

AMS PRO: A GRAPHICAL AERIAL MANEUVER
SIMULATION PROGRAM
FOR
PRELIMINARY FLIGHT TRAINING

A Thesis
in TCC 402

Presented to

The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science Computer Science

by

Terence Glenn Taylor

March 25, 1997

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in Humanities Courses.

Approved _____ (Technical Advisor)
James C. McDaniel

Approved _____ (TCC Advisor)
Patricia C. Click

PREFACE

I have had an interest in flight and aviation since my early years in grade school. This led me to enroll in the Air Force Reserve Officer Training Corps (AFROTC) program at the University of Virginia. I have received only 9 hours of flight training through ROTC incentive programs, but I am a 1997 Air Force pilot candidate. Upon graduation and commissioning, I will travel to Texas to attend the first phase of pilot training, the Air Force Flight Screening Program (FSP). After completing FSP, I will attend Undergraduate Pilot Training (UPT) for more specialized training in a particular aircraft.

Though my career goal is to be a pilot in the Air Force, my academic studies focus on computer science. Being a fourth-year computer science major, I have come to realize the importance of software usability. My interests lie in computer graphics, graphical user interface design, human-computer interaction, and usability engineering.

Besides interests in computer graphics and flight, I also have a sincere desire to teach and train others. In the summer of 1996, I was one of 30 cadets selected to travel to Lackland Air Force Base, San Antonio, Texas to participate in the training of nearly 500 Air Force ROTC cadets. I have carried this desire to lead and teach from an early age and now I have yet another opportunity to provide a tool from which many people might learn. This project has allowed me to combine my interest in flight, my training in computers, and my desire to teach others.

The program I have developed allows users to view aerobatics through a medium which, until this point, has only been available to those with access to a simulator or to those affiliated with the military. I see this project as a way of providing a tool by which anyone can learn more about flight. It is hoped that this will spark interest in a field which I believe is truly fascinating; the field of aviation.

I would like to give special thanks to the User Interface Group of the University of Virginia for their continuing support and assistance throughout the course of this project.

Table of Contents

<u>Section</u>	<u>Page Number</u>
List of Figures and Tables	6
Glossary of Terms	7
Abstract	9
I. Introduction	10
A. Statement of Project Purpose	10
B. Problem Description	10
C. Project Description	12
D. Review of Relevant Literature	13
E. Overview of Technical Report	14
II. Project Progression	15
A. The Alice Project.....	15
B. Design Process	16
III. Description of AMS Pro	19
IV. Testing of AMS Pro	22
A. Test Environments	22

B. User Feedback	23
V. Conclusion	24
A. Summary of Findings	24
B. Interpretation of Results	25
C. Project Recommendations	25
D. What was Learned	27
Annotated Bibliography	29
Appendices	32
A. Changes Made to the Interface	33
B. Computer Code for AMS Pro Version 1.4	35

List of Figures and Tables

Figures

<u>Figure</u>	<u>Page Number</u>
Figure 1. Alice Scripting Window	15
Figure 2. Alice Camera Window	16
Figure 3. AMS Pro Control Panel	19
Figure 4. T-3 Trainer Aircraft	20
Figure 5. F-15 Eagle Aircraft	20
Figure 6. F-117 Nighthawk Stealth Fighter	20
Figure 7. User Testing Task List	22

Tables

<u>Table</u>	<u>Page Number</u>
Table 1. Profile of Users Who Tested AMS Pro	23

GLOSSARY OF TERMS

aerobatics - Type of flying where the aircraft performs tricks in the air. Used for shows and training purposes.

aileron roll - An aileron roll is a 360 degree roll done either to the right or to the left. The aircraft rolls placing added pressure on one of the ailerons (the rear portion of the wing).

barrel roll - A barrel roll is a roll in which the nose of the aircraft rotates about a horizontal axis external to the plane. A barrel roll is executed with the aircraft stick whereas the aileron roll is not.

chandelle - A chandelle is a 180 degree steep climbing turn with a maximum gain of altitude.

cloverleaf - The cloverleaf is composed of four identical maneuvers, each begun in a vertical plane rotated 90 degrees from the preceding one. One enters the cloverleaf by performing a loop and executing a 90 degree roll in either direction. This patten continues until the original flight path is resumed.

cuban eight - Maneuver in which the aircraft traces a figure eight in the vertical plane by executing two loops with aileron rolls on the downward sides of each.

immelman - An immelman is a half loop followed by a half roll, resulting in a 180 degree turn and an increase in altitude.

interface - The portion of a computer program that allows the user to access or run the program. An interface provides items such as labels or buttons to make the program clear and easy to use.

landing - Deceleration of the aircraft while in flight to touchdown safely at ground level.

lazy eight - A maneuver in which the aircraft traces a figure eight in a horizontal plane. The lazy eight is performed by turning 180 degrees to the right or left, and repeating this turn in the opposite direction.

loop - A loop is a 360 degree turn in the aircraft's vertical plane executed by pulling the nose of the aircraft up.

pitch rotation - Rotation of the aircraft about the axis through its wings. Note that this maneuver cannot actually be performed by the aircraft.

robustness - In the sense of computer software, the durability or strength of a program. A robust program is one that is hard to place in an irregular state or one that has very few bugs.

roll rotation - Rotation of the aircraft about the axis through its nose and tail.

spin - A spin is a stall that results in excessive plane rotation. The aircraft spirals downward in a corkscrew path. One wing of the aircraft has more lift than the other and gravity brings the aircraft down.

split S - A 180 degree turn executed by rolling the aircraft until it is completely inverted. A half-loop is then executed to resume level flight. A large amount of altitude is lost if recovery from inverted flight is attempted in this manner.

stall - Situation in which the angle of attack exceeds the critical value in relation to oncoming airflow. Airflow separates across the upper surface of the wings and the plane begins to lose its ability to stay in the air.

takeoff - Acceleration of the aircraft at ground level to become airborne.

yaw rotation - Rotation about the aircraft's vertical axis. Note that this maneuver cannot actually be performed by the aircraft.

ABSTRACT

Programmers, scientists, and military personnel alike have carried out research as to the effectiveness of computer simulations in flight training. Researchers have explored this topic since computer graphics became popular in the 1970's. Through continued development, computer simulations have proven to be effective teaching devices. Now, with ongoing improvements to computer graphics tools, it is much easier to make simulators effective.

Since 1990, the Computer Science Department at the University of Virginia has developed a three-dimensional, interactive programming environment. The environment, called Alice, allows users to write programs that manipulate objects in a 3D scene. The Alice tool is well suited for projects that require a large amount of 3D object manipulation. For this reason, Alice was chosen as the programming environment for this thesis project: A Graphical Aerial Maneuver Simulation Program for Preliminary Flight Training (AMS Pro).

The rationale behind designing such an application was to provide a learning tool which demonstrated aerial maneuvers and aerobatics. The program is primarily intended for high school and college students with an interest in flight, but was designed for use by anyone with interests in this field. The design goal throughout development was to make the program simple and easy to use. This was accomplished by incorporating user-testing into the design process. User feedback was analyzed and changes were made to AMS Pro accordingly. Based upon general user opinion, AMS Pro served its purpose well.

I. INTRODUCTION

A. Statement of Project Purpose

The purpose of this project has been to develop a computer program that demonstrates aerial maneuvers and basic concepts of flight using detailed 3D computer graphics. Design emphasis was placed on making the application simple, interesting, and informative in an effort to provide an adequate introduction to aeronautics. It is hoped that such a program will be beneficial to high school and college students, as well as other individuals with interests in this field.

B. Problem Description

The United States is currently in a dominating role with regard to air power. However, in order to sustain this role, proper flight training of our pilots is paramount. There is currently no effective way of providing introductory training during the early stages of the Air Force pilot education program. The only information available to high school and college students entering Air Force ROTC is through literature. Computerized flight simulators are sold as video games, but these programs lack educational value. Students would benefit if they could sit down to a computer program that demonstrated techniques such as takeoff and landing, or showed different aerial maneuvers performed by modern aircraft. Such a tool is needed to introduce basic concepts of flight to those with an interest in this field.

I made the decision to become a pilot at an early age. This led me to enroll in the Air Force ROTC program upon entering college. Now, in my final year at the University of Virginia, I look back at the minimal amount of training I received towards my career as a pilot. While the AFROTC program provided a great deal of valuable training on being an officer, very little of the curriculum dealt with concepts of flight. This is not a fault of the AFROTC program, however. The curriculum goal is to build future leaders for the Air Force, not to train possible pilot candidates. Therefore, many of the AFROTC cadets who wish to fly are uneducated in flight concepts throughout the early stages of the program.

A pilot candidate who recently completed the Air Force Flight Screening Program (FSP) stated, “I definitely think that early in the [AFROTC] program there should be more exposure to flight fundamentals for those who are interested in becoming pilots. I really don’t know of any way a cadet can do this unless they do it on their own. I took ground school at my own expense. It would definitely make the transition to FSP easier” (Lieutenant Mary Casdin).

A retired Air Force pilot and current aerospace engineering professor at the University of Virginia states, “I do not remember any introduction to flight concepts during my ROTC enrollment at UVA, except for the basic flight training that ROTC provided then...The first actual flight simulator I saw was the one at pilot school after graduation. Such a program as the one you are developing would serve to motivate students to want to become a pilot, perhaps to join ROTC to do so, and give students a better idea of what flight maneuvers are before actually entering a flight training program” (Professor James McDaniel).

If students are exposed to concepts of flight earlier, the education process will yield more effective results. An application such as the Aerial Maneuver Simulation Program (AMS Pro) will provide an interactive learning environment early in the education process.

C. Project Description

Design of AMS Pro consisted of two broad areas: the internal operation of the program, and its user-interface. While design of a program such as this may be challenging, all research and development is useless if no one can access the information that it has to offer. For this reason, the design goal in this endeavor was to make the program easy to use and understand. This goal was accomplished by following an iterative design approach consisting of user testing, analysis of user feedback, and redesign of the program interface. User feedback and performance determined success and effectiveness of the final product.

The application was built using Alice, computer-aided design software recently developed by the University of Virginia Computer Science Department. ALICE allows users to develop powerful programs and demonstrations through an interactive, real-time, 3-D graphical programming environment. Use of this recently developed software made development of AMS Pro much simpler.

D. Review of Relevant Literature

In recent decades, a significant amount of research has been done regarding the use of computer-based instruction in flight training. Interest in this field has grown largely because experiments proved that interactive computer-aided learning devices contribute greatly to the education process. Not only are there educational benefits, but there are economic ones as well. In the early 70's, the Department of Defense adopted the use of flight simulators as a means of cutting back on fuel consumption of aircraft. The guiding philosophy was that simulation taught basic concepts of flight just as effectively as actual flight time. The Defense Department sought funding from Congress and submitted its Flight Simulation Report in 1974. Minutes from the Congressional hearing regarding this matter stated, "Simulation can provide better, safer training in areas such as...orientation...flying" (106). The result of these hearings was the adoption of computer simulation as a flight training tool.

Further research as to the educational value of flight simulators. Simulators have been proven to be much more effective when introduced earlier in the learning process. Naval and Air Force agencies conducted tests in the mid-80's and results showed that exposure to flight simulators prior to fighter-pilot training increased performance of students who had little or no previous training (Amdor, 24). In 1988, research was done on the feasibility of computer-based instruction (CBI) and simulators for those attending fighter lead-in training (LIT). "Former LIT graduates said their initial simulator training was invaluable" (Amdor, 6).

The Air Force Human Resources Laboratory has recommended that computer simulation be used in flight training (Amdor, 23). According to Ronald G. Hughes, computer simulation is a very powerful tool that, if utilized correctly, could have huge impacts on education:

Developments in learning theory over the past 50 years have led to principles of behavior which have been shown in innumerable applied settings to be valuable in analyzing and modifying human behavior. When applied to flying training using simulators, these principles suggest that a significant contribution could be made toward improving the way in which Instructor Pilots teach new students via more effective use of simulator functions. When the simulator is conceptualized as merely an inferior copy of an aircraft, its potential as a teaching device is likely to be overlooked. It is believed that a behavioral analysis of the optimal conditions of learning could make a major contribution to both the design and use of current and future flight simulators. (13)

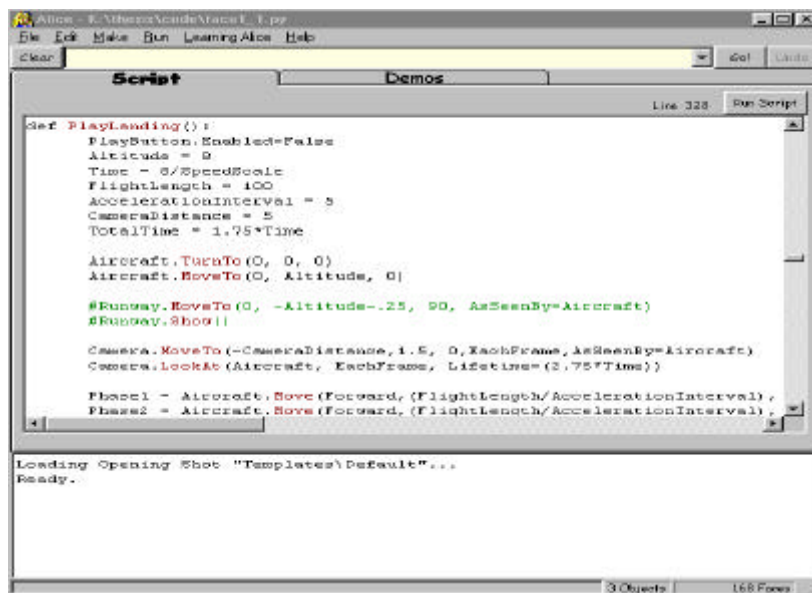
E. Overview of Technical Report

This report explains the logical progression of the thesis project and gives information on the tools and techniques used to build AMS Pro. The document also provides a detailed description of program features and functionality, along with an analysis of the testing procedures and results. The report concludes with a discussion of the effectiveness of the application and recommendations for future modifications.

II. PROJECT PROGRESSION

A. The Alice Project

Alice is a recently developed three-dimensional, interactive, graphical programming environment. It allows users to create programs called “scripts,” which control the movement of objects in the 3D environment. The Alice programming environment consists of two windows, the scripting window shown in Figure 1, and the camera window shown in Figure 2. All programming is carried out through the scripting window while the camera window shows the movement of the 3D objects. Because Alice was still in its developmental stages throughout the course of this project, some difficulties were encountered in converting code from one version of Alice to the next. However, because Alice was so well suited for three-dimensional programming projects such as this, the benefits of using the software greatly outweighed any problems that arose.



```

def PlayLanding():
    PlayButton.Enabled=False
    Altitude = 0
    Time = 3/SpeedScale
    FlightLength = 100
    AccelerationInterval = 5
    CameraDistance = 5
    TotalTime = 1.75*Time

    Aircraft.TouchTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

    #Runway.MoveTo(0, -Altitude-.25, 00, AsSeenBy=Aircraft)
    #Runway.Show()

    Camera.MoveTo(-CameraDistance, 1.5, 0, EachFrame, AsSeenBy=Aircraft)
    Camera.LookAt(Aircraft, EachFrame, Lifetime=(2.75*Time))

    Phase1 = Aircraft.Move(Foreward, (FlightLength/AccelerationInterval),
    Phase2 = Aircraft.Move(Foreward, (FlightLength/AccelerationInterval),
  
```

Loading Opening Shot "Templates\Default"...
Ready.

3 Objects | 168 Fows

Figure 1. Alice Scripting Window

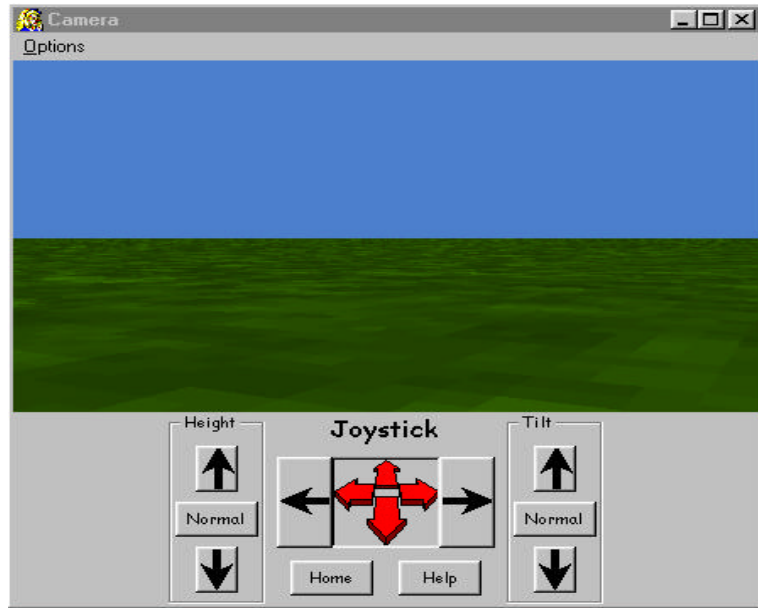


Figure 2. Alice Camera Window

B. Design Process

Preliminary Design

Before developing AMS Pro, it was important to determine exactly what the goals of the project were. The goal was to present several examples of aerial maneuvers in a way that was clear and understandable. The preliminary design provided direction in development of the final version of the application.

The first step in the preliminary design was to decide which maneuvers and demonstrations to show. It was important to choose maneuvers that provided a comprehensive overview of aerobatics and presented the users with fundamental aspects of flight. Seven demonstrations and ten aerobatic maneuvers were chosen based on their effectiveness in teaching and the amount of time given to complete the thesis project.

After determining *what* information to show, attention was directed towards the interface in an effort to determine *how* to present the information. With usability as the overall design goal, a layout of the initial interface was constructed. This interface included a window to view the aircraft as it performed each maneuver and a control panel which resembled that of a cassette or CD player. The control panel included buttons to play, pause, and stop execution of the demonstrations. This interface was chosen because of its simplicity and familiarity to users.

Application Coding

The longest and most labor-intensive portion of design was programming each of the aerial maneuvers. Using both the T-3 Aircraft Training Manual (Department of the Air Force) and a video entitled “Aerobatics II: Flying the Maneuvers” (EAA Aviation Foundation) as guidelines, each maneuver was programmed and tested separately. Different camera angles and aircraft speeds were tested in an effort to provide the most effective learning experience. Each of the individual programs was then compiled into a larger program that which the interface could access.

Interface Design

The original interface was designed using Visual Basic, a program that allows users to easily develop Windows-based application interfaces. Throughout the course development, however, the Alice designers added capabilities that allowed users to develop interfaces for their programs through Alice. Although the Alice interface development capabilities weren't as extensive as those of Visual Basic, the Alice software was used to build the interface for AMS Pro. This decision was made because Alice provided more reliable communication between the interface and the simulations. The Visual Basic interface was then reprogrammed using the Alice software. After its initial development, changes were made to the interface based upon user feedback from testing sessions. This process was repeated until a final version of the interface was submitted as part of the thesis project.

III. DESCRIPTION OF AMS PRO

In designing AMS Pro, a goal was to provide all necessary functionality to make the program effective without overloading the interface with unnecessary features. Figure 3 shows the control panel.

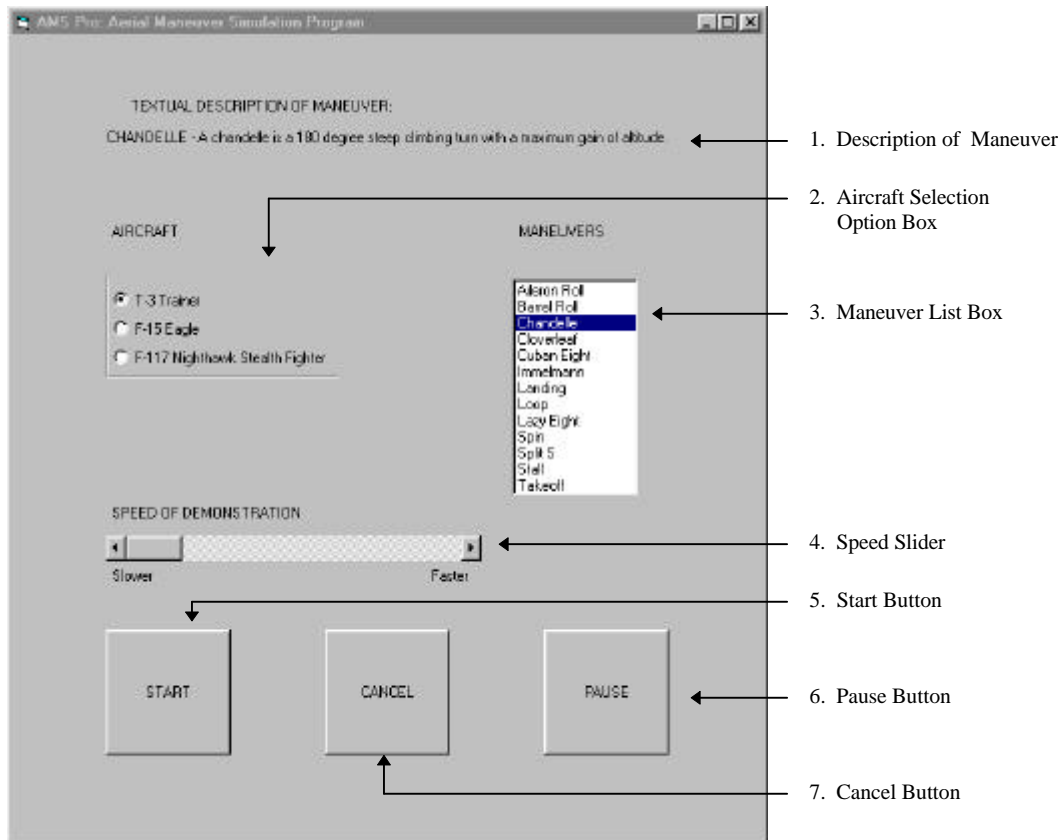


Figure 3. AMS Pro Control Panel

1. *Textual Description of Maneuver* - Each maneuver includes a brief description of what actions the aircraft is taking. This allows the user to better conceive the maneuvers before and during execution.

2. *Aircraft Selection Option Box* - Allows the user to choose the type of aircraft that will execute the maneuvers. The user may choose between a T-3 Trainer Aircraft (Figure 4), an F-15 Eagle (Figure 5), or an F-117 Nighthawk Stealth Fighter (Figure 6).



Figure 4. T-3 Trainer Aircraft



Figure 5. F-15 Eagle Aircraft

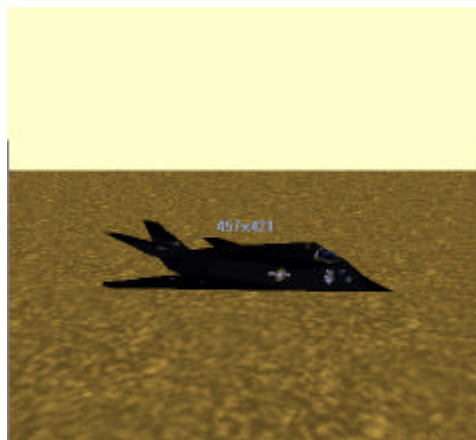


Figure 6. F-117 Nighthawk Stealth Fighter

3. *Maneuver List Box* - This box lists all maneuvers that the user may choose.
Any of the following maneuvers may be chosen: aileron roll, barrel roll, chandelle, cloverleaf, cuban eight, immelmann, landing, lazy eight, loop, spin, split S, stall, and takeoff.
4. *Speed Slider* - This control allows the user to alter the speed at which the demonstrations execute. Speed may be set before execution begins, but may not be altered during execution.
5. *Start Button* - When pressed, the Start Button begins execution of the maneuver that the user has chosen.
6. *Pause Button* - Pressing the Pause Button while a maneuver is executing results in temporary halting of the maneuver. Execution resumes when the user presses the Pause Button again.
7. *Cancel Button* - This button stops the demonstration or maneuver that is currently running. Execution may not resume until the user presses the Start Button.

IV. TESTING OF AMS PRO

A. Test Environments

In an effort to ensure usability of the application, user testing of the AMS Pro interface was conducted. Test sessions were conducted in 20-minute intervals and each user tested the application on an individual basis. After explaining the goals of the thesis project and the reasons for performing user testing, users were presented a list of five tasks to perform through the aerial maneuver program. The task list was developed to pinpoint areas of the interface which may need improvement. It required users to perform tasks that incorporated each of the interface controls and functions, thereby providing feedback as to the usability of the interface. Users were encouraged to continuously give feedback throughout the test session, and at the end of the session, to name three areas of both positive and negative aspects of the program. The task list is shown in Figure 7.

"A Graphical Aerial Maneuver Simulation Program For Preliminary Flight Training."

Task List

1. Perform a Loop with the F-15 Eagle
2. Perform an Immelmann with the F-117 Stealth Fighter.
3. Perform a takeoff with any aircraft at the fastest demonstration speed.
4. Perform a landing at the slowest demonstration speed with any aircraft BUT stop the demonstration before it is completed.
5. Perform a cloverleaf with any aircraft and pause the maneuver sometime during execution.

*Now feel free to test or 'play with' any portions of the program you choose.

Figure 7. User Testing Task List

B. User Feedback

Thirty-four users tested AMS Pro, 79% of whom are members of the University of Virginia Air Force ROTC Detachment. These users were selected because they were representative of individuals with an interest in flight and cover the age range and training levels of those for whom the application was developed. Users had a wide range of backgrounds, from technical to non-technical majors. A breakdown of pertinent areas is shown in Table 1.

<i>Category</i>	<i>Percent of Total Users</i>
<i>First Year</i>	9%
<i>Second Year</i>	32%
<i>Third Year</i>	35%
<i>Fourth Year</i>	24%
<i>Engineering School</i>	50%
<i>Non-Technical Major</i>	50%
<i>Flight Experience</i>	56%
<i>AFROTC Cadet</i>	79%

Table 1. Profile of users who tested AMS Pro.

In general, user feedback was very positive. Users commented on the detail of graphical images and particularly liked the textual description that is provided. The description was helpful to many who had no prior knowledge of what particular maneuvers were. Users also commented that there is a wide range of maneuvers to choose from. One area of concern, however, was that the aircraft were difficult to see at times due to background colors or confusing camera angles. This issue was addressed and fewer problems were encountered with each new version of AMS Pro. Finally, a large percentage of users commented that the interface was simple and easy to use. They noted its educational value as well.

V. CONCLUSION

A. Summary of Findings

Most of the areas of improvement in AMS Pro were in the realism of some of the maneuvers presented. At times, the aircraft executed maneuvers at speeds greater than actuality or with turn radii which would cause pilots to endure a severe number of G's. Users noted that camera angles were sometimes confusing and that the ground should remain in the picture more often to provide a point of reference. Despite these recommendations for improvement, users who tested AMS Pro found it very simple, informative and easy to use. Although users are to remain anonymous, a list of quotes which were stated during test sessions follows.

Quotes Regarding Usability of AMS Pro Interface

- "...simple to operate."
- "...easy to follow."
- "...simple and easy to use."
- "...fun to play with."
- "...very clear."
- "...not at all hard to understand."
- "...straightforward interface...not fancy."
- "I've never seen a demonstration with a view from outside the aircraft."
- "This would be really good for learning the maneuvers."
- "This is very educational. It explains each maneuver and then shows it."
- "This shows the aerobatics very well. It something I would use before pilot training."
- "I've never done anything like this before and it's easy for me to get the hang of it."

B. Interpretation of Results

AMS Pro was created to “provide a computer program that demonstrates aerial maneuvers and basic concepts of flight using detailed 3D computer graphics.” The final product was to be “simple, interesting, and informative” (Statement of Project Purpose, 10). In comparing initial goals of the project and results from user-testing, AMS Pro serves its purpose well. There were numerous comments from users as to the simplicity and clarity of the program interface. Users also noted the program’s educational value and informativeness. Because of these qualities of AMS Pro, users liked to use the program. This may be the most important factor in a teaching device. When students and users have a positive experience with teaching tools, the rate of learning is increased. For these reasons, I feel that the development of AMS Pro addresses its intended purpose extremely well.

C. Project Recommendations

During the iterative process of testing each version of AMS Pro, many recommendations were made as improvements to the program. Most of these improvements were implemented in later versions of AMS Pro, but due to time constraints and limitation of resources, some areas were not addressed. These recommended improvements, along with others that I have noted, are shown in the following list.

Recommended Improvements to AMS Pro

- Allow users to choose from several different views of aircraft.
- Provide more planes to choose from.
- Give descriptions of aircraft.
- Include a scrollbar indicating the elapsed time of the demonstration.
- Enhance graphics of F-15 and F-117 aircraft.
- Provide more maneuvers from which to choose.
- Add sound to give more realistic affect of maneuvers.
- Provide more scenery to enhance realism and give more frames of reference.
- Show speed numerically.
- Include a dialog box which explains each part of the maneuver as it is executing.
- Place icons on buttons to give quicker interpretation to users.
- Modify text size to make more important areas stand out when needed.
- Disable parts of interface that may not be used a particular times.
- Highlight textual descriptions to make more evident.
- Include fast-forward and rewind buttons.
- Provide split screen so users can compare maneuvers.
- Allow users to vary speed of aircraft during execution.
- Include options to create an aerobatic sequence or flight patter from the choices of maneuvers.
- Take into account screen colors for users who may be Red-Green Colorblind.
- Make this application available over the World Wide Web.
- Conduct tests as to how effective this device is as a teaching tool.

I envision AMS Pro as a program which would be accessible students across the world via the World Wide Web. It would be very beneficial for students with interests in flight to find a web page which they could access at any time to address questions regarding flight and aerobatics. Currently, AMS Pro may only be run on machines which have Alice Software installed. This limits access to AMS Pro to machines with Windows 95. Unfortunately, students cannot use AMS Pro via the Web, but it is possible to download the software to their local machines. With this feature implemented, a detailed users' manual and set of instructions should be included.

D. What was Learned

Through completion of the thesis project, I gained invaluable experience in technical, design, and personal areas. Before undergoing this project, I had very little prior experience using Windows 95, Alice, and the Python programming language. I am now proficient with each of these software packages. In the area of design issues, I was able to experience firsthand what end-to-end design is actually like. Throughout the course of my undergraduate career as a Computer Science Major, I have never had the opportunity to work on a project from its conception stages through final testing. Through this project I was able to choose a topic of interest, justify the need for such an application, design the application, and conduct testing as to its effectiveness. This “whole project” concept allowed me to better understand the relations and dependencies of different portions of the design process.

I have also come to realize the significance and impact of 3D interactive graphics on users of such programs. Users were extremely responsive to AMS Pro and none left with a negative impression of the program. On the other hand, designing such programs with 3D graphics is very time consuming. Proper attention must be paid to simulations in order to provide the most realistic impression possible.

Although I have had experience with user-testing in the past, the importance of this stage in design was again stressed. Each user who tested AMS Pro provided invaluable feedback as to what worked well and what areas of the program needed improvement. As a designer, it is sometimes difficult to point out

areas of a program that need improvement both because of designer biases, and inherent assumptions that a typical user would not have.

In conclusion, I have gained invaluable experience in flying aerial maneuvers. This will prove to be beneficial as I prepare for the first phase of the Air Force Pilot Training Program. Design of AMS Pro has been an extremely positive experience and I have been fortunate to have had an opportunity to design and produce such an application.

Annotated Bibliography

“Air Force Weapons.” Airman Magazine Sept. 1994: 38-9.

A description of current weapons used by the U.S. Air Force. Provides important graphical information concerning aircraft.

ALICE. Computer software. Three-Dimensional Programming Environment, 1995. Windows 95.

A real-time, interactive, 3-D programming environment. This language provides the backbone of the building of the thesis project.

Amdor, S.L. Computer-Based Instruction/Simulator Program for Fighter Lead-In Training: Feasibility Research. Brooks Air Force Base, TX: Air Force Human Resources Laboratory, Air Force Systems Command, 1988.

Study conducted by the United States Air Force regarding computer based instruction. Provides important historical notes.

Anderson, John D. Jr. Introduction to Flight. New York: McGraw-Hill Book Company, 1989.

An introduction to the fundamentals of aviation and flight. Contains information that will be presented in the thesis project.

Chant, Christopher. Modern Combat Aircraft. Secaucus, NJ: Chartwell Books, Inc. 1984.

Pictorial descriptions of Air Force aircraft. Provides basis for graphical portions of the project.

Department of the Air Force. Enhanced Flight Screening Program: Airmanship. Headquarters 19th Air Force, Randolph AFB, TX: 1995.

T-3 aircraft flight instruction manual. This manual is currently used in the Air Force Flight Screening Program.

Gray, Thomas H. Advanced Flight Simulator: Utilization in A-10 Conversion and Air-to-Surface attack Training. Brooks Air Force Base, TX: Air Force Human Resources Laboratory, Air Force Systems Command, 1981.

A study conducted by the United States Air Force. Provides information on the use and effectiveness of flight simulators.

Hartson, H. Rex, and Debora Hix. Developing User Interfaces. New York: John Wiley & Sons, Inc., 1993.

A design approach to developing user interfaces. Provides information on how to make an effective interface.

Hughes, Ronald G. Advanced Training Features, Bridging the Gap Between In-Flight and Simulator-Based Models of Flight Training. Brooks Air Force Base, TX: Air Force Human Resources Laboratory, Air Force Systems Command, 1979.

A study conducted by the United States Air Force. Explores methods of improving computer based training.

McDaniel, James C. Jr., Aerospace Engineering Professor, University of Virginia, Charlottesville, VA. Personal interview. 16 Sept. 1996.

Technical advisor for the project. Provided technical information and input throughout project development.

Nelson, Robert C. Flight Stability and Automatic Control. New York: McGraw-Hill Book Company, 1989.

An introduction to the basic fundamentals of flight. Contains information to be presented in the flight application program.

Nielson, Jakob. Usability Engineering. London: Academic Press, Inc., 1993.

A textbook on the designing usable applications. This is important to the interface of the application.

Paul Harvey Audio/Video Center. "Aerobatics II: Flying the Maneuvers." VHS. EAA Aviation Foundation, Oshkosh, WI. 1996.

Video showing execution of many of the aerial maneuvers presented in this project. Extremely helpful because each maneuver is explained in great detail.

Pausch, Randy. "The Alice Project." VHS. User Interface Group, University of Virginia. March 1996.

Video providing an introduction to ALICE. Shows capabilities of the new software through examples.

Pausch, Randy, Computer Science Professor, University of Virginia, Charlottesville, VA. Personal interview. 23 Jul. 1996.

Computer Science professor whose interests are in usability of systems and computer graphics. Major contributor to development of the ALICE system.

United States. Congress. Senate. Committee on Armed Services. Subcommittee on Research and Development. Flight Simulators: Hearing before the Subcommittee on Research and Development of the Committee on Armed Services, United States Senate, Ninety-fourth Congress, second session, May 13, 1976. Washington: U.S. Government Print Office, 1976.

Minutes from Congressional Hearing on the use of flight simulators. Provides input from each of the Armed Forces and political leaders.

University of Virginia, (1990-1996) *Alice: Easy to Learn Interactive 3D Graphics*. [On-line]. Available: <http://www.cs.virginia.edu/~alice>.

Homepage of the ALICE system software. Provides basic overview of the system and its capabilities.

User Interface Group, Computer Science Department, University of Virginia, Charlottesville, VA.

Members of the ALICE Software development team. Provided technical assistance concerning syntax and capabilities of ALICE programming.

Viewpoint DataLabs, (1996) *Viewpoint DataLabs 3D Digital Content*. [On-line]. Available: <http://www.viewpoint.com>.

Homepage of the ALICE system software. Provides basic overview of the system and its capabilities.

APPENDICES

APPENDIX A: CHANGES MADE BETWEEN VERSIONS OF AMS PRO

A. Changes to Version 1.0

- Pitch aircraft just before takeoff and move camera to side of aircraft.
- Flare aircraft on landing by tilting nose up just before touchdown.
- Move camera with aircraft with landing and stall.
- Change wording of textual descriptions of stall, barrel roll, cloverleaf, lazy eight, and cuban eight.
- Adjust camera angle with immelmann.
- Modify stall, spin, cloverleaf, lazy eight, and cuban eight to look more realistic.
- Turn propeller of T-3 Aircraft.
- Extend list box to show all of the maneuvers at once.

B. Changes to Version 1.1

- Speed up beginning of cloverleaf.
- Adjust camera angle of stall, spin and landing.
- Adjust spin maneuver when fastest speed is chosen.
- Change wording of stall text description.
- Include “Textual Description” label.
- Change Speed Slider label to “Speed of Demonstration” to avoid confusion with entry airspeed.

C. Changes to Version 1.2

- Consolidate “Aerobatics” list box and “Demonstrations” list box to “Maneuvers” list box.
- Move camera closer when showing chandelle.
- Correct barrel roll when fastest demo speed is chosen.
- Stop execution of maneuver when another is selected.
- Reset default position of aircraft when another is chosen.
- Change “Play” label to “Start” to avoid user confusion.
- Add runway to T-3 and aircraft carrier to F-15 takeoff and landing to give more realistic impression.
- Change background colors of F-15 and F-117 aircraft to produce better view.
- Show more of a rear camera on barrel roll.
- Reduce size of scene light to reduce user distraction.
- Correct cuban eight when fastest speed is chosen.

- Reduce taxi time of takeoff.
- Lower camera when viewing landing to show touchdown.
- Alphabetize maneuvers to make them easier to locate.
- Correct stall when executed with the F-15 aircraft.

D. Changes to Version 1.3

- Move textual description to top of control panel to make it more noticeable and easier to view.
- Capitalize labels on buttons to increase readability.
- Show simple directions of how to use AMS Pro at the beginning of the program.
- Change “Stop” button label to “Cancel.” This doesn’t imply that execution of maneuvers may be resumed when the Cancel button is pressed.

APPENDIX B: COMPUTER CODE FOR AMS PRO VERSION 1.4

```
#####
# SET CAMERA WINDOW SIZE AND POSITION #
#####

camwidth = 7000
camheight = 7000
Camera.GetWindow().Move(0, 0, camwidth, camheight)
Camera.GetWindow().SetStyle(NoControls)

#HIDE LIGHT
Light.Resize(.01)

#####
# CREATE AND POSITION AIRCRAFT WITHIN CAMERA WINDOW #
#####

Altitude = 10
CameraDistance = 3

Aircraft1 = AnObject("K:/thesis/code/t3a")
Aircraft1.Move(Up, Altitude)
Aircraft1.Resize(.125)
Aircraft1.plane.prop.Roll(Right, Rate=1000)
Aircraft2 = AnObject("K:/thesis/code/f15")
Aircraft2.Move(Up, Altitude)
Aircraft2.Resize(.05)
Aircraft3 = AnObject("K:/thesis/code/f117_a")
Aircraft3.Move(Up, Altitude)
Aircraft3.Resize(.2)

#SET DEFAULT AIRCRAFT AND HIDE OTHERS

Aircraft=Aircraft1
Aircraft2.Hide()
Aircraft3.Hide()

#CREATE RUNWAYS AND CARRIER FOR TAKEOFF AND LANDING
Runway = AnObject("K:/thesis/code/runway.x")
Runway.Resize(.5)
Runway.Hide()
Carrier = AnObject('Vehicles/Carrier')
Carrier.Turn(Right, 180)
Carrier.Hide()

#CREATE GREY RUNWAY FOR BLACK F-117 FOR VISIBILITY
Square1=AnObject('Shapes/Square')
Square1.Resize(2)
Square1.Turn(Up, 90)
Square1.MoveTo(0,.1,56)
Square2=AnObject('Shapes/Square')
Square2.Resize(2)
Square2.Turn(Up, 90)
Square2.MoveTo(0,.1,58)
```

```
Square3=AnObject('Shapes/Square')
Square3.Resize(2)
Square3.Turn(Up, 90)
Square3.MoveTo(0,.1,60)
Square4=AnObject('Shapes/Square')
Square4.Resize(2)
Square4.Turn(Up, 90)
Square4.MoveTo(0,.1,62)
Square5=AnObject('Shapes/Square')
Square5.Resize(2)
Square5.Turn(Up, 90)
Square5.MoveTo(0,.1,64)
Square6=AnObject('Shapes/Square')
Square6.Resize(2)
Square6.Turn(Up, 90)
Square6.MoveTo(0,.1,66)
Square7=AnObject('Shapes/Square')
Square7.Resize(2)
Square7.Turn(Up, 90)
Square7.MoveTo(0,.1,68)
Square8=AnObject('Shapes/Square')
Square8.Resize(2)
Square8.Turn(Up, 90)
Square8.MoveTo(0,.1,70)
Square9=AnObject('Shapes/Square')
Square9.Resize(2)
Square9.Turn(Up, 90)
Square9.MoveTo(0,.1,70)
Square10=AnObject('Shapes/Square')
Square10.Resize(2)
Square10.Turn(Up, 90)
Square10.MoveTo(0,.1,70)

def ShowRunway2():
    Square1.Show()
    Square2.Show()
    Square3.Show()
    Square4.Show()
    Square5.Show()
    Square6.Show()
    Square7.Show()
    Square8.Show()
    Square9.Show()
    Square10.Show()

def HideRunway2():
    Square1.Hide()
    Square2.Hide()
    Square3.Hide()
    Square4.Hide()
    Square5.Hide()
    Square6.Hide()
    Square7.Hide()
    Square8.Hide()
    Square9.Hide()
    Square10.Hide()

#POSITION CAMERA TO BEST INITIAL VIEW OF AIRCRAFT
```

```

Camera.MoveTo(-CameraDistance, 1, 0, AsSeenBy=Aircraft,
Sytle=Gently)
Camera.LookAt(Aircraft)

#####
#SET CONTROL PANEL SIZE AND POSITION
#####

guiwidth = 1.5*Camera._camera.Surface.Size[0]
guiheight = 1.5*Camera._camera.Surface.Size[1]
MyGUI = AToolkit(Caption='AMS Pro: Aerial Maneuver Simulation
Program')
MyGUI.SetPosition(Camera.GetWindow().GetPosition()[2], 0)
MyGUI.SetSize(guiwidth*.9, guiheight*.7)
SpeedScale = 1.0

#PLACE LABELS ON CONTROL PANEL
Label1 = MyGUI.MakeLabel(Caption='AIRCRAFT', Position=(75,150))
Label4 = MyGUI.MakeLabel(Caption='SELECT A MANEUVER AND PRESS THE
START BUTTON.', Position=(75, 75), Size=(450, 200))
Label5 = MyGUI.MakeLabel(Caption='SPEED OF DEMONSTRATION',
Position=(75, 375))
Label6 = MyGUI.MakeLabel(Caption='Slower', Position=(75, 425))
Label7 = MyGUI.MakeLabel(Caption='Faster', Position=(300, 425),
Size=(100, 30))
Label9 = MyGUI.MakeLabel(Caption='', Position=(410, 600),
Size=(125,30))
Label10 = MyGUI.MakeLabel(Caption='', Position=(75, 50),
Size=(250,15))
Label11 = MyGUI.MakeLabel(Caption='MANEUVERS', Position=(400,150))

#####
# THESE ARE THE CALLBACK FUNCTIONS FOR THE CONROL PANEL #
#####

def Option1Callback(value):
    StopButton()
    Runway.Hide()
    global Aircraft1
    global Aircraft2
    global Aircraft3
    global Aircraft
    if value == 'T-3 Trainer':
        Ground.SetColor(White)
        Ground.SetColor(0,.7,0)
        Scene.SetColor(White)
        Scene.SetColor(.1,.6,.87)
        Aircraft1.Show()
        Aircraft2.Hide()
        Aircraft3.Hide()
        Aircraft = Aircraft1
        Aircraft.MoveTo(0, 10, 0)
        Aircraft.TurnTo(0, 0, 0)
        Camera.MoveTo(-CameraDistance, 1, 0,
AsSeenBy=Aircraft, Sytle=Gently)
        Camera.LookAt(Aircraft)
    elif value == 'F-15 Eagle':
        Ground.SetColor(White)
        Ground.SetColor(Blue)
        Scene.SetColor(White)

```

```

        Scene.SetColor(.1,.6,.87)
        Aircraft1.Hide()
        Aircraft2.Show()
        Aircraft3.Hide()
        Aircraft = Aircraft2
        Aircraft.MoveTo(0, 10, 0)
        Aircraft.TurnTo(0, 0, 0)
        Camera.MoveTo(-CameraDistance, 1, 0,
            AsSeenBy=Aircraft, Sytle=Gently)
        Camera.LookAt(Aircraft)
    elif value == 'F-117 Nighthawk Stealth Fighter':
        Ground.SetColor(White)
        Ground.SetColor(.95,.8,.34)
        Scene.SetColor(White)
        Scene.SetColor(.88,.86,.65)
        Aircraft1.Hide()
        Aircraft2.Hide()
        Aircraft3.Show()
        Aircraft = Aircraft3
        Aircraft.MoveTo(0, 10, 0)
        Aircraft.TurnTo(0, 0, 0)
        Camera.MoveTo(-CameraDistance, 1, 0,
            AsSeenBy=Aircraft, Sytle=Gently)
        Camera.LookAt(Aircraft)

def ListBox2Callback(value):
    StopButton()
    Label10.Caption="TEXTUAL DESCRIPTION OF MANEUVER:"
    if value == 'Takeoff':
        Label4.Caption="TAKEOFF - Acceleration of the aircraft
            at ground level to become airborne."
        PlayButton.SetEnabled(True)
        PlayButton.Callback=PlayTakeoff
    elif value == 'Landing':
        Label4.Caption="LANDING - Deceleration of the aircraft
            while in flight to touchdown safely at ground
            level."
        PlayButton.SetEnabled(True)
        PlayButton.Callback=PlayLanding
    elif value == 'Stall':
        Label4.Caption="STALL - Situation in which the angle
            of attack exceeds the critical value in relation
            to oncoming airflow. Airflow separates across
            the upper surface of the wings but aircraft may
            recover from stall."
        PlayButton.SetEnabled(True)
        PlayButton.Callback=PlayStall
    elif value == 'Spin':
        Label4.Caption="SPIN - A spin is a stall that results
            in excessive plane rotation. The aircraft
            spirals downward in a corkscrew path. One wing
            of the aircraft has more lift than the other and
            gravity brings the aircraft down."
        PlayButton.SetEnabled(True)
        PlayButton.Callback=PlaySpin
    elif value == 'Loop':
        Label4.Caption="LOOP - A loop is a 360 degree turn in
            the aircraft's vertical plane."
        PlayButton.SetEnabled(True)

```

```

        PlayButton.Callback=PlayLoop
elif value == 'Aileron Roll':
    Label4.Caption="AILERON ROLL - An aileron roll is a
        360 degree roll done in either direction."
    PlayButton.SetEnabled(True)
    PlayButton.Callback=PlayAileronRoll
elif value == 'Barrel Roll':
    Label4.Caption="BARREL ROLL - A barrel roll is a roll
        in which the nose of the aircraft rotates about
        a horizontal axis external to the plane."
    PlayButton.SetEnabled(True)
    PlayButton.Callback=PlayBarrelRoll
elif value == 'Split S':
    Label4.Caption="SPLIT S - A reverse in aircraft
        direction by inverting the aircraft and
        performing a half-loop. A large amount of
        altitude is lost if recovery from inverted
        flight is attempted in this manner."
    PlayButton.SetEnabled(True)
    PlayButton.Callback=PlaySplitS
elif value == 'Chandelle':
    Label4.Caption="CHANDELLE - A chandelle is a 180
        degree steep climbing turn with a maximum gain
        of altitude."
    PlayButton.SetEnabled(True)
    PlayButton.Callback=PlayChandelle
elif value == 'Cloverleaf':
    Label4.Caption="CLOVERLEAF - The cloverleaf is
        composed of four identical maneuvers, each begun
        in a vertical plane rotated 90 degrees from the
        preceding one."
    PlayButton.SetEnabled(True)
    PlayButton.Callback=PlayCloverleaf
elif value == 'Immelmann':
    Label4.Caption="IMMELMANN - An immelmann is a half
        loop followed by a half roll, resulting in a 180
        degree turn and an increase in altitude."
    PlayButton.SetEnabled(True)
    PlayButton.Callback=PlayImmelmann
elif value == 'Lazy Eight':
    Label4.Caption="LAZY EIGHT - A maneuver in which the
        aircraft traces a figure eight in a horizontal
        plane."
    PlayButton.SetEnabled(True)
    PlayButton.Callback=PlayLazyEight
elif value == 'Cuban Eight':
    Label4.Caption="CUBAN EIGHT - Maneuver in which the
        aircraft traces a figure eight in the vertical
        plane by executing two loops with aileron rolls
        on the downward sides of each."
    PlayButton.SetEnabled(True)
    PlayButton.Callback=PlayCubanEight

def DisableParts():
    PlayButton.SetEnabled(False)

def EnableParts():
    PlayButton.SetEnabled(True)

def EnablePlay():

```

```

        PlayButton.SetEnabled(True)

def StopButton():
    Scene.Stop()
    Label9.Caption=''
    EnablePlay()

def PauseButton():
    if Aircraft.HasActions():
        Scene.Pause()
        Label9.Caption='PRESS PAUSE TO RESUME.'
    elif Aircraft.HasPausedActions():
        Scene.Resume()
        Label9.Caption=''

def Slider1Callback(Amount):
    global SpeedScale
    SpeedScale = Amount/10

#####
# PLACE CONTROLS ON THE GUI CONTROL PANEL #
#####

# AIRCRAFT OPTION BOX
Option1 = MyGUI.MakeOptionButtonSet(List=['T-3 Trainer', 'F-15
    Eagle', 'F-117 Nighthawk Stealth Fighter'],
    Callback=Option1Callback)
Option1.SetPosition(75, 185)

# AERIAL MANEUVER LIST BOX
ListBox2 = MyGUI.MakeListBox(List=['Aileron Roll', 'Barrel Roll',
    'Chandelle', 'Cloverleaf', 'Cuban Eight', 'Immelmann',
    'Landing', 'Loop', 'Lazy Eight', 'Spin', 'Split S', 'Stall',
    'Takeoff'], Size=(100, 180), Callback=ListBox2Callback)
ListBox2.SetPosition(400, 195)

# CONTROL BUTTONS
PlayButton = MyGUI.MakeButton(Caption='START', Position=(75,475),
    Size=(100,100), Enabled=False)
StopButton = MyGUI.MakeButton(Caption='CANCEL',
    Position=(250,475), Size=(100,100), Enabled=True,
    Callback=StopButton)
PauseButton = MyGUI.MakeButton(Caption='PAUSE',
    Position=(425,475), Size=(100,100), Callback=PauseButton)

#HORIZONTAL SLIDER TO CONTROL SPEED OF MANEUVERS
Slider1 = MyGUI.MakeHorizontalSlider(Position=(75, 400),
    Size=(300, 20), Callback=Slider1Callback, Enabled=True )
Slider1.SetMaximum(20)
Slider1.SetMinimum(10)
Slider1.SetLargeChange(2.5)
Slider1.SetSmallChange(1)
Slider1.SetValue(10)

#####
# THESE ARE THE FUNCTION DEFINITIONS FOR EACH MANEUVER #
#####

```

```

def PlayTakeoff():
    if Aircraft == Aircraft2:
        CarrierTakeoff()
    else:
        RunwayTakeoff()

def RunwayTakeoff():
    Carrier.Hide()
    PlayButton.SetEnabled(False)
    Time = 8/SpeedScale
    FlightLength = 50
    AccelerationInterval = 5
    CameraDistance = 5
    TotalTime = 1.75*Time

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, .1, 0)

    Runway.MoveTo(0, -.25, 0, AsSeenBy=Aircraft)

    if Aircraft == Aircraft3:
        Runway.Hide()
    else:
        Runway.Show()

    Camera.MoveTo(-CameraDistance, 1, 0, AsSeenBy=Aircraft,
        Sytle=Gently)
    Camera.LookAt(Aircraft, EachFrame, Lifetime=(2.75*Time))

    #Phase1 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), Time, Abruptly)
    Phase2 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.75),
        Abruptly)
    Phase3 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.5),
        Abruptly)
    Phase4 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)
    Phase5 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)

    #Cam1 = Camera.Move(Right,
        (FlightLength/AccelerationInterval), Time, Abruptly)
    Cam2 = Camera.Move(Right,
        (FlightLength/AccelerationInterval), (Time*.75),
        Abruptly)
    Cam3 = Camera.Move(Right,
        (FlightLength/AccelerationInterval), (Time*.5),
        Abruptly)
    Cam4 = Camera.Move(Right,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)
    Cam5 = Camera.Move(Right,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)

    seq_anim = MakeSequence(Phase2, Phase3, Phase4, Phase5)

```

```

cam_anim = MakeSequence(Cam2, Cam3, Cam4, Cam5)
seq_anim.Start()
cam_anim.Start()

alarm1 = Aircraft.SetAlarm(((Time*.75)+(Time*.25)),
    Aircraft.Turn,(Up, 10, 1, Abruptly))
alarm2 = Aircraft.SetAlarm(((Time*.75)+(Time*.5)),
    Aircraft.Turn,(Up, 15, 1, Abruptly))
alarm3 =
    Aircraft.SetAlarm(((Time*.75)+(Time*.5)+(Time*.25)),
    Aircraft.Turn,(Up, 15, 1, Abruptly))
alarm4 = Camera.SetAlarm(((Time*.75)+(Time*.5)+(Time*.25)),
    Camera.Move,(Back, CameraDistance, Ground, 2))
alarm5 = Alice.SetAlarm(TotalTime, EnablePlay)

def CarrierTakeoff():
    Runway.Hide()
    PlayButton.SetEnabled(False)
    Time = 8/SpeedScale
    FlightLength = 50
    AccelerationInterval = 5
    CameraDistance = 5
    TotalTime = 1.75*Time

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, 2.4, 0)

    Carrier.MoveTo(0, 0, 7)
    Carrier.Show()

    Camera.MoveTo(-CameraDistance, 1, 0, AsSeenBy=Aircraft,
        Sytle=Gently)
    Camera.LookAt(Aircraft, EachFrame, Lifetime=(2.75*Time))

    #Phase1 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), Time, Abruptly)
    Phase2 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.75),
        Abruptly)
    Phase3 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.5),
        Abruptly)
    Phase4 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)
    Phase5 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)

    #Cam1 = Camera.Move(Right,
        (FlightLength/AccelerationInterval), Time, Abruptly)
    Cam2 = Camera.Move(Right,
        (FlightLength/AccelerationInterval), (Time*.75),
        Abruptly)
    Cam3 = Camera.Move(Right,
        (FlightLength/AccelerationInterval), (Time*.5),
        Abruptly)
    Cam4 = Camera.Move(Right,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)

```

```

Cam5 = Camera.Move(Right,
    (FlightLength/AccelerationInterval), (Time*.25),
    Abruptly)

seq_anim = MakeSequence(Phase2, Phase3, Phase4, Phase5)
cam_anim = MakeSequence(Cam2, Cam3, Cam4, Cam5)
seq_anim.Start()
cam_anim.Start()

alarm1 = Aircraft.SetAlarm(((Time*.75)+(Time*.25)),
    Aircraft.Turn,(Up, 10, 1, Abruptly))
alarm2 = Aircraft.SetAlarm(((Time*.75)+(Time*.5)),
    Aircraft.Turn,(Up, 15, 1, Abruptly))
alarm3 =
    Aircraft.SetAlarm(((Time*.75)+(Time*.5)+(Time*.25)),
    Aircraft.Turn,(Up, 15, 1, Abruptly))
alarm4 = Camera.SetAlarm(((Time*.75)+(Time*.5)+(Time*.25)),
    Camera.Move,(Back, CameraDistance, Ground, 2))
alarm5 = Alice.SetAlarm(TotalTime, EnablePlay)

def PlayLanding():
    if Aircraft == Aircraft2:
        CarrierLanding()
    else:
        RunwayLanding()

def RunwayLanding():
    Carrier.Hide()
    PlayButton.Enabled=False
    Altitude = 8
    Time = 8/SpeedScale
    FlightLength = 60
    AccelerationInterval = 5
    CameraDistance = 5
    TotalTime = 2.25*Time

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

    Runway.MoveTo(0, -Altitude-.25, 70, AsSeenBy=Aircraft)

    if Aircraft == Aircraft3:
        Runway.Hide()
        ShowRunway2()
    else:
        Runway.Show()

    Camera.MoveTo(-CameraDistance, Altitude/2, FlightLength/2)
    Camera.LookAt(Aircraft, EachFrame, Lifetime=(2.75*Time))

    Phase1 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)
    Phase2 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)
    Phase3 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.5),
        Abruptly)

```

```

Phase4 = Aircraft.Move(Forward,
    (FlightLength/AccelerationInterval), (Time*.25),
    Abruptly)
Phase5 = Aircraft.Move(Forward,
    (FlightLength/AccelerationInterval)*2, (Time*.5)*2,
    Abruptly)

seq_anim = MakeSequence(Phase1, Phase2, Phase3, Phase4,
    Phase5)
seq_anim.Start()

Camera.Move(Forward, FlightLength/2, Aircraft, TotalTime,
    Abruptly)
Camera.Move(Down, Altitude/2-1, TotalTime, Abruptly)

alarm1 = Aircraft.SetAlarm((Time*.25), Aircraft.Move, (Down,
    Altitude-.5, (1.25*Time), Abruptly))
alarm2 = Aircraft.SetAlarm(Time+Time*.375,
    Aircraft.Turn, (Up, 35, 1))
alarm3 = Aircraft.SetAlarm(Time+Time*.375,
    Aircraft.Move, (Down, .5, .5))
alarm4 = Aircraft.SetAlarm(Time+Time*.375+.25,
    Aircraft.Turn, (Down, 35, 1))
alarm5 = Alice.SetAlarm(TotalTime, EnablePlay, ())

def CarrierLanding():
    Runway.Hide()
    PlayButton.Enabled=False
    Altitude = 15
    Time = 8
    FlightLength = 60
    AccelerationInterval = 5
    CameraDistance = 5
    TotalTime = 2.25*Time

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

    Carrier.MoveTo(0, -Altitude-.1, 62, AsSeenBy=Aircraft)
    Carrier.Show()

    Camera.MoveTo(-CameraDistance, .75*Altitude,
        .6*FlightLength)
    Camera.LookAt(Aircraft, EachFrame, Lifetime=(2.75*Time))

    Phase1 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)
    Phase2 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)
    Phase3 = Aircraft.Move(Forward, Abruptly)
    Phase4 = Aircraft.Move(Forward,
        (FlightLength/AccelerationInterval), (Time*.25),
        Abruptly)
    Phase5 = Aircraft.Move(Forward,
        (FlightLength*.9/AccelerationInterval)*2, (Time*.5)*2,
        Abruptly)

    seq_anim = MakeSequence(Phase1, Phase2, Phase3, Phase4,

```

```

        Phase5)
    seq_anim.Start()

    Camera.Move(Forward, FlightLength/2,Aircraft,TotalTime,
        Abruptly)
    Camera.Move(Down, .55*Altitude, .6*TotalTime, Abruptly)

    alarm1 = Aircraft.SetAlarm((Time*.25), Aircraft.Move,(Down,
        .825*Altitude,(1.25*Time), Abruptly))
    alarm2 = Aircraft.SetAlarm(Time+Time*.375,
        Aircraft.Turn,(Up, 35, .125*Time))
    alarm3 = Aircraft.SetAlarm(Time+Time*.375,
        Aircraft.Move,(Down, .5, .0625*Time))
    alarm4 = Aircraft.SetAlarm(Time+Time*.375+.25,
        Aircraft.Turn,(Down, 35, .125*Time))
    alarm5 = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlayStall():
    Carrier.Hide()
    Runway.Hide()
    PlayButton.SetEnabled(False)
    Altitude = 20
    Time = 7
    FlightLength = 15
    AccelerationInterval = 5
    CameraDistance = 5
    TotalTime = 1.5*Time

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

    Camera.MoveTo(-CameraDistance, -2.5, FlightLength,
        AsSeenBy=Aircraft)
    Camera.LookAt(Aircraft, Eachframe, Lifetime=(1.5*Time))

    NormalFlight = Aircraft.Move(Forward, FlightLength, Time,
        Abruptly)
    Descend = Aircraft.Move(Down,(Altitude*.3),(Time/3), Gently)

    seq_anim = MakeSequence(NormalFlight, Descend)
    seq_anim.Start()

    alarm1 = Aircraft.SetAlarm((.9*Time), Aircraft.Turn,(Up, 30,
        Aircraft, .5, Abruptly))
    alarm2 = Aircraft.SetAlarm(Time+Time/8, Aircraft.Turn,(Down,
        80, Aircraft, 1.5, Abruptly))
    alarm3 = Aircraft.SetAlarm(Time+Time/5, Aircraft.Turn,(Up,
        50, Aircraft, 1, Abruptly))
    alarm4 = Aircraft.SetAlarm(Time+Time/4,
        Aircraft.Move,(Forward, FlightLength/2, Time/2,
        Abruptly))
    alarm5 = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlaySpin():
    Carrier.Hide()
    Runway.Hide()
    global plane_tube_distance
    PlayButton.Enabled=False
    Altitude = 20
    Time = 8

```

```

FlightLength = 40
AccelerationInterval = 5
CameraDistance = 7
TotalTime =(7/3)*Time

Aircraft.TurnTo(0, 0, 0)
Aircraft.MoveTo(0, Altitude, 0)

thetube = AnObject("shapes/tube")
thetube.resize(toptobottom, 50)
thetube.MoveTo(2, 0, FlightLength, AsSeenBy=Aircraft)
thetube.Hide()

Camera.MoveTo(-CameraDistance, -1.5, 0, Eachframe,
    Lifetime=(Time), AsSeenBy=Aircraft)
Camera.LookAt(Aircraft, EachFrame, Lifetime=TotalTime)

NormalFlight = Aircraft.Move(Forward, FlightLength, Time,
    Abruptly)
ExecuteSpin = Aircraft.Turn(Left, 1080, thetube, Time,
    Abruptly)
ExecuteSpin.Stop()
ResumeFlight = Aircraft.Move(Forward, (FlightLength/2), Time,
    Abruptly)
ResumeFlight.Stop()

seq_anim = MakeSequence(NormalFlight, ExecuteSpin,
    ResumeFlight)
seq_anim.Start()

alarm1 = Aircraft.SetAlarm(Time-1, Aircraft.Turn,(Up, 15,
    .5))
alarm2 = Aircraft.SetAlarm(Time, Aircraft.Turn,(Down, 30,
    .5))
alarm3 = Aircraft.SetAlarm(Time, Aircraft.Roll,(Left, 20, 1,
    Abruptly))
alarm4 = Aircraft.SetAlarm(Time, Aircraft.Move,(Down,
    Altitude/2, Time, Gently))
alarm5 = Aircraft.SetAlarm(2*Time, Aircraft.Roll,(Right, 20,
    .5))
alarm6 = Aircraft.SetAlarm(2*Time, Aircraft.Turn,(Up, 15,
    1))
alarm7 = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlayLoop():
    Carrier.Hide()
    Runway.Hide()
    PlayButton.Enabled=False
    Altitude = 20
    LoopRadius = 2
    FlightDistance = 10
    LevelDuration = 5/SpeedScale
    LoopDuration = 5/SpeedScale
    CameraDistance = 5
    TotalTime = 2*LevelDuration + LoopDuration

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

```

```

Tack=AnObject("shapes/cube")
Tack.Resize(.01)
Tack.SetColor(SkyBlue)
Tack.Move(Up, Altitude + LoopRadius)
Tack.Move(Forward, FlightDistance)

Camera.MoveTo(-CameraDistance, 0, FlightDistance,
              AsSeenBy=Aircraft)
Camera.LookAt(Aircraft, EachFrame, Lifetime=(LevelDuration*2
              + LoopDuration))

#LOOP SEQUENCE IS BROKEN INTO THREE SEPARATE PARTS
InitLevelFlight = Aircraft.Move(Forward, FlightDistance,
                                LevelDuration, Abruptly)
ExecuteLoop = Aircraft.Turn(Up, 360, Tack, LoopDuration,
                             Abruptly)
ResumeLevelFlight = Aircraft.Move(Forward, FlightDistance,
                                  LevelDuration, Abruptly)

seq_anim = MakeSequence(InitLevelFlight, ExecuteLoop,
                        ResumeLevelFlight)
seq_anim.Start()

alarm = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlayAileronRoll():
    Carrier.Hide()
    Runway.Hide()
    PlayButton.Enabled=False
    Altitude = 20
    FlightDistance = 20
    FlightDuration = 10/SpeedScale
    RollDuration = 5/SpeedScale
    CameraDistance = 5
    TotalTime = FlightDuration

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

    Camera.MoveTo(-CameraDistance, 0, FlightDistance/2,
                  AsSeenBy=Aircraft)
    Camera.LookAt(Aircraft, EachFrame, Lifetime=FlightDuration)

    Aircraft.Move(Forward, FlightDistance, FlightDuration,
                  Abruptly)

    alarm1=Aircraft.SetAlarm(FlightDuration/3,
                             Aircraft.Roll,(Right, 360, RollDuration))
    alarm2 = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlayBarrelRoll():
    Carrier.Hide()
    Runway.Hide()
    PlayButton.Enabled=False
    Altitude = 20
    FlightDistance = 20
    FlightDuration = 15/SpeedScale
    RollDuration = 7/SpeedScale
    RollRadius = 3
    CameraDistance = 5

```

```

TotalTime = FlightDuration

Aircraft.MoveTo(0, Altitude, 0)

Tack=AnObject("shapes/cube")
Tack.Resize(.01)
Tack.SetColor(SkyBlue)
Tack.Move(Up, Altitude + RollRadius)

Camera.MoveTo(-CameraDistance, 0, FlightDistance/3,
              AsSeenBy=Aircraft)
Camera.LookAt(Aircraft, EachFrame, Lifetime=FlightDuration)

ExecuteRoll=Aircraft.Roll(Right, 360, RollDuration,
                          AsSeenBy=Tack)

Aircraft.Move(Forward, FlightDistance, FlightDuration,
              Abruptly)

alarm1 = Aircraft.SetAlarm((FlightDuration/3), ExecuteRoll)
alarm2 = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlaySplitS():
    Carrier.Hide()
    Runway.Hide()
    PlayButton.Enabled=False
    Altitude = 15
    LoopRadius = 2
    FlightDistance = 10
    LevelDuration = 7/SpeedScale
    RollDuration = 2/SpeedScale
    LoopDuration = 2/SpeedScale
    CameraDistance = 5
    TotalTime = 1.5*LevelDuration + LoopDuration

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

    Tack=AnObject("shapes/cube")
    Tack.Resize(.01)
    Tack.SetColor(SkyBlue)
    Tack.Move(Up, Altitude - LoopRadius)
    Tack.Move(Forward, FlightDistance)
    Tack.Turn(Right, 180)

    Camera.MoveTo(-CameraDistance, 0, LoopRadius,
                  AsSeenBy=Aircraft)
    Camera.LookAt(Aircraft, EachFrame,
                  Lifetime=(1.5*LevelDuration +
                             LoopDuration))

    InitLevelFlight = Aircraft.Move(Forward, FlightDistance,
                                    LevelDuration, Abruptly)
    ReverseDirection = Aircraft.Turn(Up, 180, Tack,
                                     LoopDuration, Abruptly)
    ResumeFlight = Aircraft.Move(Forward, FlightDistance,
                                 LevelDuration/2, Abruptly)
    seq_anim = MakeSequence(InitLevelFlight, ReverseDirection,
                             ResumeFlight)
    seq_anim.Start()

```

```

#EXECUTE ROLL AT THE END OF LEVEL FLIGHT
alarm1 = Aircraft.SetAlarm((LevelDuration-RollDuration),
    Aircraft.Roll,(Right, 180, RollDuration))

alarm2 = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlayChandelle():
    Carrier.Hide()
    Runway.Hide()
    PlayButton.Enabled=False
    Altitude = 10
    LoopRadius = 3
    FlightLength = 40
    Time = 10/SpeedScale
    TurnDuration =(Time/2)
    CameraDistance = 5
    TotalTime = Time*2

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

    Camera.MoveTo(-CameraDistance, .5,(FlightLength/4),
        AsSeenBy=Aircraft)
    Camera.LookAt(Aircraft, EachFrame, Lifetime=(Time*2))

    Aircraft.Move(Forward, FlightLength,(Time * 2), Abruptly)

    #ROLL LEFT 45 DEGREES
    alarm1 = Aircraft.SetAlarm((Time-2), Aircraft.Roll,(Left,
        45, 2))

    #EXECUTE TURN
    alarm2 = Aircraft.SetAlarm(Time, Aircraft.Turn,(Up, 180,
        TurnDuration, Abruptly))

    #ROLL OUT AND RESUME LEVEL FLIGHT
    alarm3 = Aircraft.SetAlarm((Time + TurnDuration),
        Aircraft.Roll,(Right, 125, 3))

    alarm4 = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlayCloverleaf():
    Carrier.Hide()
    Runway.Hide()
    PlayButton.Enabled=False
    Altitude = 20
    FlightLength = 70
    Time = 10
    CameraDistance = 5
    TotalTime = 3.5*Time

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

    Camera.MoveTo(-CameraDistance, 3.5, 0, Eachframe,
        Lifetime=.4*Time, AsSeenBy=Aircraft)
    Camera.LookAt(Aircraft, EachFrame, Lifetime=TotalTime)

```

```

Clover = MakeSequence(Aircraft.Roll(Left, 90, .05*Time),
    Aircraft.Turn(Up, 360, Rate=60))
Clover.Stop()
ResumeFlight = MakeSequence(Aircraft.Roll(Left, 90,
    .1*Time), Aircraft.Turn(Up, 270, Rate=60))
ResumeFlight.Stop()

Aircraft.Move(Forward, FlightLength, TotalTime, Abruptly)

seq_anim = MakeSequence(Clover.Start(), Clover.Start(),
    Clover.Start(), ResumeFlight.Start())

alarm1 = Aircraft.SetAlarm(.4*Time, Aircraft.Turn,(Up, 90,
    .1*Time))
alarm2 = Alice.SetAlarm(.5*Time, seq_anim.Start)
alarm3 = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlayImmelmann():
    Carrier.Hide()
    Runway.Hide()
    PlayButton.Enabled=False
    Altitude = 15
    FlightDistance = 50
    RollDuration = 3/SpeedScale
    LoopDuration = 4/SpeedScale
    CameraDistance = 5
    Time = 15/SpeedScale
    TotalTime = Time

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

    Camera.MoveTo(-CameraDistance, 6, CameraDistance*4,
        AsSeenBy=Aircraft)
    Camera.LookAt(Aircraft, EachFrame, Lifetime=Time)

    Aircraft.Move(Forward, FlightDistance, Time)

    #EXECUTE HALF LOOP TO REVERSE DIRECTION
    alarm1 = Aircraft.SetAlarm(Time/2, Aircraft.Turn,(Up, 180,
        LoopDuration))
    #EXECUTE HALF ROLL TO RESUME LEVEL FLIGHT
    alarm2 = Aircraft.SetAlarm(Time*.75, Aircraft.Roll,(Right,
        180, RollDuration))

    alarm3 = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlayLazyEight():
    Carrier.Hide()
    Runway.Hide()
    PlayButton.Enabled=False
    Altitude = 10
    LoopRadius = 3
    FlightLength = 80
    Time = 20/SpeedScale
    TurnDuration = 3/SpeedScale
    CameraDistance = 5
    TotalTime = 1.5*Time

    Aircraft.TurnTo(0, 0, 0)

```

```

Aircraft.MoveTo(0, Altitude, 0)

Camera.MoveTo(-CameraDistance, 1.5, 0, Eachframe,
    Lifetime=.075*Time, AsSeenBy=Aircraft)
Camera.LookAt(Aircraft, EachFrame, Lifetime=TotalTime)

Aircraft.Move(Forward, FlightLength, 1.5*Time, Abruptly)

#EXECUTE FIRST TURN
alarm1 = Aircraft.SetAlarm(.15*Time, Aircraft.Roll,(Left,
    45, .1*Time))
alarm2 = Aircraft.SetAlarm(.25*Time, Aircraft.Turn,(Up, 225,
    .25*Time))
alarm3 = Aircraft.SetAlarm(.5*Time, Aircraft.Roll,(Right,
    135, .1*Time))
alarm4 = Aircraft.SetAlarm(.6*Time, Aircraft.Turn(Up, 45,
    .1*Time))
alarm5 = Aircraft.SetAlarm(.6*Time, Aircraft.Turn(Right, 45,
    .1*Time))

#PLANE IS NOW FLYING LEVEL IN OPPOSITE DIRECTION

#EXECUTE SECOND TURN
alarm6 = Aircraft.SetAlarm(.75*Time, Aircraft.Roll,(Right,
    45, .1*Time))
alarm7 = Aircraft.SetAlarm(.85*Time, Aircraft.Turn,(Up, 225,
    .25*Time))
alarm8 = Aircraft.SetAlarm(1.1*Time, Aircraft.Roll,(Left,
    135, .1*Time))
alarm9 = Aircraft.SetAlarm(1.2*Time, Aircraft.Turn,(Up, 45,
    .1*Time))
alarm10 = Aircraft.SetAlarm(1.2*Time, Aircraft.Turn,(Left,
    45, .1*Time))
alarm11 = Alice.SetAlarm(TotalTime, EnablePlay,())

def PlayCubanEight():
    Carrier.Hide()
    Runway.Hide()
    PlayButton.Enabled=False
    Altitude = 30
    LoopRadius = 2
    FlightDistance = 80
    Time = 20
    LoopDuration = 3/SpeedScale
    RollDuration = 1/SpeedScale
    CameraDistance = 5
    TotalTime = Time

    Aircraft.TurnTo(0, 0, 0)
    Aircraft.MoveTo(0, Altitude, 0)

    Tack=AnObject("shapes/cube")
    Tack.Resize(.01)
    Tack.SetColor(SkyBlue)
    Tack.Move(Up, Altitude + LoopRadius)
    Tack.Move(Forward, FlightDistance)

    Camera.MoveTo(-CameraDistance, 1.5, 6, Eachframe,
        Lifetime=.1*Time, AsSeenBy=Aircraft)
    Camera.LookAt(Aircraft, EachFrame, Lifetime=Time)

```

```
Aircraft.Move(Forward, FlightDistance, Time)

alarm1 = Aircraft.SetAlarm(.25*Time, Aircraft.Turn,(Up, 225,
    Aircraft, LoopDuration))
alarm2 = Aircraft.SetAlarm(.4*Time, Aircraft.Roll,(Right,
    180, RollDuration))
alarm3 = Aircraft.SetAlarm(.425*Time, Aircraft.Turn,(Up,
    270, Aircraft, LoopDuration))
alarm4 = Aircraft.SetAlarm(.55*Time, Aircraft.Roll,(Left,
    180, RollDuration))
alarm5 = Aircraft.SetAlarm(.575*Time, Aircraft.Turn,(Up, 45,
    Aircraft, RollDuration))
alarm6 = Alice.SetAlarm(TotalTime, EnablePlay,())

#####
# END OF PROGRAM #
#####
```