Development of Advanced Terrain Awareness & Warning Display System

FINAL PROJECT PROPOSAL

16.621

Spring 2003

Author: Chinwe Nyenke

Advisor: Professor R. John Hansman Partner: Shen Qu

Date: May 13, 2003

Table of Contents

List of Tables

List of Figures

EXECUTIVE SUMMARY

This 16.621 project is part of research efforts to enhance the performance of the terrain awareness and warning system (TAWS) display in aviation. The proposed design varies from the typical TAWS display in that it replaces the red-yellow-green representative terrain regions with a single red contour. It is consequently hoped that this less cluttered design will improve the pilot's readability of typical TAWS information (weather, air traffic, etc.) without diminishing his/her response to hazardous terrain.

Implementation of the design will require written software that accepts flight position information from Microsoft Flight Simulator and terrain data from a database to generate the TAWS display. The testing phase will then involve performance measurement and comparison of 10 subjects flying Microsoft Flight Simulator with a baseline design and the proposed design. Subjective information will be gathered from pre- and post-experiment surveys. The implementation phase and operating phases are expected to be completed by the 5th and 10th weeks, respectively. Data analysis will then occur until the 12th week, and a final report will be written by December 9. The anticipated budget for this project is \$0, which is astonishing when considering the scope of the proposed project.

1. INTRODUCTION

1.1 BACKGROUND AND SIGNIFICANCE OF PROJECT:

Ground Proximity Warning Systems (GPWS) have been in use since they were first introduced to the world of aviation in the 1970s. Before that period, pilots relied primarily on paper maps and out-of-window scenery for a sense of the terrain in the aircraft's immediate vicinity. Since the introduction of GPWS, not only have pilots had the addition of a third visual aid, but advancements leading to the current Enhanced GPWS (EGPWS) – also called the Terrain Awareness and Warning System (TAWS) – allow for more warning time to avoid impact with terrain. Unlike the GPWS, which is limited to look-down capability (i.e., detects terrain immediately below the aircraft), the TAWS uses a look-ahead sensor, which estimates terrain around the aircraft via information gathered from a terrain elevation database of the region.[1] Both terrain systems are still in use today, but despite the relatively new technology of TAWS, controlled flight into terrain (CFIT) remains a major cause of accidents. As of 2001, it was the leading source of fatal accidents in both commercial and general aviation.[1]

Though the TAWS display design is generally standard in commercial aircraft, it continues to evolve in the area of general aviation. TAWS, which is required by the Federal Aviation Administration (FAA) in commercial aircraft, generally consist of a multi-colored display, in which red-yellow-green terrain elevation regions represent dangerous, cautious, and non-hazardous terrain, respectively.[2] Thus, most commercial aircraft have this design as a standard addition to the rest of the cockpit display system. On the other hand, TAWS remains to be widely accepted in the world of general aviation, largely because FAA does not enforce a similar mandate for such aircraft. At the same time, the size and cost of the designs that are used in commercial aircraft are too large and expensive for general aviation aircraft. Consequently, it is generally left up to the pilot's discretion to purchase smaller terrain displays from private companies while research continues in the area of designing compact, inexpensive useful TAWS for general aviation aircraft.

This project focuses on the use of TAWS in general aviation. The intent is to assess the effects of reducing the clutter of the conventional TAWS design on the readability of the display. This is a particularly important aspect, because many of the TAWS that are available for general aviation squeeze the multi-colored design of those in commercial aircraft onto a smaller display screen. Furthermore, this project hopes to determine that removing some of this clutter will not take away from information the pilot needs to avoid imminent terrain danger. Thus, the project's significance stems from the implications its results will have for the use of TAWS in the world of general aviation.

1.2 BRIEF OVERVIEW OF PREVIOUS WORK

Much previous work has been done over the years in an attempt to improve the accuracy and clarity of TAWS. Research has ranged from adjusting the amount and type of data presented on the display screen to modifying the perspective in which the terrain is shown (i.e., 2-D, 3-D, perspective view, plan view, profile view, etc.) For instance, terrain displayed in a color-coded contour format as opposed to a complex map-like format is one way in which researchers have sought to minimize the amount of unnecessary clutter displayed on the screen (see Appendix). As a result, contour displays, such as Boeing's Green Arc TAWS, have become popular over the years. Meanwhile, research concerning the effect of terrain perspective on the pilot's recognition of danger has shown some interesting results. In particular, plan (bird's eye) and perspective (forward) views have been shown to be the most preferred amongst pilots.[3]

Though much of this recent work concerns improving TAWS in commercial and military aviation, the world of general aviation is not entirely neglected. Companies, like Honeywell, have attempted to overcome the difficulties that arise with designing an inexpensive and compact display for general aviation pilots.[1]

1.3 CONCEPTUAL OVERVIEW OF EXPERIMENT

In this experiment, the effect of screen display clutter on the recognition of and response to terrain hazards will be examined. Measurements of reaction time and the number of false alarms will be used to make conclusions about the pilot's discernment of terrain hazard. The two variables will be compared using a baseline display and a reduced clutter design.

2. HOS

HYPOTHESIS: Reducing clutter in the terrain awareness and warning systems will improve overall readability of the display without having any adverse effects on the terrain awareness (i.e., number of false alarms and reaction time).

OBJECTIVE: Use computer simulation to measure and compare pilot's performance with baseline and a reduced clutter design.

SUCCESS CRITERION: Obtain data to determine whether the performance of the reduced cluttered design is 1) better than the baseline's, 2) worse than the baseline's, or 3) indeterminate in relation to the baseline's performance due to a lack of statistical significance in the level of difference.

3. LITERATURE REVIEW

It is apparent through a review of literature that much research concerning TAWS has been done in the recent years. Such work primarily concerns the effect of certain parameters – e.g., the predictive capability of TAWS, the display design, etc. – on the discernment of terrain hazards (or, terrain awareness). Though the impact of screen clutter on terrain awareness has been mentioned, it has not been extensively studied. Thus, this review of literature concerns studies that are not directly related to this project's hypothesis but research that may provide valuable guidance to the development of this experiment in terms of methodology and background information.

Since the introduction of TAWS in the late 1990s, many researchers have questioned the effect of the advanced warning time provided by look-ahead altimeters (which, as described earlier, replaces the look-down sensors of the GPWS) on the pilot's awareness of terrain hazard. In a study thoroughly presented by de Muynck and Khatwa[4], it is concluded that "predictive terrain alerting" results in no obvious improvement in terrain situational awareness. This conclusion is confirmed by Moroze and Snow, who note that though TAWS provides more warning time, it does not give the pilot a better understanding of the emergency situation in which he is placed (i.e., situation awareness).[1] However, conflicts with reports from other resources exist indicating the complete opposite – an apparent improvement in situational awareness.[2, 5] One of the resources in conflict, Sachs' and Sperl's "Experimental Low Cost 3D-Display for General Aviation Aircraft," goes on to conclude that terrain displays with predictive capability "enhance the control performance of the pilot and reduce his workload," thus allowing the pilot to pay greater attention to his surroundings.[5] Though this discrepancy between resources exists, it does not significantly impact this project since the major focus is on the readability of the display (i.e., screen clutter) – and not predictive versus immediate alerting.

In recent years, researchers have also addressed the effect of display design on the pilot's terrain awareness – many arguing that terrain awareness can be improved or reduced depending on how the information is displayed. Glover,[2] as well as Wickens[6] and de Muynck,[4] all conclude that pilots tend to prefer (and react more readily to) such display colors as red and/or yellow because such colors instinctively indicate danger or caution. Thus, it is no surprise that majority of the TAWS in use consist of these colors (including green) to symbolize areas of hazardous terrain. However, the line is not so clear in terms of the effects of 3D versus 2D displays, and within the latter class, plan view (i.e., bird's eye view), side (profile) view, and forward (perspective) view of terrain information. Glover does present the point that "engineering limitations and economic necessities" constrain the evolution of TAWS, noting that most displays are plan view for the sake of compatibility with the rest of the flight deck plan-oriented display hardware.[2] Thus, these general display design choices reinforce certain aspects of the Red Contour design (which consists of a red line indicating danger in a plan view format) but does not imply a halt in the evolution in TAWS design. Other aspects of the paper, such as testing methods and the error challenges faced, are similar to what will occur in this experiment.

Though this literature review has revealed extensive work in TAWS research, little has been undertaken in the area of clutter reduction. While researchers have acknowledged the possible effect of clutter on readability of information (other than terrain) presented by the display, this 16.621 project will be one of a few to analyze the question in detail.

4. TECHNICAL APPROACH

4.1 GENERAL DESCRIPTION OF THE ENTIRE EXPERIMENT

The experiment will consist of ten subjects who will each complete 6 flight scenarios using the baseline design and 6 scenarios using the reduced clutter design. (Details as to the counterbalance setup are explained in the Error Analysis section.) All scenarios will contain a given flight course and zero visibility via a nighttime or foggy sky. Scenarios will also range from having zero, to single, to multiple hazards at different segments of the flight course as well as having zero, to one, to multiple pop-up distractions (i.e., other planes, random objects, etc.) to test the terrain awareness and readability of the display design, respectively. The subjects will be instructed to press a HAZARD button and change course accordingly when dangerous terrain is perceived. The subject's reaction time and the number of false alarms will then be measured as well. Each subject will also be required to fill out a pre- and post-experiment survey to obtain background and subject preference data.

4.2 DESCRIPTION OF THE APPARATUS WITH DIAGRAM(S)

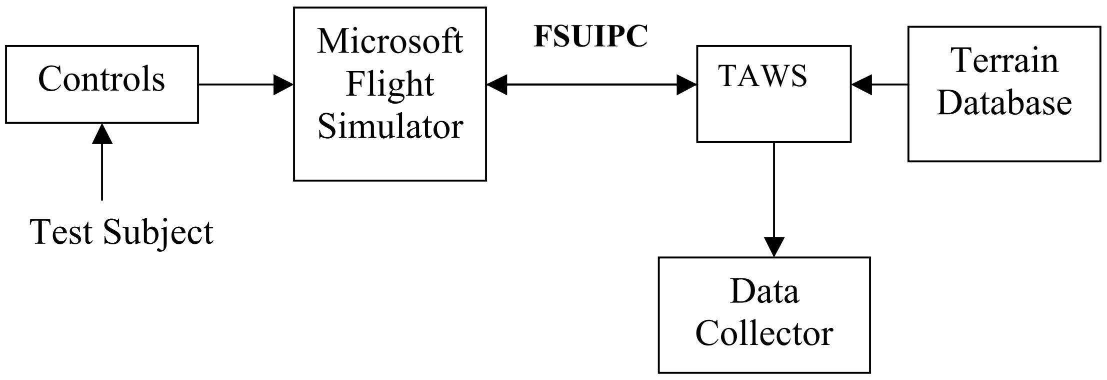
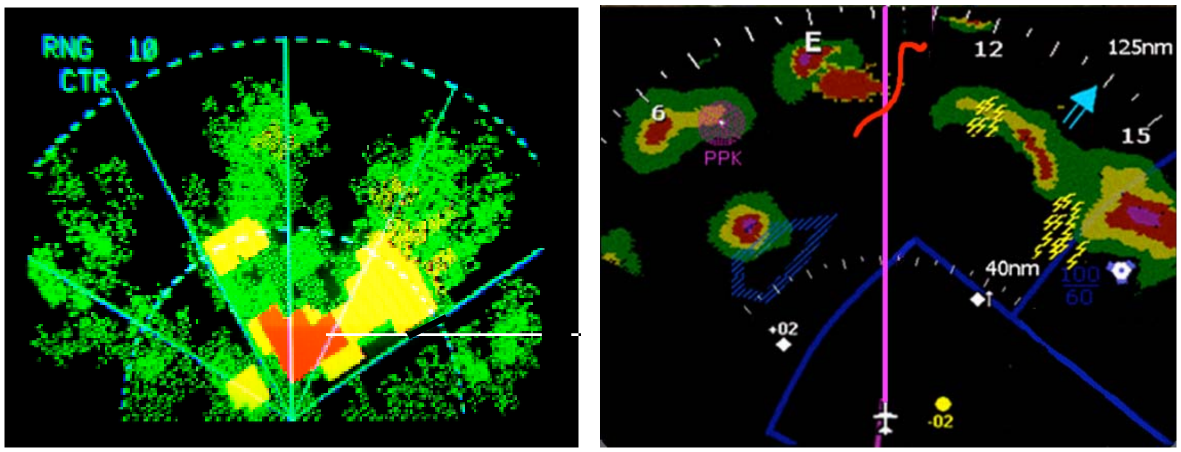


Figure 1. Schematic of Apparatus

The apparatus used in this experiment is depicted in Figure 1. Cockpit panels and out-of-window scenery (night or foggy sky) will be displayed on a computer monitor using the Microsoft Flight Simulator program. Controls, in the form of a joystick, will be attached to the computer so the subject can maneuver the simulated aircraft. The terrain awareness and warning system (either the Red Contour design or the baseline design) will be positioned next to the Microsoft Flight Simulator window on the same computer screen. Data exchange between the simulator and the TAWS will be accomplished via the Flight Simulator Unit Inter-Process Communication (FSUIPC). A terrain database of the nation's topography (possibly provided by Honeywell) will feed terrain longitude, latitude, and altitude into the TAWS for display. Finally, a data collector (in the form of a C program) will collect both aircraft and terrain position as well as velocity from the TAWS (also coded in C) to be used for calculations of reaction time and other variables.

4.3 DESCRIPTION OF TEST ARTICLES OR SUBJECTS

The test articles in this project will consist of 2 terrain awareness and warning system displays. One will be the conventional multi-colored display used by most commercial aviation pilots, and recently redesigned by Honeywell as a compact unit for use by general aviation pilots (see Figure 2a). The second will be the reduced clutter design, which replaces the red/yellow/green terrain regions with a single red contour (see Figure 2b).



(a) (b)

Figure 2. Baseline and Reduced Clutter Design Displays

The test subjects will preferably have had some actual flight experience prior to the experiment; at a minimum, the subjects should have had prior Microsoft Flight Simulator experience. These considerations will reduce the amount of time needed to train or update all subjects on the flight simulation program. Approximately 10 subjects will be used in this experiment.

4.4 DESCRIPTION OF WHAT WILL BE MEASURED

The independent variables in this experiment will be the display design and the scenarios, thus resulting in a 2D matrix. As mentioned earlier, two different TAWS during several scenarios (the exact number is to be determined later) will be tested in this project; thus, the exact dimensions of the matrix are not known at the moment.

The parameters in this experiment will consist of the 10 test subjects. (The subjects will not be considered as independent variables because this project is more concerned with the mean performance from display to display and scenario to scenario.)

The dependent variables will be the reaction time, number of false alarms, and number of pop-up distractions recognized, all of which will be recorded. Furthermore, pre- and post-experiment surveys will be used to obtain subjective data for each display design.

DISPLAY DESIGN / SCENARIO	**Red Contour**	**Baseline**
1		
2		
3		
4		
5		
6		

Figure 3. Test Matrix

4.5 DISCUSSION OF ERRORS

Certain noise may affect the results of the experiment in the following manner.

Subject variability – i.e., the fact that each subject is different and thus reacts at a different speed compared to the next subject – will have to be considered. In order to mitigate this source of error, the mean of the reaction times of all ten subjects will be recorded. Thus, though one subject may be faster than another one in responding to a terrain alert, if both subjects show improved performance between displays, the mean will demonstrate this.

Fatigue is another possible source of error and may occur in two forms. 1) Fatigue may reveal itself as the simulation progresses and the subject's reaction time begins to decrease, thus influencing the data negatively. This source of error will be mitigated via counterbalance; the subjects will be divided into 2 groups, one group using the baseline display first and the other using the reduced clutter design first. Thus, effects of fatigue will be cut out. In addition, the experiment itself will be designed so as to not take beyond an hour or so, at which point the subject may begin to get tired. There will also be a short break between the switching of TAWS displays. 2) Fatigue may also manifest itself if the subject comes into the experiment not having had sufficient rest the night before simulation day. This will be mitigated by reminding subjects ahead of time of the importance of a good night's rest.

Learning effect (i.e., getting better as the experiment progresses) is another possible source of error in this experiment, which may manifest itself in an improvement of reaction times. A counterbalance, again, will be used to mitigate this possible challenge as well as training time to ensure that everything is learned before the actual experiment. The subjects of this experiment will be divided into 2 groups in order to create a counterbalance (explained in more detail in the Error Analysis segment). One group will complete its first flight session using the Red Contour display and its second session using the baseline design. Meanwhile, the other group will complete its sessions in the reverse order: baseline design first and Red Contour display second.

Bias towards the TAWS (i.e., focusing on the display because it is known that it is being tested) is the last major source of error that this experiment may encounter. Distractions (or workloads), such as storm watch and attention to staying on course, will be set up in each scenario so that subjects do not pay unrealistic attention to the display.

5. <u>EXPERIMENTAL DESIGN</u>

5.1 DESIGN AND CONSTRUCTION METHODS FOR APPARATUS

Hardware:

The construction method for the apparatus requires that the Microsoft Flight Simulator and the TAWS run on one computer as opposed to two. This method is preferred because the two-computer setup will not allow for real-time sharing of data between the programs. Furthermore, connecting the two computers via a network crossover cable would require extra coding to run WideFS – the dynamically linked library (dll) needed for the sharing of Flight Simulator data across computer.

Software:

Two software modules will be designed to generate the baseline display and the proposed design. The modules will be written in the C language, and each will consist primarily of algorithms that do the following:

1) accept and convert data from FS2002 into usable data,
2) accept and convert data from the terrain database into usable data,
3) use the data from 1) and 2) to generate the TAWS display with OpenGL (a graphics library)

The software modules will only differ in the latter algorithm; the baseline module will generate red-yellow-green terrain elevation regions while the reduced-clutter module will depict a red contour representative of imminent terrain danger. A software flowchart follows in Figure 4 while a detailed description of the modules is provided in Appendix A.

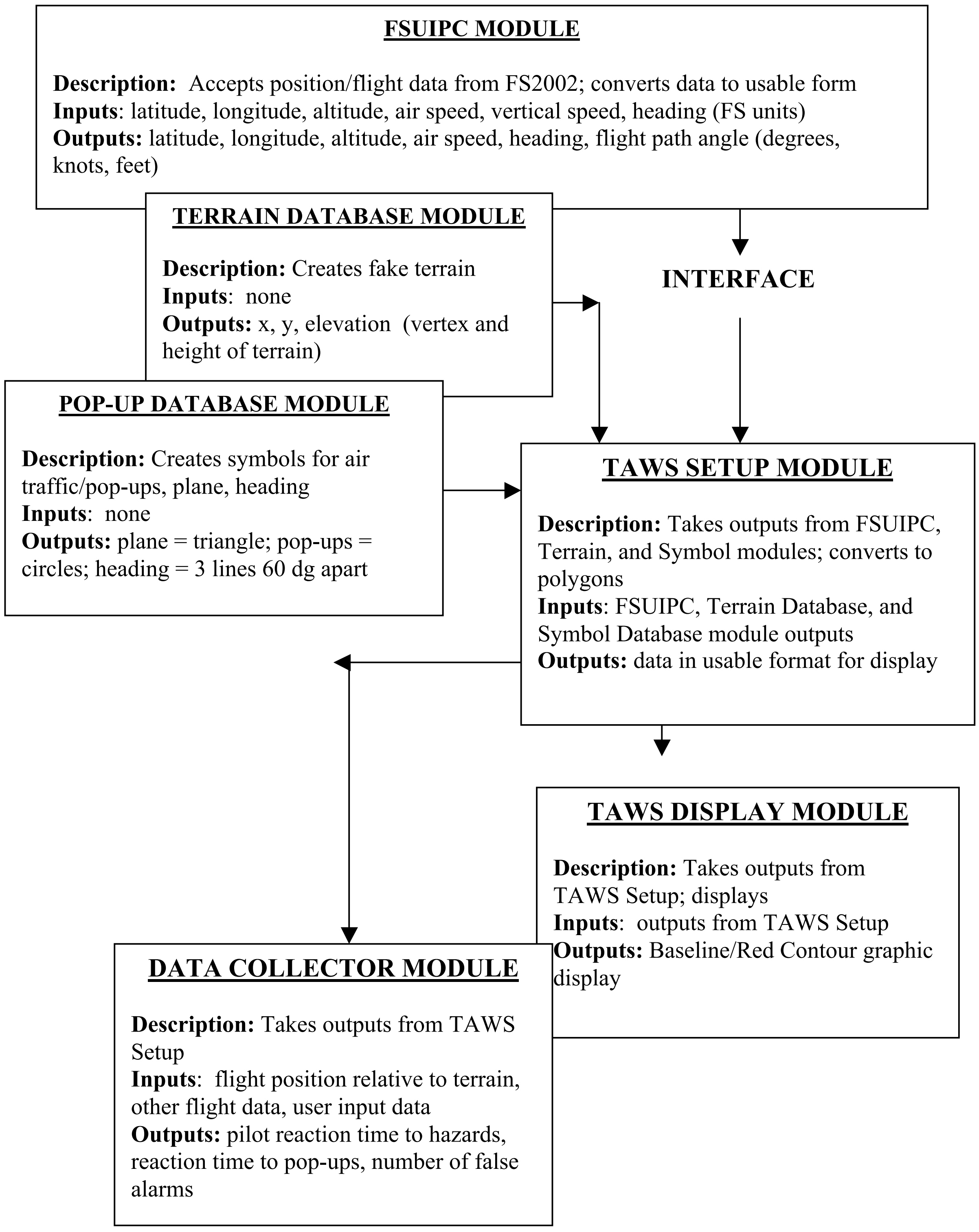

FSUIPC MODULE
Description: Accepts position/flight data from FS2002; converts data to usable form
Inputs: latitude, longitude, altitude, air speed, vertical speed, heading (FS units)
Outputs: latitude, longitude, altitude, air speed, heading, flight path angle (degrees, knots, feet)
TERRAIN DATABASE MODULE
Description: Creates fake terrain
Inputs: none
Outputs: x, y, elevation (vertex and height of terrain)
INTERFACE
POP-UP DATABASE MODULE
Description: Creates symbols for air traffic/pop-ups, plane, heading
Inputs: none
Outputs: plane = triangle; pop-ups = circles; heading = 3 lines 60 dg apart
TAWS SETUP MODULE
Description: Takes outputs from FSUIPC, Terrain, and Symbol modules; converts to polygons
Inputs: FSUIPC, Terrain Database, and Symbol Database module outputs
Outputs: data in usable format for display
TAWS DISPLAY MODULE
Description: Takes outputs from TAWS Setup; displays
Inputs: outputs from TAWS Setup
Outputs: Baseline/Red Contour graphic display
DATA COLLECTOR MODULE
Description: Takes outputs from TAWS Setup
Inputs: flight position relative to terrain, other flight data, user input data
Outputs: pilot reaction time to hazards, reaction time to pop-ups, number of false alarms

Figure 4. Software Flowchart

5.2 DESIGN OF HUMAN EXPERIMENT PROTOCOL

Obtaining Test Subjects:

Subject recruitment will be accomplished by way of email to the MIT undergraduate community and postering across campus. Students who have had either piloting or Microsoft Flight Simulator experience will be considered. Subjects must also have vision correctable to 20/20 and not be colorblind.

Simulation to Evaluate Skill:

Subjects will fly a 5-minute simulation that begins and ends at cruising altitude and speed. Completion of the planned flight course is necessary. Those who crash or stop before the 5-minute period ends will not move on to the experiment.

Surveys:

Subjects will be required to fill out a pre- and post-experiment survey. The pre-survey will gather necessary background information, including the subject's flight/simulation experience. Meanwhile, the post-experiment survey will gather subjective data on the TAWS performance, including which display is preferred. See Appendix for actual surveys.

Trial Simulation:

Subjects will then be asked to perform a 15-minute trial scenario to get adjusted to the simulation setup and TAWS display. The trial simulation will last 15 minutes and consist of zero out-of-window visibility. The primary task will be to maneuver around any dangerous terrain and press the HAZARD button anytime a hazard appears on the flight path. Subjects must also stay within the simulation environment boundaries. Crashing into terrain or failing to stay within the environment will lead to an end in the simulation. If this end occurs before the 15-minute period, they will restart the simulation until they complete a full trial run. (Maximum number of trial runs needed to adjust will be gathered from a preliminary experiment.) The simulation will start at cruising speed and altitude. One trial simulation for each TAWS must be completed.

For the information retrieval task, pop-ups will occur at a certain frequency (to be determined in preliminary experiment) and subjects must note them by pressing the corresponding key on the keyboard. For example, if aircraft A pops up and it is within 2000feet relative to your altitude, you must press the letter "A" on the keyboard immediately. This task will directly test the effect of readability of other display information on terrain awareness.

Actual Simulation:

The primary task of the subjects in the actual simulation is to fly from waypoint A to B (and then to C in a few cases). Subjects will receive X points for staying on the designated flight course, Y points for recognizing hazardous terrain, and Z points for noting relevant pop-ups. As is the case in the trial simulation, they will need to push the HAZARD button when they see dangerous terrain on the flight path. Also, they will again have to press the appropriate key on the keyboard when a pop-up within 2000feet relative to the aircraft's altitude appears.

5.3 MEASUREMENT SYSTEMS

This project will consist of two major measurement systems: the Data Collector and Pre/Post Surveys.

Data Collector Software:

This software module will generate the data collector system, which will measure the quantitative data in the experiment. It will be coded in the C language and consist of algorithms that do the following:

1) accept and convert data from the TAWS (red contour/baseline) display module
2) use data to calculate subject's reaction time to terrain hazard and pop-ups

The algorithm in 2) requires outputs of distance between the terrain and aircraft, air speed, and the times at which the user pressed the HAZARD button and keys, as well as the time at which the terrain hazard and pop-ups actually appeared. See software flowchart in Figure 4 and detailed description of the modules in Appendix A.

Surveys:
The surveys will measure qualitative data, such as level of background experience in piloting or flying Microsoft Flight Simulator and subjective preferences of the TAWS.

5.4 MAKE/BUY DESCISIONS ON EQUIPMENT

There are no major make/buy decisions on equipment in this project. Materials like software must be made while equipment like computers must be bought.

5.5 SAFETY CONCERNS

This project is required to have a Committee on the Use of Humans As Experimental Subjects (COUHES) Approval though there are no apparent safety concerns. Because the project is being conducted under the guidance of Professor Hansman, it is automatically granted a blanket approval for the use of human test subjects. See Appendix for COUHES Approval form.

6. **DATA ANALYSIS**

Paired Sample T Test:
The Paired Sample T Test will be used to analyze the quantitative data obtained in the experiment. This form of analysis will allow us to observe any differences in the reaction time to hazards and to pop-ups reducing the effect of nuisance factors. An example of the analysis is provided in Table 1. Two paired sample T tests – one for reaction time to hazards and another for reaction time to pop-ups – will be done. The variability in the mean difference between the two displays will be calculated, and the probability that there is not a difference in the data will ultimately be computed. If this probability is less than or equal to 5% in both cases, the overall pilot performance will have shown a statistically significant improvement when using the red contour display. Analyses of probabilities in further detail

will also be done to assess the hypothesis correctly. The paired sample analysis will be done with the aid of Microsoft Excel. (The number of false hazards will also require a paired sample analysis.)

Table 1. Paired Sample T Test Example

Subject	BASELINE	RED CONTOUR	BASE-RED
1	Mean rxn time 1	Mean rxn time 2	Difference in mean rxn times
2	"	"	"
3 …	"	"	"

Sample Mean Difference	X1	X2	X1-X2
Sample Standard Deviation	SD1	SD2	
N			10 (subjects)
Standard Error of Mean Difference			SE
…			

Survey Analysis:

The surveys will be analyzed for qualitative data, which will be valuable if it correlates with the reaction times and number of false hazards discerned. It may further strengthen the conclusion of the experiment, but will not weigh as heavily as the quantitative data.

7. <u>PROJECT PLANNING</u>

<u>7.1 BUDGET</u>

The following table summarizes the major expenditures for this project.

Table 2. 16.622 Budget

ITEM	AVAILABILITY	ACTUAL COST	16.62X COST
Microsoft Flight Simulator 2002	MIT	$49.99	$0
FSUIPC	Internet Download	$0	$0

Computer	MIT	$499.97	$0
Monitor	MIT	$249.97	$0
Test Subjects	MIT (tentative)	~$750	$0
Testing Facility Rental	MIT	~$900	$0
Joystick	MIT	$59.99	$0
Software Modules	N/A	~$2000	$0
Microsoft Visual C++	MIT	$112.09	$0
OpenGL	Internet Download	$0	$0
TOTAL		**$4622.01**	**$0**

7.2 DETAILED SCHEDULE (FOR 622)

The following table is the project schedule for 16.622. Padding time is already incorporated into the breakdown. The Preliminary Data stage will consist of running a trial experiment with two or three subjects.

***Table 2. 16.622* Schedule**

		SEPT.				OCT.				NOV.			DEC.
WEEK **TASK**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
IMPLEMENT PHASE	■	■	■	■	■								
Write/Revise Software	▨	▨	▨										
Test Software		▨	▨										
Take Preliminary Data				▨									
Revisions					▨								
OPERATE PHASE						■	■	■	■	■			
Collect/Reduce Data						▨	▨	▨	▨				

REPORT PHASE									■	■	■	■	■
Analyze Data									▨	▨	▨	▨	
Assess Hypothesis											▨	▨	
Report Writeup												▨	▨

7.3 FACILITIES, TECH STAFF SUPPORT AND SPACE NEEDED

The Breakout Room across from the Machine Shop in the Gelb Lab will be needed to set up and run the experiment. Paul Bauer and Fred Donovan will provide technical staff support.

8. SUMMARY

This document has presented the experimental design methods and requirements for carrying out the proposed project: development of a reduced-clutter terrain display as well as its effects on readability of other information (weather, air traffic, etc.) and terrain awareness. Feasibility of the reduced-clutter display has already been the implementation of a prototype design, which successfully reads data from FS2002 via FSUIPC and generates a red contour using fake terrain data. Pre-experimentation efforts have also gone into obtaining lab space to run simulations. The next steps are to continue implementation and begin the operation and reporting phases in 16.622 next fall.

9. LIST OF REFERENCES

1. Moroze, Michael L., Snow, Micheal P., "Causes and Remedies of Controlled Flight into Terrain in Military and Civil Aviation," http://www.hec.afrl.af.mil/publications/CFIT990429.pdf, March, 2003.
2. Glover, Howard, "Terrain Displays for Enhanced Ground Proximity Warning Systems," AIAA Paper 97-5557, Sep. 1997.
3. Kuchar, James K., Hansman Jr., R. John, "Part_Task Simulation Study of Candidate Terrain Alerting Displays," 1993.
4. de Muynck, R. J., Khatwa, R., "Flight Simulator Evaluation of the Safety Benefits of Terrain Awareness and Warning Systems," AIAA Paper 99-3965, Aug 1999.

5. Sachs, G., Sperl, R., "Experimental Low Cost 3D-Display for General Aviation Aircraft," AIAA Paper 01-37256, 2001.
6. Sachs, G., Sennes, U., "Design and Flight Test of a Ground Collision Avoidance System," AIAA Paper 99-36810, 1999.
7. Wickens, Christopher. "An Introduction to Human Factors Engineering," Displays, 1998, pp. 223-258.

10. APPENDICES

A: SOFTWARE DESIGN AND DEVELOPMENT DOCUMENT

1.0 Databases

1.1 Terrain Database

A terrain database will be artificially made up for each test scenario. The file will contain a list of rectangles represented by the x and y positions of it's 4 vertices. The positions are given in a relative scale to the simulation world normalized from 0 to 1. Each rectangle has an altitude value attached to it. Thus the file will be in matrix like format with each row containing 9 values representing the rectangle and altitude and a number of different rectangles along the columns. The rectangles are sorted along the columns by its altitude with the lower altitudes appearing first in the file. When the file is read by either the baseline or the red contour, the lower altitude values will be read in first. Thus, if a higher altitude rectangle occupies the same space, it will over ride the lower altitude one at the areas of intersection since it will be read in later. A detailed list of such rectangles will create realistic looking terrains.

The simulation world is the area that the aircraft is allowed to travel in for each simulation. This is a rectangular world with its four corners normalized to (0, 0), (0, 1), (1, 0), and (1, 1). All internal position parameters and headings will be in reference to this world.

The terrain data can also be constructed using other geometric shapes (circles, points, etc.), there are certain mathematical functions that can be used to generate random looking terrain, and we have a terrain database of the Eastern part of the US. None of these three terrain formats are used at the moment, however, they do serve as possible backups.

1.2 Pop-up Database

There will also be a pop-up database made for each scenario. Each pop-up is represented by x variables: x position, y position, letter reference, altitude value, time of appearance (time from start of simulation), and duration of appearance. The database will be a list of pop-ups each with these parameters.

2.0 Prototype

The prototype for our software is an extremely simplified version of the final software program. It contains three main pieces: C code using FSUIPC to read in MSFS parameters and convert them into standard units, a <u>very</u> simple terrain database file, and C code using Open GL to run a moving display of the terrain database in both baseline and red contour format. It is important to note that the FSUIPC and Open GL codes are not integrated. For the purpose of this prototype, an aircraft parameter file is used to provide AC parameters to the TAWS. Such a file will not exist for the final software. AC parameters will be obtained from MSFS real time.

The main purpose of this prototype is to demonstrate that we can obtain appropriate parameters from MSFS, and we can achieve baseline and red contour displays using the database and Open GL. The terrain and AC parameters provided are not realistic. And the specific colors, size, and position of the display may not be the same as the final design.

2.1 MSFS and FSUIPC

FSUIPC is a package in C with functions that allows for communication with MSFS. The code responsible for this communication and information retrieval will here after be referred to as the FSUIPC code. Below is an overview of what the prototype version of this code accomplishes.

1) checks if MSFS is running. If not, nothing happens
2) If so: gets latitude, longitude, heading, vertical speed, altitude, and air speed from MSFS
3) Converts data from MSFS units to real/usable units (degrees, knots, feet)
using conversion factors from MSFS
4) generates flight path angle
5) displays 15 example data to screen
6) ends

2.2 Aircraft(AC)Parameters File

The AC parameter files provides 6 AC parameters in the database "World" coordinate system: x position, y position, altitude, heading, flight path angle, and velocity. There are 2 AC parameter files provided (see appendix). Each file contains 10 sets of parameters. Only the y position is updated to move the terrain display area along the World. One file has constant altitude while the other has variable altitude levels.

2.3 Terrain Database

The terrain database in this case is a very simple file containing 3 rectangles at altitudes 1000, 2000, and 3000 ft respectively. In the terrain display if forms a "mountain" with 3 contour layers.

2.4 Baseline

The baseline prototype takes in terrain and AC data and displays an example baseline that update once every second for 10 updates. Red represents terrain > 500ft above the aircraft, green represents terrain with in 500ft of the aircraft, and blue represents terrain over 500ft below the aircraft. This is not the final color code used for the baseline. For the correct color-coding refer to the text. When reading in the AC data file 2 with different altitudes the color scheme changes accordingly for each update.

Main() opens a TAWS window with the appropriate size and location. It also calls init() and display(). Init() is just an initialization function. The main display function display() is responsible for displaying and updating the appropriate terrain. Find_taws_position() finds the corners and center of the TAWS in the World reference frame.

This prototype does not allow for rotation of TAWS with respect to the World.

2.6 Red Contour

The red contour prototype takes in terrain and AC data and displays an example red contour that update once every second for 10 updates. It does so by dividing the TAWS area into a 100 x 100 grid area with each grid represented by a single point. The finds the altitude of the terrain at each point and also the lowest altitude at which the aircraft will impact with terrain at each of these points. The latter is calculated using aircraft flight path angle. If the impact altitude is lower then the terrain value then the aircraft will crash if it hit that point. Finally, an algorithm is designed to search for the impact points closest to the aircraft along each heading and display it as part of the red contour.

Main() opens a TAWS window with the appropriate size and location. It also calls init() and display(). Init() is just an initialization function. The main display function display() is responsible for displaying and updating the appropriate terrain. Find_taws_position() finds the corners and center of the TAWS in the World reference frame. Unlike the baseline prototype, it also has functions clear_terrain_alt() that resets all elements in the global variable terrain_alt to 0, find_terrain_alt that finds the altitude value of at each point, and find_impact_alt() that find all of the impact altitude for each point.

This prototype does not allow for rotation of TAWS with respect to the World.

3.0 Software Architecture

There are 3 coordinate systems used through the entire code. First is the system with longitude, latitude, heading, and other values given by MSFS. Next is the reference coordinate for the simulation World with each axis normalized from 0 to 1 and the heading set at 0 degrees along the y direction. This reference frame is used for all internal calculations. Finally, there is the coordinate system used for displaying the TAWS. This is also a normalized reference frame with the TAWS center at (0, 0) with x and y both ranging from –1 to 1 at the four corners. This knowledge is needed to understand part of the discussion below.

The sections describe the changes and additions needed for the prototype before the software can be finalized. All of these changes can be easily made by person with adequate programming experience in the language C and a basic understanding of FSUIPC and Open GL.

3.1 FSUIPC

1) set aircraft at cruising altitude and airspeed at start of simulation
2) transfer data to TAWS module
3) record time and other necessary info for when user hits hazard button and keyboard
4) end when MSFS ends

3.2 Baseline and Red Contour

1) Interface with FSUIPC
2) Be able to move data from World coord. to TAWS coord.
   (Allow TAWS to rotate with respect to the World.)
3) Read pop-up display
4) Display pop-ups. (The ideas are the same as terrain read and display)
5) Display all symbols and colors according to design parameters.
   (see text.)

1.5 Interface

1) Change MSFS parameters into World coord
   (Must consider initial values and rotation factors)
2) Set the first set of MSFS variables as reference values for the rest
3) Provide parameters to TAWS code real time

1.6 Data Collector

1) Extract relevant data from TAWS code
2) Write parameters to file

4.0 Code

4.1 Terrain Database

0.4	0.2	0.6	0.2	0.6	0.8	0.4	0.8	1000.0
0.45	0.5	0.55	0.5	0.55	0.7	0.45	0.7	2000.0
0.475	0.55	0.525	0.55	0.525	0.65	0.475	0.65	3000.0

4.2 AC Parameter File 1

0.5	0.0	2000	0	0	100
0.5	0.1	2000	0	0	100
0.5	0.2	2000	0	0	100
0.5	0.3	2000	0	0	100
0.5	0.4	2000	0	0	100
0.5	0.5	2000	0	0	100
0.5	0.6	2000	0	0	100
0.5	0.7	2000	0	0	100
0.5	0.8	2000	0	0	100
0.5	0.9	2000	0	0	100

4.3 AC Parameter File 2

0.5	0.0	1000	0	0	100
0.5	0.1	2000	0	0	100
0.5	0.2	2000	0	0	100
0.5	0.3	3000	0	0	100
0.5	0.4	3000	0	0	100
0.5	0.5	2000	0	0	100
0.5	0.6	2000	0	0	100
0.5	0.7	2000	0	0	100
0.5	0.8	1000	0	0	100
0.5	0.9	2000	0	0	100

4.4 FSUIPC Code

TASK: This code uses FSUIPC to obtain Microsoft Flight Simulator data (aircraft's

```
   position, speed, vertical speed, and heading). First it verifies that the simulator is
         running, exiting if the sim is not running; then it runs a loop, outputting
   the position data, speed, heading, and flight path angle to screen  */


#include <stdio.h>
#include <math.h>
#include "IPCuser.h"

int main () {

  unsigned int counter = 0;

  /* used to hold a result code from any of the FSUIPC functions. */
  DWORD dwResult;

  /* the memory offsets for the altitude, latitude and slew mode control */
  DWORD altitudeoffset = 0x0574;
  DWORD headingoffset = 0x0580;  /* heading */
  DWORD latitudeoffset = 0x0560;
  DWORD longitudeoffset = 0x0568;   /* longitude */
  DWORD speedoffset = 0x02B8;  /* true air speed */
  DWORD v_speedoffset = 0x02C8;     /* vertical speed */



  /* the size of altitude, latitude, and slew mode control in FS2002. */
  DWORD altitudesize = 4;
  DWORD headingsize = 4;
  DWORD latitudesize = 8;
  DWORD longitudesize = 8;
  DWORD speedsize = 4;
  DWORD v_speedsize = 4;


  /* used to hold the altitude and latitude in real (not FS) units. */
  double altitude = 0;
  double heading = 0;
  double latitude = 0;
  double longitude = 0;
  double speed = 0;
  double v_speed = 0;
  double path_angle;


  /* used to hold the altitude, latitude, and slew mode status in
     FS2002's format. */
  INT32 fs_altitude = 0;
  INT32 fs_heading = 0;
  INT64 fs_latitude = 0;
  INT64 fs_longitude = 0;
```

```
INT32 fs_speed = 0;
INT32 fs_vspeed = 0;

/* conversion factors */
double alt_conv = 1/3.28084;  /* multiply by this -- feet */
INT64 head_conv = 11930465;  /*divide by this -- degrees */
INT64 lat_conv = 477302101697422;   /*divide by this -- degrees (+N, -S)*/
INT64 long_conv = 51240955760300000;   /*divide by this -- degrees (+E, -W)*/
double speed_conv = 1/128; /* multiply by this -- knots */
double vspeed_conv = 1.944/256; /* multiply by this -- knots */

/* open a link to flight sim. (FSUIPC Demo)*/
if (!FSUIPC_Open(SIM_ANY, &dwResult))
  {
   printf("Cannot start the FSUIPC client.\n");
   printf("Flight Simulator is probably not running.\n");
   return -1;
  }

/* read the current altitude and latitude, storing the
   values in fs_altitudevalue and fs_latitudevalue */
if (!FSUIPC_Read(altitudeoffset, altitudesize, &fs_altitude, &dwResult))
  {
   printf("FSUIPC_Read (altitude) failed.\n");
   FSUIPC_Close();
   return -1;
  }

if (!FSUIPC_Read(headingoffset, headingsize, &fs_heading, &dwResult))
  {
   printf("FSUIPC_Read (heading) failed.\n");
   FSUIPC_Close();
   return -1;
  }

if (!FSUIPC_Read(latitudeoffset, latitudesize, &fs_latitude, &dwResult))
  {
   printf("FSUIPC_Read (latitude) failed.\n");
   FSUIPC_Close();
   return -1;
  }

if (!FSUIPC_Read(longitudeoffset, longitudesize, &fs_longitude, &dwResult))
  {
   printf("FSUIPC_Read (longitude) failed.\n");
   FSUIPC_Close();
   return -1;
  }

if (!FSUIPC_Read(speedoffset, speedsize, &fs_speed, &dwResult))
  {
      printf("FSUIPC_Read (speed) failed.\n");
      FSUIPC_Close();
      return -1;
    }

  if (!FSUIPC_Read(v_speedoffset, v_speedsize, &fs_vspeed, &dwResult))
    {
      printf("FSUIPC_Read (vertical speed) failed.\n");
      FSUIPC_Close();
      return -1;
    }


  /* process all pending requests. (FSUIPC Demo)*/
  if (!FSUIPC_Process(&dwResult))
    {
      printf("FSUIPC_Process (data read) failed.\n");
      FSUIPC_Close();
      return -1;
    }

  /* LOOP to obtain data.  Each time convert the data from fs units into
     real ones, storing the result in the "<>value" variables.
     .....*/


  printf("ANGLE ALTITUDE  HEADING  LATITUDE  LONGITUDE  SPEED\n");

  for (counter = 0; counter < 15; counter++)
          {
      /* convert from FS2002 units into real ones */
      altitude = (double) fs_altitude / alt_conv;     /*feet*/
            heading = (double) fs_heading / (double) head_conv + 349;    /*degrees*/
      latitude = (double) fs_latitude / (double) lat_conv;   /*degrees*/
            longitude = (double) fs_longitude / (double) long_conv;   /*degrees*/
            speed = (double) fs_speed *1/128;   /*knots*/
            v_speed = (double) fs_vspeed *vspeed_conv;   /*knots*/
            path_angle = asin(v_speed/speed)*180/3.1416;   /*calculation of flight path angle  */

      /* print altitude and latitude values to screen. */
            printf("%.1f    %.1f      %.1f   %.2f    %.2f    %.2f            %.1f\n", path_angle, altitude,
heading, latitude, longitude, speed, v_speed);

    }

  /* close link to FS2002 */
  FSUIPC_Close();

  return 1;
}
```

4.5 Baseline Code

```
#include <GL/glut.h>
#include <time.h>
#include <stdio.h>

void init(void);
void display(void);
void read_data (void);
void find_taws_position (float ac_x, float ac_y);

//global variables

float ACParamList[10][6] = {0};
float TerrainDataList[3][9] = {0};

float world_AC_x_Pos = 0.5;
float world_AC_y_Pos = 0.0;

const float x_scale = 2.0;
const float y_scale = 2.0;

float taws_pos[5][2]={0.0};
const float taws_AC_x_pos = 0.5;
const float taws_AC_y_pos = 0.0;

int main(int argc, char **argv)
{
        read_data();
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
        glutInitWindowSize (400, 400);
        glutInitWindowPosition (600, 250);
0
        glutCreateWindow("TAWS");

        init();
        glutDisplayFunc(display);


        glutMainLoop();
        return 0;
}

void init(void)
{
 glClearColor(0.0, 0.0, 0.0, 0.0);

}
```

```
void display(void)

{
        float start_time=0.0;
        float rel_alt = 0.0;
        int i = 0;
        int j = 0;

        for (i=0;i<10;i++)
        {
                find_taws_position (ACParamList[i][0], ACParamList[i][1]);

                glClear(GL_COLOR_BUFFER_BIT) ;
                start_time=clock();
                //glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

                for(j=0;j<3;j++)
                {
                        rel_alt = TerrainDataList[j][8]-ACParamList[i][2];

                        glColor3f(1.0, 1.0, 1.0);

                        if(rel_alt>-500.0 && rel_alt<=500.0)
                                glColor3f(0.0, 0.0, 1.0);

                        if (rel_alt>500.0)
                                glColor3f(1.0, 0.0, .0);

                        if (rel_alt <= -500.0)
                                glColor3f(0.0, 1.0, 0.0);

                        glBegin(GL_QUADS);
                                glVertex2f(2*x_scale*(TerrainDataList[j][0]-taws_pos[4][0]),
2*y_scale*(TerrainDataList[j][1]-taws_pos[4][1]));
                                glVertex2f(2*x_scale*(TerrainDataList[j][2]-taws_pos[4][0]),
2*y_scale*(TerrainDataList[j][3]-taws_pos[4][1]));
                                glVertex2f(2*x_scale*(TerrainDataList[j][4]-taws_pos[4][0]),
2*y_scale*(TerrainDataList[j][5]-taws_pos[4][1]));
                                glVertex2f(2*x_scale*(TerrainDataList[j][6]-taws_pos[4][0]),
2*y_scale*(TerrainDataList[j][7]-taws_pos[4][1]));


                        glEnd();
                }

                glColor3f(1.0, 1.0, 1.0);

                glBegin(GL_TRIANGLES);
                        glVertex2f(-0.04,-0.99);
                        glVertex2f(0.04, -0.99);
```

```
				glVertex2f(0.0, -0.90);
			glEnd();

			glFlush();
			glutSwapBuffers() ;
			while(((clock() - start_time)/CLOCKS_PER_SEC)<1.0 ) {}
		}
}

void read_data (void)
{
		FILE *ACParamFile; //format x(rel) y(rel) alt(ft) heading(deg) FPAngle(deg) vel(ft/s)
		FILE *TerrainDataFile; //format vert1	vert2	vert3	vert4	alt
												//each vert has x y vals
		int i=0;

		ACParamFile = fopen("AC_param2.dat","r");
		TerrainDataFile = fopen("terrain.dat","r");

		while(!feof (TerrainDataFile))
		{
			fscanf(TerrainDataFile, "%f", &TerrainDataList[i][0]);
			fscanf(TerrainDataFile, "%f", &TerrainDataList[i][1]);
			fscanf(TerrainDataFile, "%f", &TerrainDataList[i][2]);
			fscanf(TerrainDataFile, "%f", &TerrainDataList[i][3]);
			fscanf(TerrainDataFile, "%f", &TerrainDataList[i][4]);
			fscanf(TerrainDataFile, "%f", &TerrainDataList[i][5]);
			fscanf(TerrainDataFile, "%f", &TerrainDataList[i][6]);
			fscanf(TerrainDataFile, "%f", &TerrainDataList[i][7]);
			fscanf(TerrainDataFile, "%f", &TerrainDataList[i][8]);

			i++;
		}

		i=0;

		while(!feof (ACParamFile))
		{
			fscanf(ACParamFile, "%f", &ACParamList[i][0]);
			fscanf(ACParamFile, "%f", &ACParamList[i][1]);
			fscanf(ACParamFile, "%f", &ACParamList[i][2]);
			fscanf(ACParamFile, "%f", &ACParamList[i][3]);
			fscanf(ACParamFile, "%f", &ACParamList[i][4]);
			fscanf(ACParamFile, "%f", &ACParamList[i][5]);

			i++;
		}

		fclose(ACParamFile);
		fclose(TerrainDataFile);
```

```
        return;

}

void find_taws_position (float ac_x, float ac_y)
{

        //taws_pos[1] to [4] are the 4 corners of the taws, taws_pos[5] is the center of taws
        taws_pos[0][0] = ac_x - 1/(2*x_scale);
        taws_pos[0][1] = ac_y;
        taws_pos[1][0] = ac_x + 1/(2*x_scale);
        taws_pos[1][1] = ac_y;
        taws_pos[2][0] = ac_x - 1/(2*x_scale);
        taws_pos[2][1] = ac_y + 1/y_scale;
        taws_pos[3][0] = ac_x + 1/(2*x_scale);
        taws_pos[3][1] = ac_y + 1/y_scale;
        taws_pos[4][0] = ac_x;
        taws_pos[4][1] = ac_y+1/(y_scale);

        return;

}
```

4.6 Red Contour Code

```
#include <GL/glut.h>
#include <time.h>
#include <stdio.h>
#include <math.h>

void init(void);
void display(void);
void read_data (void);
void find_taws_position (float ac_x, float ac_y);
void clear_terrain_alt(void);
void find_impact_alt(float alt, float FPangle);
void find_terrain_alt(void);

//global variables

const float x_scale = 2.0;
const float y_scale = 2.0;

const float x_dist = 32808.3989; // ft, for TAWS
const float y_dist = 32808.3989; // ft, for TAWS

const float taws_AC_x_pos = 0.0;
const float taws_AC_y_pos = -1.0;

float taws_pos[5][2]={0.0};
float world_AC_x_Pos = 0.5;
float world_AC_y_Pos = 0.0;

float ACParamList[10][6] = {0};
float TerrainDataList[3][9] = {0};

float impact_alt[100][100] = {0};
float terrain_alt [100][100]= {0};

int main(int argc, char **argv)
{
        read_data();
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
        glutInitWindowSize (400, 400);
        glutInitWindowPosition (600, 250);

        glutCreateWindow("Red Contour");

        init();
        glutDisplayFunc(display);


        glutMainLoop();
        return 0;
}

void init(void)
{
  glClearColor(0.0, 0.0, 0.0, 0.0);

}

void display(void)

{
        float start_time=0.0;
        float rel_alt = 0.0;
        int i=0;
        int j=0;
        int k=0;

        for (i=0;i<10;i++)
        {
                find_taws_position (ACParamList[i][0], ACParamList[i][1]);
                clear_terrain_alt();
                find_terrain_alt();
                find_impact_alt(ACParamList[i][2], ACParamList[i][4]);

                glClear(GL_COLOR_BUFFER_BIT) ;
                start_time=clock();0
		//glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);

		glColor3f(1.0, 0.0, 0.0);

		glPointSize(5.0);
		glBegin(GL_POINTS);
		for(j=0;j<100;j++)
		{
			for(k=0; k<100; k++)
			{
				if(terrain_alt[j][k]>=impact_alt[j][k])
				{
					glVertex2f((j/50.0)-1.0, (k/50.0)-1.0);
					break;
				}
			}
		}
		glEnd();

		glColor3f(1.0, 1.0, 1.0);

		glBegin(GL_TRIANGLES);
			glVertex2f(-0.04,-0.99);
			glVertex2f(0.04, -0.99);
			glVertex2f(0.0, -0.90);
		glEnd();

		glFlush();
		glutSwapBuffers() ;
		while(((clock() - start_time)/CLOCKS_PER_SEC)<1.0 ) {}
	}
}

void read_data (void)
{
	FILE *ACParamFile; //format x(rel) y(rel) alt(ft) heading(deg) FPAngle(deg) vel(ft/s)
	FILE *TerrainDataFile; //format vert1	vert2	vert3	vert4	alt
								//each vert has x y vals
	int i=0;

	ACParamFile = fopen("AC_param2.dat","r");
	TerrainDataFile = fopen("terrain.dat","r");

	i=0;

	while(!feof (TerrainDataFile))
	{
		fscanf(TerrainDataFile, "%f", &TerrainDataList[i][0]);
		fscanf(TerrainDataFile, "%f", &TerrainDataList[i][1]);
		fscanf(TerrainDataFile, "%f", &TerrainDataList[i][2]);
		fscanf(TerrainDataFile, "%f", &TerrainDataList[i][3]);
```

```
                fscanf(TerrainDataFile, "%f", &TerrainDataList[i][4]);
                fscanf(TerrainDataFile, "%f", &TerrainDataList[i][5]);
                fscanf(TerrainDataFile, "%f", &TerrainDataList[i][6]);
                fscanf(TerrainDataFile, "%f", &TerrainDataList[i][7]);
                fscanf(TerrainDataFile, "%f", &TerrainDataList[i][8]);

                i++;
        }

        i=0;

        while(!feof (ACParamFile))
        {
                fscanf(ACParamFile, "%f", &ACParamList[i][0]);
                fscanf(ACParamFile, "%f", &ACParamList[i][1]);
                fscanf(ACParamFile, "%f", &ACParamList[i][2]);
                fscanf(ACParamFile, "%f", &ACParamList[i][3]);
                fscanf(ACParamFile, "%f", &ACParamList[i][4]);
                fscanf(ACParamFile, "%f", &ACParamList[i][5]);

                i++;
        }
        fclose(ACParamFile);
        fclose(TerrainDataFile);

        return;

}

void find_taws_position (float ac_x, float ac_y)
{

        //taws_pos[1] to [4] are the 4 corners of the taws, taws_pos[5] is the center of taws
        taws_pos[0][0] = ac_x - 1/(2*x_scale);
        taws_pos[0][1] = ac_y;
        taws_pos[1][0] = ac_x + 1/(2*x_scale);
        taws_pos[1][1] = ac_y;
        taws_pos[2][0] = ac_x - 1/(2*x_scale);
        taws_pos[2][1] = ac_y + 1/y_scale;
        taws_pos[3][0] = ac_x + 1/(2*x_scale);
        taws_pos[3][1] = ac_y + 1/y_scale;
        taws_pos[4][0] = ac_x;
        taws_pos[4][1] = ac_y+1/(y_scale);

        return;

}

void clear_terrain_alt(void)
{
        int i=0;
```

```
	int j=0;

	for(i=0;i<100;i++)
	{
		for(j=0;j<100;j++)
		{
			terrain_alt[i][j] = 0.0;
		}
	}
	return;
}

void find_impact_alt(float alt, float FPangle)
{
	float dist = 0.0;
	int i=0;
	int j=0;

	for(i=0;i<100;i++)
	{
		for(j=0;j<100;j++)
		{
			dist = sqrt(pow((x_dist*(i-50)/100), 2)+pow((y_dist*i/100), 2));
			impact_alt[i][j] = alt- sin(FPangle*3.14/180)*dist;
		}
	}
	return;
}

void find_terrain_alt(void)
{

	float xMin=0.;
	float xMax=0.;
	float yMin=0.;
	float yMax=0.;

	int i=0;
	int j=0;
	int k=0;

	for(k=0;k<3;k++)
	{
		xMin = x_scale*(TerrainDataList[k][0]-taws_pos[0][0]);
		xMax = x_scale*(TerrainDataList[k][4]-taws_pos[0][0]);
		yMin = y_scale*(TerrainDataList[k][1]-taws_pos[0][1]);
		yMax = y_scale*(TerrainDataList[k][5]-taws_pos[0][1]);


		if(xMin<0.)
			xMin=0.;
		if(xMax>1)
			xMax=1.0;
		if(yMin<0)
			yMin=0.0;
		if(yMax>1)
			yMax=1.0;


		for(i= (int) (xMin*100); i < (100 * xMax) && xMax>0;i++)
		{
			for (j= (int) (yMin*100); j<(100*yMax) && yMax>0; j++)
			{
				terrain_alt[i][j] = TerrainDataList[k][8];
			}
		}
	}

	return;

}
```

B: DETAILED PARTS LIST

Software:

- TAWS software
- Data Collector software
- Terrain Database software
- Pop-Up Database software
- FSUIPC software
- FSUIPC dynamically linked library
- Microsoft Visual C++
- OpenGL
- Microsoft Excel
- Microsoft Flight Simulator 2002

Hardware:

- Joystick
- Computers
- Chair

C: COUHES APPROVAL FORM

D: TEST SCENARIOS

Subject Skill Test:

Subjects will fly a 5-minute simulation that begins and ends at cruising altitude and speed. Completion of the planned flight course is necessary. Those who crash or stop before the 5-minute period ends will not move on to the experiment.

Learning Scenario:

This is a test scenario designed to help subjects learn our simulation set up. The simulation world is a terrain-rich area several times larger then the TAWS display area. Clear boundaries will be given. Subject will crash if he/she tries to fly outside these boundaries. The learning time is tentatively 15 min. This time may change pending our evaluation of the difficulty of the simulation setup.

The simulation will start with the aircraft at cruising speed and altitude. Subject will learn by flying around the simulation world, avoiding terrain with the TAWS display. If the subject crashes before 15min, the scenario will be restarted. Time between restarts will not count towards the 15min.

Data will be collected during this scenario as it would any other scenario (data collection methods and parameters described in the body of Version III). Data from these scenarios will be analyzed differently then all other scenarios. (Analysis method described in the text.)

Below is a list of actual test scenarios. All scenarios start with the aircraft at cruising speed and altitude. Scenarios 2-6 will end a few seconds after the subject recognizes a terrain hazard or crashes. Scenario 1 will end when the subject is close to point B. Pop-ups will appear once every second in all scenarios. Each scenario will be programmed in two alternative forms for the two TAWS (same idea, different surrounding factors). The number of crashes in each case will be noted. The planned flight course routing will be updated on the display screen after each waypoint (if applicable).

Scenario 1: No Hazard

Flight from point A to point B. Flat terrain; no hazard appears. Simulation ends when the subject is within 10 seconds of reaching point B or crashes. Scenario 1 addresses the effect of the display on the subject's recognition of pop-ups.

Scenario 2: Real Hazard

Flight from point A to point B. Terrain hazard appears midway through course. Simulation ends after subject recognizes hazard or crashes into terrain. Scenario 2 addresses the effect of the display on the subject's recognition of pop-ups. It also addresses the effect of the pop-up discerning task on the subject's terrain awareness.

Scenario 3: False Hazard

Flight from point A to point B to point C. Terrain hazard appears after the turn at point B such that if the course continues in a straight line from A to B and beyond, the aircraft runs into the terrain. This hazard is false because it does not appear on the planned course but several kilometers off of it. Simulation ends after subject recognizes hazard or crashes into terrain. It also addresses the effect of the display on the subject's recognition of pop-ups as well as the effect of the pop-up discerning task on the subject's terrain awareness. This is also used to balance out Scenario 4 so that the subject does not expect a real hazard every time he/she makes a turn. (*Note: in this scenario, since the hazard is false and on the line past A to B, there is no hazard present on the course from B to C.)*

Scenario 4: Turn into Hazard

Flight from point A to point B. Terrain hazard appears several seconds after the ATC (Air Traffic Controller) asks the subject to maneuver West. Simulation ends after subject recognizes hazard or crashes into terrain. Scenario 4 addresses a possible design issue in the reduced clutter display. While the terrain hazard is readily visible on the baseline display, it does not appear onto the screen of the test display until the turn is in progress. This scenario also addresses the effect of the display on the subject's recognition of pop-ups and the effect of the pop-up discerning task on the subject's terrain awareness.

E: TEST PROTOCOL

<u>Obtaining Test Subjects</u>

(via email and poster)

Have you ever piloted a plane …
… either in reality OR on Microsoft Flight Simulator?

If so, you are just the subject we are seeking for a Flight Simulator experiment. If you have some actual piloting experience OR are familiar with Microsoft Flight Simulator, we need YOU to reply to this email/contact us at __________. We only require ____ minutes/hours of your time, and all you need to do is play several Flight Simulator games we have designed!!!! The dates we will run the experiment are as follows:

Looking forward to hearing from you!!!

<u>Subject Qualification:</u> in addition to having pilot/FlightSim experience, subject MUST NOT be colorblind and must have vision correctable to 20/20.

<u>Pre-testing reminder email</u>

Hi <name>,

This is a reminder you have a Microsoft Flightsim simulation appointment with us tomorrow at <time>.

We look forward to seeing you. Thank you for your time.

Sincerely

Shen Qu and Chinwe Nyenke

<u>Simulation Procedures</u>

The same procedure will apply to all subjects. Only one subject will be in the simulation room at any given time. Subject will be asked if s/he is ready before the beginning of each scenario.

1) Give subject 5-minute skills test; if the subject passes, go to step 2.
2) Ask subjects to complete pre-simulation survey

3) Seat subjects in front of simulator
4) Start simulator with baseline TAWS (or red contour)
5) Bring up learning scenario
6) Read simulation instructions section 1
7) Start learning scenario when subjects are ready
8) If subject crashes within 15 min, restart the scenario
9) End scenario at 15 min
10) Bring up Scenario 1
11) Read simulation instructions section 2.
12) Start scenario
13) Repeat steps 10 through 12 for scenarios 2-6. No step 11
14) 10-minute break
15) Bring up the other TAWS display
16) Repeat steps 5-12. Step 6 reads simulation instruction section 3. No step 11
17) Ask subject to complete post-simulation survey
18) Read simulation instruction section 4

Subject Screening

Pilots will be asked to present pilot license.

MSFS users will be asked to fly a 5-minute simulation that begins and ends at cruising altitude and speed. They must be able to complete the planned flight course without crashing.

Pre-Simulation Survey

Name:
Age:
Gender:

1. Have you ever piloted an aircraft before? Yes/No

2. If you answered *No* to the question 1, skip to question 3.

    a) Do you have a piloting license? Yes/No

    b) How long has it been since you've gotten your license (years):

    c) What type of aircraft have you flown:

    d) How many piloting hours do you have logged:

    e) When was the last time that you've flown (please give approximate month/day/year):

    f) How would you describe your skill as a pilot (1 = poor, 7 = excellent):

1	2	3	4	5	6	7

3. How many hours of experience do you have on Microsoft Flightsim (MSFS):

4. How often do you use MSFS (everyday, once a week/month/year):

5. When was the last time that you used MSFS:

6. How would you describe your skill in using MSFS (1 = poor, 7 = excellent):

1	2	3	4	5	6	7

7. How many hours of sleep did you get last night:

8. How would you describe your level of alertness today:

1	2	3	4	5	6	7

9. Do you have any comments relating to the experiment (general conditions, experiences, concerns, etc.)?

Test Instructions (to be read to the subject)

General Welcome:

Hi <name>. Thank you for coming. This experiment will run no longer than X hours. You will be asked to complete several flight simulations using Microsoft Flight Simulator. We will first have you undergo a skill test before we go on any further. *(If subject recruited as a pilot, request pilot license at this point.)* Any questions at the moment?

Section 1 – Trial Simulation:

The trial simulation will last 15 minutes and consist of zero out-of-window visibility. Your primary task is to maneuver around any dangerous terrain and press this button *(point to HAZARD button)* anytime a hazard appears on your flight path. You must also stay within the simulation environment boundaries. Crashing into terrain or failing to stay within the environment will lead to an end in the simulation. If this end occurs before the 15-minute period, you will restart the simulation until you complete a full trial run. The simulation will start at cruising speed and altitude.

In order to navigate around the environment, you will have to use the FlightSim cockpit panels and this terrain display as a guide.
- *(Pull out terrain display diagram.)*
- *Description of display, HAZARD button, and pop-ups:*

If using BASELINE display first:

- The red-yellow-green regions represent different terrain elevations – red being 2000+ feet above the plane, yellow being within 2000 feet above the aircraft and 500 feet below the plane, and green being 500-2000 feet below the plane. Thus, you must attempt to avoid the red regions and be cautious of the yellow areas on the display. Press this HAZARD button whenever you notice a red region on your flight path.
- The blue line on the display represents your flight course, which you must do your best to follow in the actual simulation. This will not be important in the trial run.
- Pop-ups such as these *(point to display diagram)* may occur at random intervals, and you must note them by pressing the corresponding key *(point to keyboard).* For example, if aircraft A pops up and it is within 2000feet relative to your altitude, you must press the letter "A" on the keyboard immediately.
- So, in this trial simulation, you will need to: discern hazardous terrain by pushing the HAZARD button, note pop-ups by pushing the corresponding key, and do your best to stay within the environment boundaries. Any questions?

If using RED CONTOUR display first:

- A red contour will appear on the screen only if there is imminent terrain danger within 60 degrees of your flight path. Thus, you must attempt to avoid the red contour or you will crash.

- The blue line on the display represents your flight course, which you must do your best to follow in the actual simulation. This will not be important in the trial run.
- Pop-ups such as these *(point to display diagram)* may occur at random intervals, and you must note them by pressing the corresponding key *(point to keyboard).* For example, if aircraft A pops up and it is within 2000feet relative to your altitude, you must press the letter "A" on the keyboard immediately.
- So, in this trial simulation, you will need to: discern hazardous terrain by pushing the HAZARD button, note pop-ups by pushing the corresponding key, and do your best to stay within the environment boundaries. Any questions?

Section 2 – Scenario Description:

Before each scenario, tell/reinforce to the subject: Your primary task is to fly from waypoint A to B (and then to C – *depending on the scenario*). You will receive X points for staying on the designated flight course, Y points for recognizing hazardous terrain, and Z points for noting relevant pop-ups. As was the case in the trial simulation, you will need to push the HAZARD button when you see dangerous terrain on your flight path. Also, you will again have to press the appropriate key on the keyboard when a pop-up within 2000feet relative to your altitude appears. The values of Y and Z depend on how fast you react to the hazards and pop-ups. Any questions?

Section 3 – Trial Simulation**:**

You will now perform another 15-minute trial simulation using a different terrain display. The same rules apply: press this HAZARD button when necessary, stay within the simulation environment boundaries, and press the appropriate key on the keyboard corresponding to any pop-ups. Once again, crashing into terrain or failing to stay within the environment will lead to an end in the simulation, and if this end occurs before the 15-minute period, you will restart the simulation.

Here is the terrain display you will have to use to navigate in the environment.
- *(Pull out terrain display diagram.)*
- *Description of display, HAZARD situation:*

If using BASELINE display now:
- The red-yellow-green regions represent different terrain elevations – red being within 2000+ feet above the plane, yellow being within 2000 feet above the aircraft and 500 feet below the plane, and green being 500-2000 feet below the plane. Thus, you must attempt to avoid the red regions and be cautious of the yellow areas on the display. Press this HAZARD button whenever you notice a red region on your flight path.
- The blue line on the display represents your flight course, which you must do your best to follow in the actual simulation. This will not be important in the trial run.
- Any questions?

If using RED CONTOUR display now:

- A red contour will appear on the screen only if there is imminent terrain danger within 60 degrees of your flight path. Thus, you must attempt to avoid the red contour or you will crash.
- The blue line on the display represents your flight course, which you must do your best to follow in the actual simulation. This will not be important in the trial run.
- Any questions?

Section 4 – Post-Experiment Survey:

Now we ask that you fill out one more short survey concerning your experience during the flight simulation. Please be honest and give plenty of feedback. Feel free to ask us any questions.
After subject finishes survey: Thank you for your participation. We ask that you do not discuss the nature and specifics of these tests to others as there are other subjects who have not yet preformed these tasks.

Post-Simulation Survey

Name:

On scale of 1to 5 (1 being the *red-contour*, 5 being the *multi-colored design*), rate the display that provided you with BETTER ABILITY TO:

1. View waypoints:

1	2	3	4	5
red-contour		same		multi-colored

2. Recognize and respond to pop-ups:

1	2	3	4	5
red-contour		same		multi-colored

3. Follow the planned flight course:

1	2	3	4	5
red-contour		same		multi-colored

4. Discern hazardous terrain:

1	2	3	4	5
red-contour		same		multi-colored

5. Read information other than terrain data:

1	2	3	4	5
red-contour		same		multi-colored

Other

6. Which display do you prefer? Why?

7. Any comments:

Thank you for your participation!!!