atlantic Ocean killed all 228 onboard [3].

In this paper, we explore different software techniques for joystick conflict resolution through virtualization, a technique we hope will reduce the likelihood of another situation similar to Air France flight 447 repeating itself.

3 Related Work

To control an aircraft in flight, a pilot has 6 main controls: engine speed, ailerons, elevators, the rudder, flaps, and spoilers [18]. A NASA-made diagram with the location of each of these controls is shown in Figure 5.

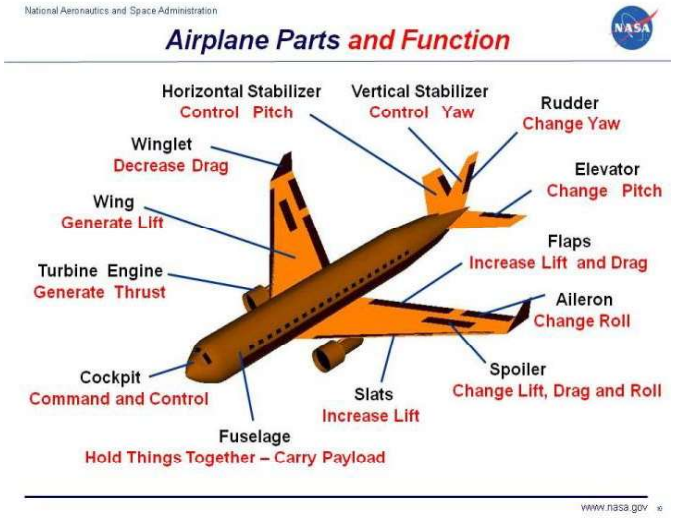

Figure 5: Critical parts of airplane flight [4]

These 6 controls allow the pilot to control the pitch, roll, and yaw of the aircraft (as well as its speed, and other aerodynamic properties). The pilot needs to keep constant mind of these controls, so that he or she can keep the plane flying and headed in the right direction.

As technology has improved, computers have been integrated into almost everything [12]. This is no different in the aviation industry. Nowadays, computer software is an essential part of keeping traffic organized in the air traffic controller (also sometimes abbreviated as ATC) [5, 17]. An example of this software's interface is shown in Figure 6.

Technology has changed not just the ATC, but also how the pilots fly planes. An auto-pilot can fly the plane for hundreds of miles without needing human input. In addition, technology has also changed how pilots are trained.

Airline pilots have used flight simulators as an essential training and educational device for decades [1]. Flight simulators have been in consistent use in commercial aviation since the 1960s, and are a safer alternative to training their pilots [2, 11].

Ironically, sometimes the simulator used for training can cost more than the actual airplane.

This paper explores different software mechanisms for addressing the problem of joystick conflict resolution. In terms of physical equipment, the problem of two-person pilot cockpits is replicated in a hardware prototype.

Figure 6: An example of the software used in the air traffic controller [5]

4 Hardware Prototype

In order to model a software framework for the joystick conflict resolution, first a hardware prototype was needed. We used a flight simulator for this. There are two input devices in our simulator, and each operator (pilot and co-pilot) has a duplicate set of controls. The existing simulator does not handle multi-axis joystick input well. For example, if there are two pilots (a pilot and a co-pilot) and both press the deploy landing gear button, then the simulator will be unable to distinguish who pressed the button (and hence to whose responsibility it was to be assigned to). This is similar to the problems encountered by Air France flight 447, as outlined earlier in Section 2.

The first control is a joystick. The joystick in the simulator has two axes (x, y) that allow the pilot to control the elevators and ailerons. The ailerons control the roll, and the elevators control the pitch of the aircraft. In the joystick, this input is obtained via two analog potentiometers. The two potentiometers capture the commands on the x and y axes. When the joystick is swung to the right or to the left, the resistivity of the potentiometer changes, and thus a different voltage is read from the joystick in that axis. This style of input conversion is called an analog joystick, and according to an ACM SIGCHI bulletin, this joystick would be classified as a multi-axis joystick [7].

In addition, there are 4 buttons that are mappable to various controls, and a 4-way hat switch which allows the pilot to look around the cockpit.

An image of the first joystick is shown in Figure 7 and a potentiometer is shown in Figure 8.

The next control is the throttle body. The throttle controls the thrust of the engines, by way of controlling the speed of the engines.

The throttle is a single-axis joystick that uses an analog potentiometer to measure the "throw" of the throttle. In

Figure 7: The 2-axis, analog, 4-button joystick [15]

Figure 8: The potentiometer inside the joystick x-axis. The potentiometer senses the axis position and conveys the information over 3 voltage regulated wires

addition, the joystick has two buttons that can be mapped to various functions (such as deploying/retracting the landing gear). The second joystick is shown in Figure 9.

There are exact duplicates of the controls described above inside the cockpit: one for the pilot, one for the copilot. Thus, we have a challenge, which set of controls should we use?

In a real airplane, the controls would typically be tied together via a mechanical linkage. However, in the simulator we have no such linkage. Thus, we need to create a system which can respond to inputs from both pilots with some sort of definable priority.

For the hardware portion of our solution, we used an Arduino Leonardo. We chose this board because unlike the very popular

Figure 9: An example single-axis analog joystick similar to the one we use, but not exactly the same model [16]

Arduino Uno, the Leonardo has an ATMega 32u4 chipset that can emulate HCI devices, and operate in USB-slave mode.

We found that we wouldn't be able to get an Arduino Uno recognized by the computer as a joystick without having to write our own Windows drivers. Instead, we found that using the ATMega 32u4's built in HCI device emulation allowed us to get around this issue.

To connect everything together, a wiring harness was used to convert the Molex connectors supplied by the manufacturer on the ends of the joystick, into jumper cables that can be plugged into the Arduino.

The six analog joystick inputs wire into six analog inputs, and the 20 buttons wire into digital inputs (via a shift register due to there being only 14 inputs available). A circuit diagram showing how all these ties together is shown in Figure 10. A picture of the hardware breadboard prototype is presented in Figure 11, and the wirings to the joysticks are shown in Figure 12.

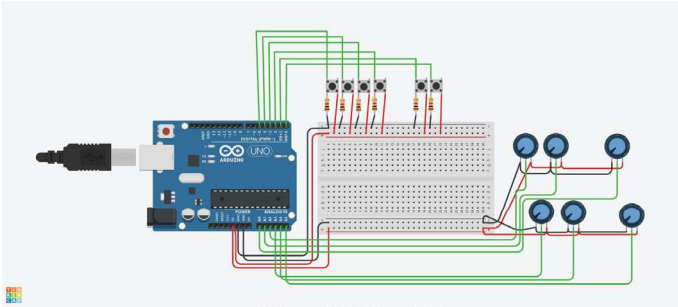

Figure 10: A depiction of the circuit diagram. Note that the breadboard is not used, except to consolidate the 5V power and ground lines

5 Software Design

First, the requirements of a software system that would process the multi-axis input from multiple joystick controllers are outlined in Table 1.

To better understand the interaction of the system, use case diagrams were created, as shown in Figure 13.

For example, in one of the use cases, only one operator is

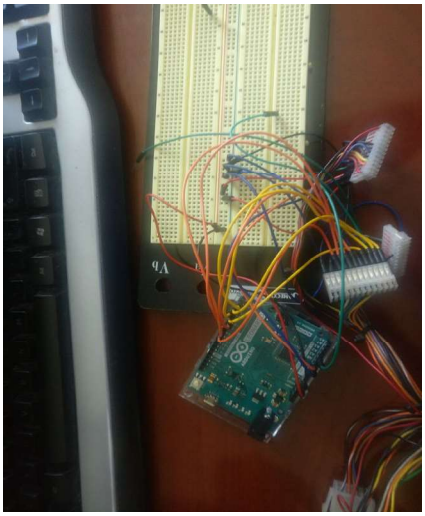

Figure 11: A photo of the actual hardware

Figure 12: Additional details of the actual hardware

pushing a control, say the deploy landing gear button. In this simple case, the controller maps the control command to a keyboard command and passes it along to the flight simulation software.

A more complicated use case scenario might involve both pilots providing input to the controller simultaneously, as shown in the MultiInput use case. In this case, the input of each pilot's control is tested against a weight, and combined together to form a "score" for that axis. Afterward, that axis is passed along to the simulation software via the USB joystick interface.

The code itself was written as a C++ 'sketch' in the Arduino IDE (v 1.6.5).

The analog inputs are read (those would be the axes: rudder, ailerons, elevators, etc.) and converted to a 10-bit digital value (in the range 0-1023).

The button presses are read via digital inputs hooked into a pull-down resistor. The software keeps track of the prior known state of the button. When the circuit senses a change in the voltage, the current state is compared against the prior known state. If the state has changed, we report that change to the OS, and override the prior known state with the current state. At the

Table 1: Software requirements

Requirement	Priority	Description
1	High	The software must be able to convert an analog axis input into a digital 0-1023 value.
2	High	The software must be able to convert button presses into a digital 0/1 value.
3	High	The software must be able to handle at least 4 analog axes of input.
4	High	The software must be able to handle at least 6 different types of button presses.
5	Medium	The software should be able to handle simultaneous inputs from multiple sources (analog axes, buttons, etc.).
6	Medium	The software should be able to map the analog axes to various software controls (such as rudder, throttle, elevators, or ailerons).
7	Medium	The software should be able to prioritize multiple inputs from the pilot and co-pilot.
8	Low	The software should be able to emulate keyboard presses.

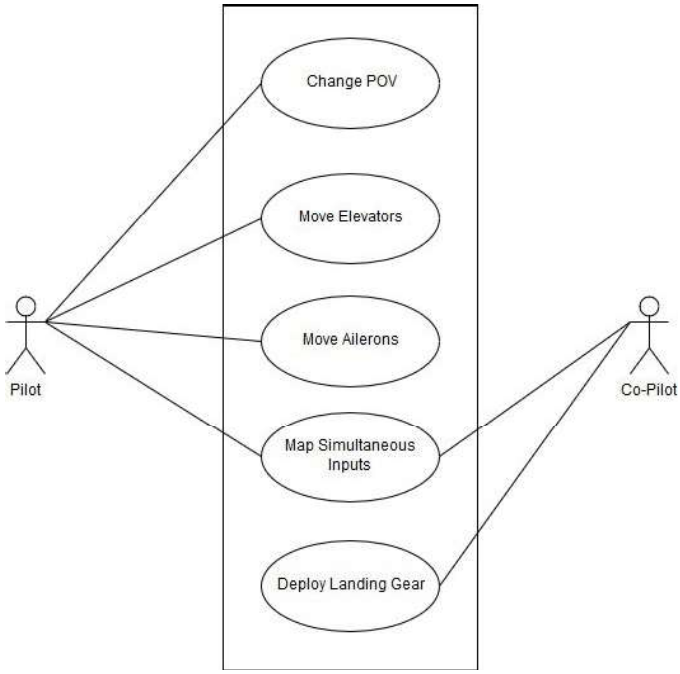

Figure 13: The use case diagram of the developed software

moment, we can detect changes as fast as 1/20th of a second. It may be possible to detect changes faster, but for our purposes (civil aviation simulation) this seems sufficient.

6 Methods Evaluated

When co-pilots David Robert and Pierre Bonin provided conflicting input through the Airbus's side-sticks during the Air France flight 447 flight, the computer handled the conflict by averaging the inputs from the two joysticks. This led to disaster, as discrepancies within the joystick system, known as dual-inputs, was resolved in the Airbus software by averaging the joystick axis values of the two joysticks. Since the joystick inputs were the exact opposite, the flight computer cancelled out the pilots' commands and did nothing as the aircraft plummeted into the ocean. The co-pilots struggled to understand why they were not in control of the aircraft. Each had assumed their joystick was in control.

There are several methods that could be employed in such scenarios. Using the virtual joystick controller described above, we replicated three different scenarios. The first is rather simple: it involves replicating the value-averaging technique of the Airbus A330. Simply put, the firmware takes a voltage reading on each potentiometer, and averages the two values across the joystick axes. This technique has the advantage that agreeable dual-input can multiply the speed of a turn, but it does not handle miscommunication well between the pilots.

The second technique implemented and evaluated was to simply make the left-seat joystick the master joystick. Whenever the left-joystick was active, commands from the right-seat joystick would be ignored. While this technique would have prevented the Air France flight 447 tragedy, what if the roles were reversed and the pilot in the left seat was in a confused state? Or what if the joystick in the left seat was damaged, providing incorrect input and overriding the commands of the valid right-seat pilot? One of the advantages to having two pilots with two independent joysticks in the cockpit is the ability to transfer control in the event some part of the cockpit was damaged or sabotaged. Unfortunately, no matter which joystick was set as the master, experimentation with this technique left much to be desired in the way of redundancy.

Finally, the third technique evaluated was the use of a joystick toggle button. Upon pressing the toggle button, the virtualization firmware would transfer control to the right-side or left-side joystick. Commands from the other joystick would be ignored when the active joystick was in use. Commands from the non-active joystick may have been accepted when the active joystick was not in use. To handle such scenarios, the software uses a weight distribution table. Each joystick's input is converted into an integer value (such as 0-1023 for an analog joystick axis). That value is then multiplied against a weight in the table (based on the role of the person in control of that joystick), and then assigned an action in software (such as moving the elevators to achieve a change in pitch). In the end, we believe that this was the right conflict resolution technique for the joystick virtualization. It allowed for a redundant cockpit, by utilizing the joystick toggle to switch joystick control, while not permitting there to be confusion about who is in control, as it is the case with the other two techniques.

7 Experimental Results

Utilizing the ATMega 32u4 micro controller, we emulated a native USB joystick to the flight computer. Putting the device into slave USB mode, it is recognized as a peripheral. This peripheral is both powered by the USB cable and can send/receive data through it. The firmware on the ATMega then converts the input signals from the joystick axes, buttons, and controls into a virtual game controller device in Windows.

This interface does not require a driver. One simply connects it to the computer and it emulates a game controller, thus using the built-in game controller of Microsoft Window®.

A virtual joystick with 4 buttons, two axes (x, y), a rudder control, and a POV (point of view) hat switch is depicted in Figure 14. The joystick axes are mapped to pitch and roll control of the aircraft.

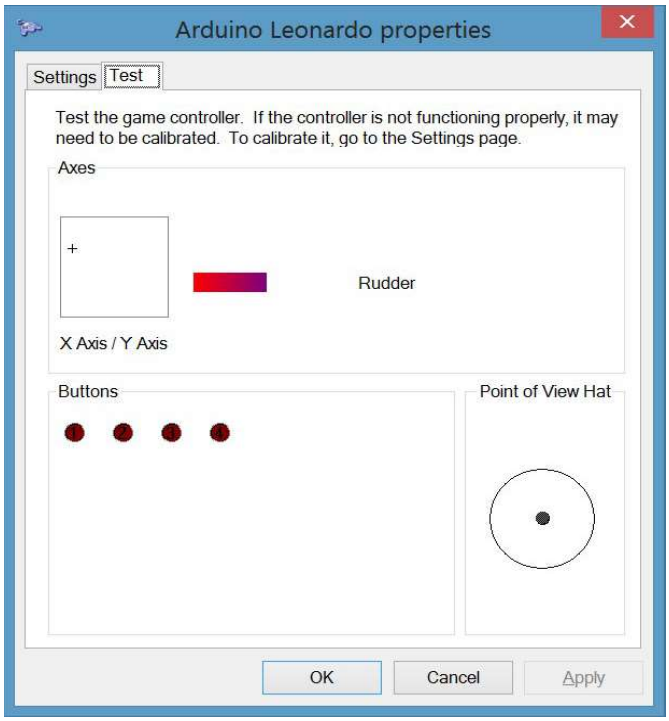

Figure 14: A joystick being recognized in windows by the system and the 3rd analog axis being mapped to the rudder

The initial experience was seamless. For those that are familiar with flight simulators, all the controls were mapped in the typical way.

The pilot controls the ailerons and elevator (pitch and roll) via the first joystick, and the throttle (engine speed) via the second joystick. The co-pilot controls the landing gear via a button on their second joystick (and has other possibilities with his or her controls that are not yet defined).

Things get interesting when multiple people, such as the pilot and the co-pilot, start to provide input at the same time. Based on each operator's role, and a weight tied to their particular joysticks, the software computes a resulting feedback value that then gets passed along to the simulation. For example, the system allows for a co-pilot to take over a flight if the pilot is incapacitated (of course, all these happen in simulation).

Better yet, this solution allows us to try new things, such as mapping the controls in a unique way. For example, we could map the co-pilot's joystick to control the rudders (pitch), while the pilot's joystick could control the ailerons and elevators (pitch and roll).

Our flight simulator was initially lacking any sort of pitch movement. This made the plane harder to control. Others have got around this problem with hardware solution: a joystick that can "twist." The twisting motion is measured and transformed into rudder movement that controls the pitch of the aircraft. However, in our flight simulator this would not be easily done because replacing the joysticks would require more space, and may not be compatible with other components. Hence, this solution allows us to achieve novel things such as repurposing a co-pilot control to maneuver the rudders, something that was not possible with our existing setup.

8 Conclusion

In this paper, we presented a method for handling input from multiple axes in a flight simulator application. As our related research showed, while joysticks and peripherals are commonplace, the systems to handle multi-joystick input, especially with different people at the helm of the controls, are not as mature.

The initial experimental results showed that our method could create a system capable of reading and responding multi-axis controls, without the need to write a driver in Windows. In the future we hope to extend this method to include the ability to emulate key presses on a keyboard.

References

[1] BAA Training Inc., "Full Flight Simulators – In 20 Years' the Number Will Double," [Online], Available: https://www.baatraining.com/full-flight-simulators-in-20-years-the-number-will-double/, [Accessed January 15, 2019].

[2] R. Bradley and D. Abelson, "Desktop Flight Simulators: Simulation Fidelity and Pilot Performance," *Behavior Research Methods, Instruments, and Computers,* 27(2):152-159, 1995.

[3] B. Gilissen, "The Last Four Minutes of Air France Flight 447," [Online], Available: http://www.spiegel.de/international/world/death-in-the-atlantic-the-last-four-minutes-of-air-france-flight-447-a-679980.html, [Accessed April 1, 2019].

[4] N. Hall, "Airplane Parts and Functions," [Online], Available: https://www.grc.nasa.gov/www/k-12/airplane/airplane.html, [Accessed August 17, 2018].

[5] C. Howard, "Airservices Australia Selects Saab Integrated Tower Automation Technology for Four Airports," [Online], Available: https://www.intelligent-aerospace.com/articles/2015/02/airservices-australia-selects-saab-integrated-tower-automation-technology-for-four-airports.html, [Accessed August 14, 2018].

[6] T. Inefuku, "FAA Requires Two People in Cockpit on U.S. Flights at all Times," [Online], Available: https://www.khon2.com/news/local-news/faa-requires-two-people-in-cockpit-on-u-s-flights-at-all-times_20180309115902834/1025802042, [Accessed July 3, 2018].

[7] J. S. Lipscomb and M. E. Pique, "Analog Input Device Physical Characteristics," *ACM SIGCHI,* 25(3):40-45, 1993.

[8] National Aeronautical and Space Administration, "The Intertropical Convergence Zone," [Online], Available: https://earthobservatory.nasa.gov/images/703/the-intertropical-convergence-zone, [Accessed April 3, 2019].

[9] F. Nobre, "A330 Flight Deck Systems and Briefing for Pilots," [Online], Available: https://www.slideshare.net/FernandoNobre1/a330-flight-deck-and-systems-briefing-for-pilots, [Accessed April 8, 2019].

[10] N. North and F. Zhang, *Encyclopedia of Atmospheric Sciences*., Academic Press, 2015.

[11] ProFlight Inc., "Pilot at Zero Altitude: A Brief History of Flight Simulators," [Online], Available: https://www.proflight.com/en/full-flight-simulatoren/historie.php, [Accessed April 17, 2018].

[12] A. Redei, E. Tumbusch, and J. Koberstein, "AVRATAR: A Virtual Environment for Puppet Animation," *Proceedings of the International Conference on Software Engineering and Data Engineering*, Las Vegas, NV, pp. 14-19, 2007.

[13] A. Sims, "Germanwings Crash," [Online], Available: https://www.independent.co.uk/news/world/europe/germanwings-crash-co-pilot-andreas-lubitzs-final-email-reveals-depression-and-fear-of-going-blind-a6915736.html, [Accessed July 7, 2018].

[14] B. Skoloff, "Equatorial Region Known for Massive Storms," [Online], Available: https://phys.org/news/2009-06-equatorial-region-massive-storms.html, [Accessed April 11, 2019].

[15] Suzo-Happ Inc., "Analog Flight Joystick with 5k Potentiometers & B5 Grip," [Online], Available: https://na.suzohapp.com/products/joysticks/95-0251-00. [Accessed June 10, 2018].

[16] Suzo-Happ Inc., "Speed Shifter," [Online], Available: https://na.suzohapp.com/products/driving_controls/50-8018-00, [Accessed June 10, 2018].

[17] Wikipedia, "Air Traffic Controller," [Online], Available: https://en.wikipedia.org/wiki/Air_traffic_controller, Accessed [July 15, 2018].

18] Wikipedia, "Flight Simulator," [Online]. Available: https://en.wikipedia.org/wiki/Flight_simulator, [Accessed April 8, 2019].

[19] J. Wise, "What Really Happened Aboard Air France 447," [Online] Available: https://www.popularmechanics.

com/flight/a3115/what-really-happened-aboard-air-france-447-6611877/, [Accessed April 11, 2019].

Alex Redei is an Assistant Professor at Central Michigan University working in the Department of Computer Science. He received his MS and PhD degrees in Computer Science and Engineering from the University of Nevada, Reno in 2013 and 2019, respectively. His interests include flight simulation, software engineering, and human-centered design. His research focuses on using flight simulators to experiment with new techniques for improving pilot training.

Sergiu Dascalu is a Professor in the Department of Computer Science and Engineering at the University of Nevada, Reno, USA, which he joined in 2002. He received in 1982 a Master's degree in Automatic Control and Computers from the Polytechnic Institute of Bucharest, Romania and, in 2001, a PhD in Computer Science from Dalhousie University, Halifax, NS, Canada. His main research interests are in software engineering, human computer interaction, and data science. He has published over 180 peer reviewed papers and has been involved in numerous projects funded by industrial companies as well as federal agencies such as NSF, NASA, and ONR. He has advised 10 PhD and over 40 Master students who graduated so far.

Frederick C. Harris, Jr. received his BS and MS degrees in Mathematics and Educational Administration from Bob Jones University, Greenville, SC, USA in 1986 and 1988, respectively. He then went on and received his MS and Ph.D. degrees in Computer Science from Clemson University, Clemson, SC, USA in 1991 and 1994 respectively. He is currently a Professor in the Department of Computer Science and Engineering and the Director of the High-Performance Computation and Visualization Lab at the University of Nevada, Reno, USA. He has published more than 200 peer-reviewed journal and conference papers along with several book chapters. His research interests are in parallel computation, computational neuroscience, computer graphics, and virtual reality. He is a Senior Member of the ACM, and a Senior Member of the International Society for Computers and their Applications (ISCA).