University of Louisville
ThinkIR: The University of Louisville's Institutional Repository

Electronic Theses and Dissertations

5-2012

3D visualization of in-flight recorded data.

Gyuchoon Cho 1975-
University of Louisville

Follow this and additional works at: https://ir.library.louisville.edu/etd

Recommended Citation

Cho, Gyuchoon 1975-, "3D visualization of in-flight recorded data." (2012). *Electronic Theses and Dissertations.* Paper 244.
https://doi.org/10.18297/etd/244

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

3D VISUALIZATION OF IN-FLIGHT RECORDED DATA

By

Gyuchoon Cho
B.S, Armstrong Atlantic State University, 2006
M.S, Western Kentucky University, 2009

A Dissertation
Submitted to the faculty of the
J.B. Speed School of Engineering of the University of Louisville
In Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

May 2012

Copyright 2012 by Gyuchoon Cho

All rights reserved

3D VISUALIZATION OF IN-FLIGHT RECORDED DATA

By

Gyuchoon Cho
B.S, Armstrong Atlantic State University, 2006
M.S, Western Kentucky University, 2009

A Dissertation Approved on

April 20, 2012

by the following Dissertation Committee:

Dissertation Director - Ming Ouyang, Ph.D.

Dar-jen Chang, Ph.D.

Ibrahim N. Imam, Ph.D.

Roman Yampolskiy, Ph.D

Tim Hardin, Ph.D.

DEDICATION

This dissertation is dedicated to my parents

Mr. Dong-Hak Cho

and

Mrs. Kwi-Ran Son

who have given me a full confidence to study in U.S.

And, my brother

Gyu-Hoon Cho

who was my most respected person and who will be with me in the heaven in the future.

ACKNOWLEDGMENTS

While preparing this dissertation, I have felt that I am the lucky person who can finish this with many thankful people. The first person who I would like to thank is my adviser, Dr. Ming Ouyang. He led me the right direction when I was in dejection because I couldn't find my research area at that moment. One day, he gave me a key for his research laboratory, VaIG or Visualization and Intensive Graphics Lab. I believe that the key was not only physically open the door but also opened my future. At the VaIG, I could learn about the latest technologies and share useful information with other lab members. Dr. Dar-Jen Chang, the one of my respect people, always encouraged me when I doubted whether I could finish this project and dissertation. I took his class about Computer Graphics when I began as a Ph.D. student. He introduced students his full knowledge of its area and I was deeply impressed enough to choose this project. I also thank Sung-bo Jung, lab-mate at the VaIG. He was like my second advisor during my Ph.D. period. As a new student, sometimes, I did not know what I should do. He showed me a way to solve my problem in the campus life and I think I couldn't graduate this school without his helps. I would like to thank my wife, Jungmee Myung, who has supported me to do this dissertation by taking care of our lovely children, Minseo and Mini, without me. I would like to thank my committee members, Dr. Ibrahim N. Imam, Dr. Roman Yampolskiy and Dr. Tim Hardin. They gave me instructive comments. Special thanks to my parents and siblings in Korea.

ABSTRACT
3D VISUALIZATION OF IN-FLIGHT RECORDED DATA

Gyuchoon Cho

May 20, 2012

Human being can easily acquire information by showing the object than reading the description of it. Our brain stores images that the eyes are seeing and by the brain mapping, people can analyze information by imagination in the brain. This is the reason why visualization is important and powerful. It helps people remember the scene later. Visualization transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations (Flurchick, 2001). As a consequence, many computer scientists and programmers take their time to build better visualization of the data for users. For the flight data from an aircraft, it is better to understand data in 3D computer graphics rather than to look at mere numbers.

The flight data consists of several fields such as elapsed time, latitude, longitude, altitude, ground speed, roll angle, pitch angle, heading, wind speed, and so on. With these data variables, filtering is the first process for visualization in order to gather important information. The collection of processed data is transformed to 3D graphics form to be rendered by generating Keyhole Mark-up Language (KML) files in the system. KML is an XML grammar and file format for modeling and storing geographic features such as points, lines, images, polygons, and models for display in Google Earth or Google Maps. Like HTML, KML has a tag-based structure with names and attributes used for specific display purposes.

In the present work, new approaches to visualize flight using Google Earth are developed. Because of the limitation of the Google Earth API, the Great Circle Distance calculation and trigonometric functions are implemented to handle the position, angles of roll and pitch, and a range of the camera positions to generate several points of view. Currently, visual representation of flight data depends on 2D graphics although an aircraft flies in a 3D space. The graphical interface allows flight analysts to create ground traces in 2D, and flight "ribbons" and flight "paths with altitude" in 3D. Additionally, by incorporating weather information, fog and clouds can also be generated as part of the animation effects. With 3D stereoscopic technique, a realistic visual representation of the flights is realized.

TABLE OF CONTENTS

PAGE

DEDICATION iii
ACKNOWLEDGMENTS iv
ABSTRACT v
LIST OF TABLES x
LIST OF FIGURES xi

CHAPTER

1. INTRODUCTION 1
   1.1 Motivation 1
   1.2 3D Flight Visualization System 2
   1.3 Organization of the Dissertation 3
2. RESEARCHES RELATED TO THE TOPIC 4
   2.1 The Uses of Multivariate Data Analysis 4
   2.2 Geographical Calculations 4
      2.2.1 Coordinate Systems 6
      2.2.2 Geographical Calculation 10
   2.3 Visualization 14
   2.4 3D Computer Graphics 15
3. FLIGHT VISUALIZATION 17
   3.1 Introduction 17
   3.2 Flight Visualization 17
      3.2.1 CodeGear RAD Studio 2009 17

3.2.2 Google SketchUP.......... 19

3.2.3 Google Earth.......... 20

3.2.3.1 Direct X Versus OpenGL.......... 21

3.2.3.2 Terrains and Accuracy.......... 22

3.2.4 System Architecture.......... 29

3.2.4.1 Raw Data.......... 30

3.2.4.2 Convert Options.......... 33

4. RESULTS AND COMPARISONS.......... 49

4.1 Introduction.......... 49

4.2 Effects.......... 49

4.2.1 Visibility 49

4.2.2 Clouds Effects 51

4.3 Analysis 55

4.3.1 Noisy Data Removal 55

4.3.2 Ribbon Representation 56

4.3.3 Path with Altitude 58

4.4 Comparisons 59

4.4.1 Flight Graphical User Interface 59

4.4.2 Autonomous Quadrotor Flight Test 60

4.4.3 Flight Visualization using the HUMS 61

5. FUTURE WORKS.......... 63

5.1 Introduction.......... 63

5.2 Flight Data.......... 63

5.3 Weather condition 64

5.4 Collision Detection for Turbulence Effect 67

5.5 Survey of Geographical Calculations.. 69

5.6 Stereoscopic Vision... 70

REFERENCES 71

APPENDICES... 74

APPENDIX A: GLOSSARY... 74

APPENDIX B: SAMPLES OF IN-FLIGHT RECORDED DATA 76

APPENDIX C: SAMPLES OF GENERATED KML CODES 77

CURRICULUM VITAE.. 80

LIST OF TABLES

TABLE PAGE

2.1 Notation for the Vincenty formula.. 12

3.1 Camera's directions.. 44

LIST OF FIGURES

FIGURE PAGE

2.1 Distance and bearing between two points 5

2.2 Path from Baghdad to Osaka 5

2.3 The WGS84 coordinate system definition 7

2.4 Different types of the map projection 8

2.5 Cylindrical projection of the globe 9

2.6 World map on a cylindrical projection 9

3.1 Screen shot of SketchUP 20

3.2 OpenGL Volumizer 22

3.3 Google Earth's 3D buildings and Terrains 23

3.4 Standard of Digital Evaluation Model 24

3.5 Example of 1m and 7m DEM 26

3.6 DEM at Pennsylvania, USA 28

3.7 EarthExplorer: New Orleans region SRTM data 29

3.8 System Architecture 29

3.9 Screen shot of an application 30

3.10 Fixed-length versus Variable-length data type 31

3.11 Orientation in Google Earth 32

3.12 Convert Option Screen 33

3.13 Path and Ribbon Options 34

3.14 Origin of the aircraft from data 35

3.15 Changing origin based on distance and pitch angle 35

3.16 The shortest path on Earth 36

3.17 Circle of radius 1 37

3.18 Sphere of radius 1 37

3.19 Section of the Point A, C, E, G, and O 38

3.20 Trapezoid of ABCD 39

3.21 Positions of wingtips relative to the aircraft center of mass........................ 41
3.22 Adjustment of wingtip positions due to aircraft roll angles......................... 42
3.23 Select Direction tab in Animation Option.. 43
3.24 Directions in Google Earth.. 45
3.25 Range, orientation, and altitude in Google Earth 46
3.26 Distance vs. Range.. 47
3.27 Sequences Option.. 48
4.1 Clamped to ground and Absolute mode.. 50
4.2 Visibility Option dialog.. 50
4.3 Result of different opaque values... 51
4.4 Clouds option dialog.. 52
4.5 Common types of cloud... 53
4.6 A cloud model... 53
4.7 Combination of cloud types generated.. 54
4.8 Result of flight visualization with clouds... 54
4.9 An example of noisy data.. 55
4.10 Cleaned data... 55
4.11 Analyzing flight data in 2D.. 57
4.12 3D Visualization of ribbon of 17 flights... 57
4.13 3D Visualization of paths with altitudes... 58
4.14 Multiple Flight Analysis menu.. 58
4.15 FlightGUI user interface... 60
4.16 Snapshot of simulation in Flightgear... 61
4.17 Full screen shot display of flight visualization..................................... 62
5.1 Different time in Google Earth.. 64
5.2 Regional or uniform haze on a Glacier National Park 66
5.3 Teschner's collision detection algorithm 68

CHAPTER 1

INTRODUCTION

1.1 Motivation

Human being can easily acquire information by looking at the object than by reading a description of it. Our brain stores images, and by brain mapping, people can analyze information gathered from memory in the brain. This is the reason why visualization is important and powerful. It helps people remember the scene later. Visualization transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. As a consequence, many computer scientists and programmers take their time to build better visualization of their data for users. For the data from a flight, it is better to understand them with 3D computer graphics rather than just looking at all the numbers.

Flight simulation is to create virtual flights and to visualize how aircraft flies. One of the main purposes for flight simulation is flight training for pilots. Before taking the first real flight, pilots are trained with various scenarios in flight simulators, and they learn how react to hazardous situations such as turbulence, precipitation, and so on. By experiencing flight in the virtual world, pilots are able to prepare for real flights, and it is more realistic than reading about flights in a handbook. This is why flight simulation is considered an important step for the pilots, and this technology has been improved in the previous couple of decades. There are flight simulation software packages that visually display flight in 3D graphics. These software packages are getting more realistic with accurate worldwide scenery based on the actual terrain. In the market place, most products are developed as an alternative to professional flight simulators for pilot training with yoke and throttle devices, or are used to play games with others in the sky of the virtual battle field. However, the interest is to have a system that analyzes real world in-flight recorded data with meteorological effects via 3D computer graphics. The purpose of the present work is completely different from these products. Details of flight simulation and flight visualization will be discussed in Chapter 2. The present work proposes a new approach to visualize flight using Google Earth.

1.2 3D Flight Visualization System

The system used as a flight data analysis tool has several advantages:

- Converting data into an optimized KML code
- Setting view properties by geographic calculations
- Configurable flight visual animations
- Custom data template
- Data analysis

Details of the system architecture and algorithms will be discussed in Chapter 3. The flight data consist of several fields such as latitude, longitude, ground speed, magnetic heading, wind direction, aircraft heading (corrected), pressure altitude, roll angle, and pitch angle. Different sources of raw data may have different fields. This system collects important information from the raw data and generates Keyhole Mark-up

Language (KML). KML is an XML grammar and file format for modeling and storing geographic features such as points, lines, images, polygons, and models for display in Google Earth or Google Maps. This file is processed by Google Earth in a similar way that HTML and XML files are processed by web browsers. Like HTML, KML has a tag-based structure with names and attributes used for specific display purpose. Thus, Google Earth acts as browsers of KML files.

Because of the limitation of the Google Earth API, the system uses the Great Circle Distance calculation to handle the position, angle, and range of the camera to generate several points of view. With this calculation, it displays a data with computer graphics in more efficient way to show the result. Currently, visual representation of flight data depends on 2D graphics although an aircraft flies in a 3D space. By adding graphical interface for flight analysts, they are able to create ground trace in 2D and ribbons in 3D. With 3D stereoscopic technique, they can also watch it with a realistic visual representation of the flights.

1.3 Organization of the Dissertation

Chapter 2 surveys different algorithms from different aspects regarding flight simulation and visualization, and the basic concepts of the geographical calculation. Chapter 3 describes the system architecture and how it is applied to flight data analysis. It also covers the details of the Great Circle Distance calculation and theoretical foundation of terrain.

CHAPTER 2

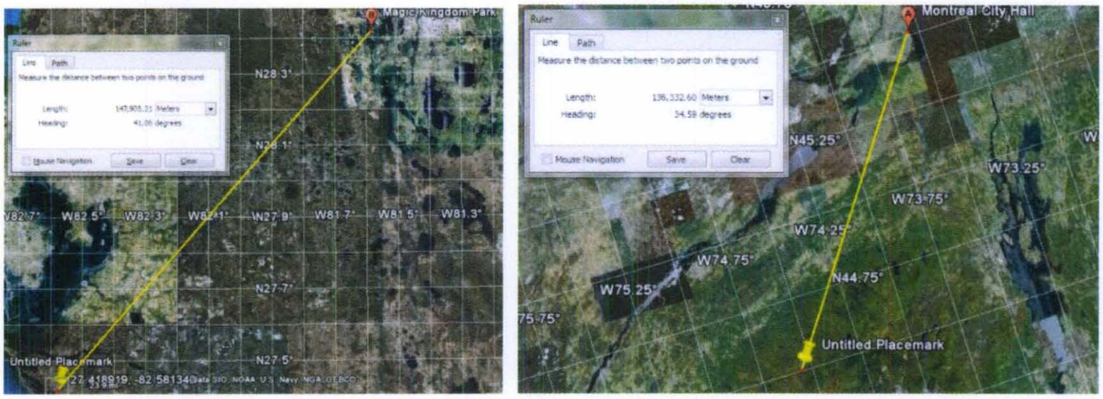
RESEARCHES RELATED TO THE TOPIC

2.1 Introduction

In this chapter, some geographic calculations are introduced. The system needs to use these formulas in order to set up the camera position. Google Earth API or KML do not support any function to calculate the position from the distance and angle. Finding the position of camera is the most important task in the system. Moreover, this chapter explains why this calculation has to be used instead of simple mathematical calculation with an example. Additionally, research on visualization is introduced in this chapter too.

2.2 Geographical Calculations

Two simple examples are initially provided before geographical calculations are discussed. The Cinderella Castle in the Magic Kingdom of Disney World (Florida, USA) is located at this coordinate: N28.418919° and W81.581340°. The city hall of Montreal, Canada is at N45.508463° and W73.554019°. The Cinderella Castle is closer to the Equator, while the Montreal city hall is closer to the North Pole. For each of these two places, let the coordinates of the camera position be 1° (both latitude and longitude) less than the site. Figure 2.1 shows the resulting distance and bearing from the camera to the site.



Measure the distance between two points on the ground
Length: 147,905.21 Meters
Heading: 41.06 degrees

Measure the distance between two points on the ground
Length: 136,332.60 Meters
Heading: 34.59 degrees

Magic Kingdom: N28.418919°, W81.581340°

Destination point: N27.418919°, W82.581340°

Montreal city hall: N45.508463°, W73.554019°

Destination point: N44.508463°, W74.554019°

Figure 2.1 Distance and bearing between two points

As Figure 2.1 shows, the distance between the Cinderella Castle and the camera position is 147,905.21 meters, whereas the distance between Montreal city hall and its camera position is 136,332.60 meters. There is about 11.6 km difference. Moreover, bearing to the Cinderella Castle is 41.06°, whereas it is 34.59° to the Montreal city hall.

In the second example, for a great circle line (on the surface of the Earth) from Baghdad (N35°, E45°) to Osaka (N35°, E135°), the start bearing value is 60° while the end bearing value is 120°. Figure 2.2 shows the great circle path from Baghdad to Osaka.

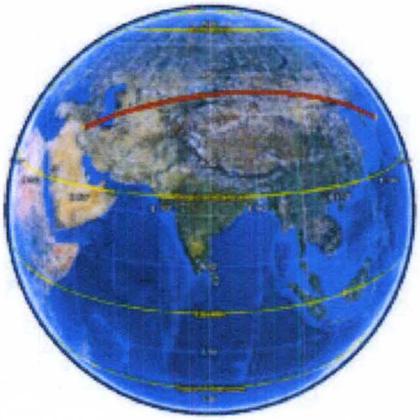


Figure 2.2 Path from Baghdad to Osaka

As the above examples show, the bearing and distance are not the same in different places on the Earth. That is because the shape of the Earth is not flat and not quite a sphere. The Earth is an irregular shape approximating a biaxial ellipsoid. The radius at the equator of the Earth is about 0.3% longer than the radius that is measured through the poles. For these reasons, simple calculation cannot be applied to the Earth and many researchers have developed and improved the geographic coordinate system and its calculations. The system uses one of the geographical calculations, Great Earth distance or also known as Haversine formula. This formula is not the most accurate one because it can have large rounding errors if the distance between two points is small, such as less than 1 meter. However, this is not a concern in the system because the camera position will be at least several meters away from the aircraft. Chapter 3 describes this formula in detail. In the real world of coordinated points, all formulas are not perfect because the Earth does not have the perfect geometry and a point in a coordinate system is obtained from several observations that are averaged together using some assumptions.

2.2.1 Coordinate Systems

There are two types of coordinate systems: Geographic Coordinate Systems and Projected Coordinate systems.

Geographic coordinate system

A geographic coordinate system is a reference system to define the location of points on the Earth specified by a set of numbers such as latitude and longitude. In this coordinate system, lines of latitude and longitude are determined by the datum, the position of the spheroid related to the center of the Earth. The geodetic datum is customized for different parts of the world.

Common datums used in U.S.A are North American Datum 1927 (NAD27), North American Datum 1983 (NAD83), and World Geodetic System 1984 (WSG84). NAD27 uses the Clarke Ellipsoid of 1866 and was created by manual ground surveys in 1980's. The geodetic center of NAD27 is located at Meades Ranch in Kansas. However, NAD83 is based on both ground surveys and additional satellite information. The WGS84 is the most recently developed datum and framework. The origin of WGS 84 is located at the center of the Earth.

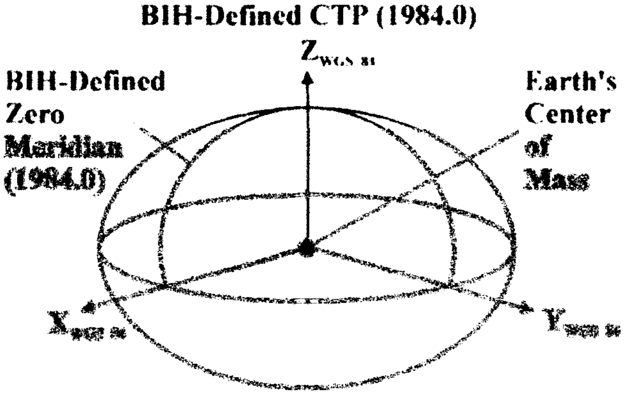


Figure 2.3 The WGS84 coordinate system definition (EUROCONTROL, 1998)

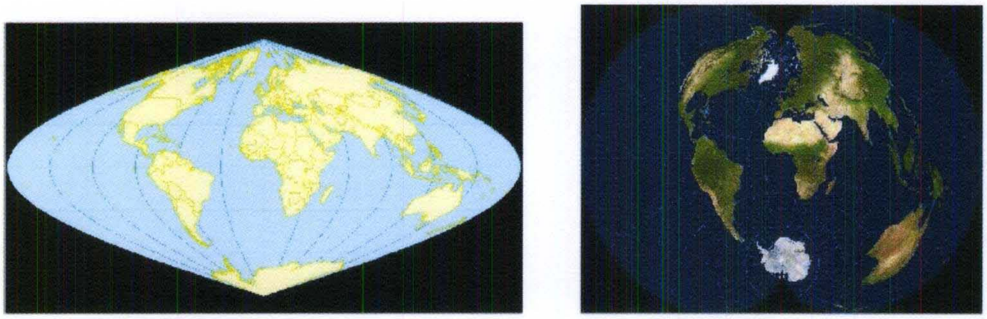
Projected coordinate systems

The advances in information technology during the last two decades have given a boost to automatic cartography and hence, to digital mapping. A hard copy analogue map can be digitized and transformed into a computer compatible data base, which can then be used for a variety of Computer Aided Design (CAD) applications in planning, civil engineering and Geographical Information Systems (GIS) (EUROCONTROL, 1998).

The flattening the Earth causes distortions based on the following properties:

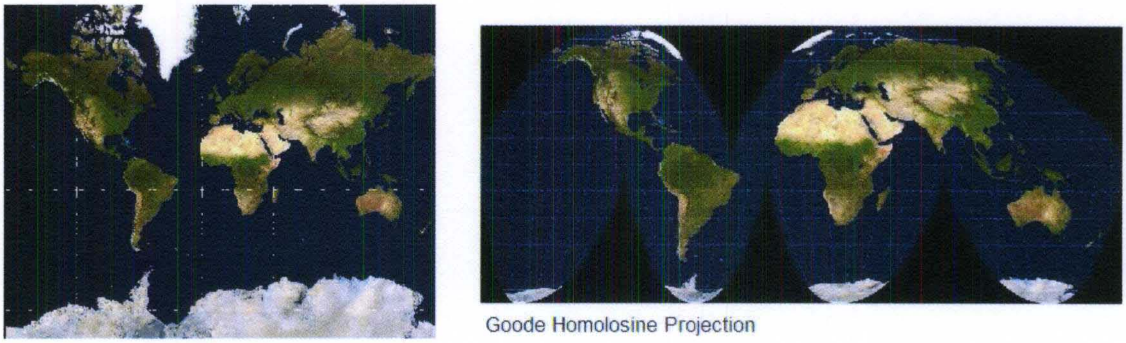
- **Shape**: conformal map projections preserve shape
- **Area**: equal area map projections preserve area
- **Distance and scale**: equidistant map projections preserve distance
- **Direction and angle:** azimuthal map projections preserve true direction

Based on these properties, several projections can be produced. Figure 2.4 introduces some of the map projections.



Sinusoidal Projection

Polyconic Projection



Goode Homolosine Projection

Mercator Projection

Figure 2.4 Different types of the map projection

The Mercator projection is a cylindrical map projection that assumes a cylinder is wrapped around the globe so that its surface touches the equator; the meridians of longitude can be projected onto the cylinder as equally spaced straight lines perpendicular to the equator (Moore, 1997). This projection draws a conformal map of the globe on a rectangular grid. In Figure 2.5, a point P on the globe is projected onto a point P', the shadow or image of P on the cylinder. When the cylinder is unwrapped, a flat map of the entire Earth can be obtained.

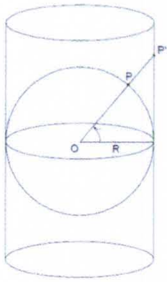


Figure 2.5 Cylindrical projection of the globe (Maor, 1998)

In order to find the relation between a point P and its image P', it must first express the location of P in terms of its longitude and latitude. Denoting the longitude, latitude and radius of P by λ, ϕ and R, the coordinates of P', x and y, can be obtained by Equations 2.1 and 2.2.

$$x = R\lambda \tag{2.1}$$

$$y = R\tan\phi \tag{2.2}$$

The most remarkable feature of the cylindrical projection is the excessive north and south stretching at high latitudes, resulting in drastic distortion of the shape of the continents. This is a consequence of the presence of tan ϕ in the Equations 2.2 (Maor, 1998).

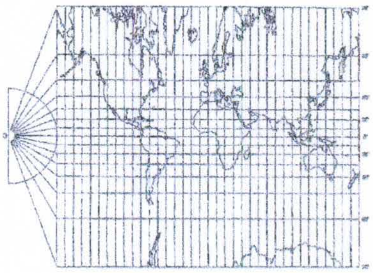


Figure 2.6 World map on a cylindrical projection (Maor, 1998)

2.2.2 Geographical Calculation

As already commented in the introduction, the system needs geographical calculations in order to get the distance measured along the surface of the Earth or bearing between two points. Common abstractions for the surface between two points are flat surface, spherical surface, and ellipsoidal surface. The system uses the great circle distance calculation which is one of spherical surface formulas.

Flat surface formulas are for calculation of the planar space. It assumes a flat Earth that has been projected on the plane. As a consequence, accuracy of these calculations is not increasingly accurate if a position is near the pole and the distance between two positions is large. The Federal Communications Commission (FCC) presents the equations for the ellipsoidal Earth projected plane (FCC, 2010). In these equations, d is the distance between two points in kilometers, longitude is λ, latitude is ø, K_1 is the number of kilometers per degree of latitude at the given middle latitude and K_2 is the number of kilometers per degree of longitude at the given middle latitude.

$$\phi_m = \frac{(\phi_1 + \phi_2)}{2} \qquad (2.3)$$

$$K_1 = 111.13209 - 0.56605\cos(2\phi_m) + 0.0012\cos(4\phi_m) \qquad (2.4)$$

$$K_2 = 111.41513\cos(\phi_m) - 0.09455\cos(3\phi_m) + 0.0012\cos(5\phi_m) \qquad (2.5)$$

$$d = \sqrt{(K_1\Delta\phi)^2 + (K_2\Delta\phi)^2} \qquad (2.6)$$

This formula shall be used to compute the distance between two reference points. However, it is valid only for distances not exceeding 475 km or 295 miles (FCC, 2010).

Spherical Surface assumes that the shape of the Earth is sphere, and it calculates an approximate distance with a possible error of 0.5 %. The Great Circle distance d between two points (λ_0, ϕ_0) and (λ_1, ϕ_1) on a spherical Earth can be derived from the spherical trigonometry cosine rule:

$$d_{x0,y0}^{x1,y1} = R_E \cdot \arccos(\sin(\varphi_0) \times \sin(\varphi_1) + \cos(\varphi_0) \times \cos(\varphi_1) \times \cos(\lambda_1 - \lambda_0)) \tag{2.7}$$

where R_E is the mean radius of the Earth, taken as 6372.7 km (Garcia, 2007).

Ellipsoidal Surface calculates the shortest distance along the surface of an ellipsoid. Bowring developed a formulation for the direct problem using a conformal projection of the ellipsoid on the sphere and it is called a Gaussian projection of a second kind. The simplicity of the system lies in that the ellipsoidal geodesic is projected to its corresponding line on a sphere thereby allowing the formulation using spherical trigonometry (Burtch, 2004).

The most accurate formula among geographical calculation is the Vincenty's formula in the ellipsoidal surface formula. The Vincenty formula is an iterative way for computing the distance between two points along the surface of the Earth. This formula was developed by Thaddeus Vincenty in 1975 and it is more accurate than the great circle distance because it is accurate to within 0.5 mm. There are two methods in this formula: direct and inverse. The direct method calculates the destination point from the given direction from the start point. The inverse method computes the distance between two points. Before introducing these two methods, the following notation is needed (Vincenty, 1975).

A	Length of major semi axes of the ellipsoid; radius at equator. 6,378,137 meters in WGS-84.
B	Length of minor semi axes of the ellipsoid; radius at the poles. 6,356,752.314 meters in WGS-84.
F	Flattening of the ellipsoid. $(a - b) / a$ = 1/298.257223563 in WGS-84.

Φ	Geodetic latitude, positive north of the equator.
U	Reduced latitude; *arctan* ((1 - *f*) *tan* φ)
Λ	Geodetic longitude.
Σ	Angular distance on the sphere
L	Difference is longitude, positive east. $\lambda_2 - \lambda_1$
A	Azimuths of the geodesic at the equator.
α_1, α_2	Azimuths of the geodesic, clockwise from north; α_2 in the direction P_1 and P_2 produced
S	Ellipsoidal distance between the two points

Table 2.1 Notation for the Vincenty formula

The direct method employs the following equations.

$$\tan U_1 = (1-f)\tan\phi_1 \tag{2.8}$$

$$\sigma_1 = \arctan\left(\frac{\tan U_1}{\cos\alpha_1}\right) \tag{2.9}$$

$$\sin\alpha = \cos U_1 \cdot \sin\alpha_1 \tag{2.10}$$

$$\cos^2\alpha = (1-\sin\alpha)(1+\sin\alpha) \tag{2.11}$$

$$u^2 = \cos^2\alpha\frac{a^2-b^2}{b^2} \tag{2.12}$$

$$A = 1+\frac{u^2}{16384}\left(4096+u^2\left(-768+u^2\left(320-175u^2\right)\right)\right) \tag{2.13}$$

$$B = \frac{u^2}{1024}\left(256+u^2\left(-128+u^2\left(74-74u^2\right)\right)\right) \tag{2.14}$$

$$2\sigma_m = 2\sigma_1 + \sigma \tag{2.15}$$

$$\Delta\sigma = B\sin\sigma\left(\cos(2\sigma_m) + \frac{1}{4}B\left(\begin{array}{l}\cos\sigma\left(-1+2\cos^2(2\sigma_m)\right) \\ -\dfrac{B\cos(2\sigma_m)(-3+4\sin^2\sigma)\left(-3+4\cos^2(2\sigma_m)\right)}{6}\end{array}\right)\right)$$

(2.16)

$$\sigma = \frac{s}{bA} + \Delta\sigma \tag{2.17}$$

Equations 2.19, 2.20, and 2.21 are iterated until the change in σ is negligible. The first approximation of σ is the first term of Equation 2.17.

$$\phi_2 = \arctan\left(\frac{\sin U_1\cos\sigma + \cos U_1\sin\sigma\cos\alpha_1}{(1-f)\sqrt{\sin^2\alpha + (\sin U_1\sin\sigma - \cos U_1\cos\sigma\cos\alpha_1)^2}}\right) \tag{2.18}$$

$$\lambda = \arctan\left(\frac{\sin\sigma\sin\alpha_1}{\cos U_1\cos\sigma - \sin U_1\sin\sigma\cos\alpha_1}\right) \tag{2.19}$$

$$C = \frac{f}{16}\cos^2\alpha\left(4 + f\left(4 - 3\cos^2\alpha\right)\right) \tag{2.20}$$

$$L = \lambda - (1-C)f\sin\alpha\left(\sigma + C\sin\sigma\left(\cos(2\sigma_m) + C\cos\sigma\left(-1+2\cos^2(2\sigma_m)\right)\right)\right) \tag{2.21}$$

$$\alpha_2 = \arctan\left(\frac{\sin\alpha}{-\sin U_1\sin\sigma + \cos U_1\cos\sigma\cos\alpha_1}\right) \tag{2.22}$$

The inverse method employs the following equations.

$$\sin\sigma = \sqrt{(\cos U_2\sin\lambda)^2 + (\cos U_1\sin U_2 - \sin U_1\cos U_2\cos\lambda)^2} \tag{2.23}$$

$$\cos\sigma = \sin U_1\sin U_2 + \cos U_1\cos U_2\cos\lambda \tag{2.24}$$

$$\sigma = \arctan\frac{\sin\sigma}{\cos\sigma} \tag{2.25}$$

$$\sin\alpha = \frac{\cos U_1\cos U_2\sin\lambda}{\sin\sigma} \tag{2.26}$$

$$\cos^2\alpha = 1 - \sin^2\alpha \tag{2.27}$$

$$\cos(2\sigma_m) = \cos\sigma - \frac{2\sin U_1 \sin U_2}{\cos^2\alpha} \tag{2.28}$$

$$C = \frac{f}{16}\cos^2\alpha\left(4 + f(4 - 3\cos^2\alpha)\right) \tag{2.29}$$

$$\lambda = L + (1 - C)f\sin\alpha\left(\sigma + C\sin\sigma\left(\cos(2\sigma_m) + C\cos\sigma(-1 + 2\cos^2(2\sigma_m))\right)\right) \tag{2.30}$$

The geodetic longitude λ is obtained by Equations 2.29 and 2.30. This procedure is iterated starting with Equation 2.23 until the change in λ is negligible. The distance *s* is calculated by:

$$s = bA(\sigma - \Delta\sigma) \tag{2.31}$$

where Δσ comes from Equation 2.15.

$$\tan\alpha_1 = \frac{\cos U_2 \sin\lambda}{\cos U_1 \sin U_2 - \sin U_1 \cos U_2 \cos\lambda} \tag{2.32}$$

$$\tan\alpha_2 = \frac{\cos U_1 \sin\lambda}{-\sin U_1 \cos U_2 + \cos U_1 \sin U_2 \cos\lambda} \tag{2.33}$$

The inverse method may give no solution over a line between two nearly antipodal points. This will occur when λ, as computed by Equation 2.21, is greater than π in absolute value.

2.3 Visualization

Visualization is the creation of information to show the important matters and is used as an effective way of communication. For instance, from cave paintings, researchers may learn the living style of the ancients. People can read the revolutionary mind of Leonardo da Vinci because of his drawings. Children are able to build toys by following the instructions with pictures. Visualization is all over the places in our lives.

Nowadays, visualization is enhanced with new technologies in science, medicine, engineering, and so on. It is a great event after the industrial revolution. Especially, since the field of computer graphics was invented, visualization techniques have been introduced in all the fields.

Information Visualization

Information visualization is generally applied to the visual representation of large-scale collections of non-numerical information, such as files and lines of code in software system, library and bibliographic databases, networks of relations on the internet, and so forth (Eick, 1994). The term *Information Visualization* was coined by Dr. Jock Mackinlay.

Scientific Visualization

Scientific Visualization is primarily concerned with the visualization of multi-dimension phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth (Friendly, 2009).

Data Visualization

Data visualization is the science of visual representation of data, defined as information which has been abstracted in some schematic form, including attributes or variables for the units of information (Friendly, 2009).

2.4 3D Computer Graphics

Computer graphics is a process of producing pictures or images using computers. By this definition, computer graphics is one of the visualization techniques. In our world, most visualization is made by computer graphics. Many areas, such as art, science, education, engineering, architect, and so on, are helped by computer graphics for their rapid growth.

As techniques of computer graphics improve, a new way of visualization is introduced. It is a three dimensional representation of geometric data called 3D computer graphics. 3D computer graphics is now so realistic that it overcomes the limitation of 2D images. The basic process of 3D computer graphics is modeling, layout, and rendering. Modeling is the process of building the shape of an object. Layout is the step to set up the location and size of the model within a scene. Rendering is the process of transforming 3D scene data into something that is visible on display devices such as a screen or printed paper. It creates an image either by special visual effects such as ray tracing, global illumination algorithms, shadows, and so on, to get photorealistic images, or by adding a unique visual quality in non-photorealistic rendering for artwork or illustrations. In the next chapter, 3D computer graphics in Google Earth are discussed.

CHAPTER 3

FLIGHT VISUALIZATION

3.1 Introduction

This chapter introduces some tools and a programming language that are used to build the system. An application has been built using CodeGear RAD Studio 2009 (Embarcadero). It includes Delphi and C++ Builder 2009. With this programming language, this application is able to be created rapidly. SketchUP 8 (Google SketchUp) is used to build the 3D airplane model. SketchUP is a 3D software tool that combines a tool-set with an intelligent drawing system. With SketchUP, the model can be created and modified and convert to a Collada model. The file that stores the Collada model has the extension "dae", which is recognized by Google Earth. Google Earth allows us to browse the world through a virtual globe, and to view satellite imagery, maps, and terrain. To get the best results from the system, Google Earth version 5.2.1.1547 or higher should be installed. These software tools are discussed in details in the following sections.

3.2 Flight Visualization

3.2.1 CodeGear RAD Studio 2009

Integrated development environment (IDE) for this project is CodeGear RAD Studio 2009, which includes C++ builder and Delphi 2009. Delphi is a software development environment for Microsoft Windows and web applications. It is an object oriented language based on Pascal, and it is usually used for the development of desktop and enterprise database applications. However, it is not limited to database applications; it can be used as a general-purpose software development tool suitable for most software projects. Web applications can also be built with CodeGear RAD Studio 2009 if several libraries are included. The language is suitable for Rapid Application Development (RAD) and comes with an integrated IDE. All Delphi products are shipped with a large framework called Visual Component Library (VCL), which includes most of its source code.

The first version of Delphi was released in 1995, and it became a standard for visual programming tool for the 16-bit Windows 3.1. Then, Borland released a new version of Delphi every year. Now, the newest version of Delphi is called XE. According to Tim Del Chiaro (Chiaro, 2010), the following features are added.

- Heterogeneous database support where the programmer gets a consistent experience when working with multiple databases and the programmer does not have to pay an additional cost per platform for database support.
- XE products are ToolCloud-enabled. The ToolCloud license management and provisioning system is a great solution for companies and organizations that want to streamline license management.
- Embarcadero XE products offer an easy upgrade to Embarcadero All-Access XE.
- XE can build applications that require speed-visualizing data, controlling hardware in real time, manipulating 3D objects, financial modeling, gaming, imaging, medical equipment, point of sale systems, and more.

The disadvantage of Delphi is that it is not a cross platform IDE. Although XE includes Delphi Prism, which is used for developing .NET applications, Delhi can only work under a Windows platform.

3.2.2 Google SketchUP

SketchUP is a software tool that combines a tool-set with an intelligent drawing system for modeling of 3D objects. With SketchUP, a model can be created and modified and converted to a Collada model, whose extension is .dae, in order to be displayed in Google Earth.

On March 14, 2006, Google acquired @Last Software because of their work in developing a plugin for Google Earth. One year later, Google released SketchUP 6. However, Google had to release updates several times to fix a lot of bugs. In 2010, an improved version of SketchUP was released. SketchUP 8 now supports more accurate terrain, matches photo improvement, provides numerous models from 3D Warehouse (http://sketchup.google.com/3dwarehouse/), and exports 2D vector images and 3D objects such as 3DS, OBJ, XSI, FBX, VRML and DAE.

For the present system, as mentioned above, Collada 3D objects are used. Collada stands for COLLAborative Design Activity and it is an open standard XML schema to exchange digital assets among multiple graphics programs including Google Earth. It was originally developed by Sony and now owned by Autodesk.

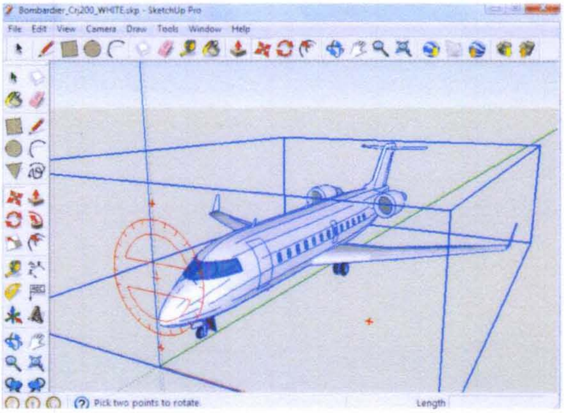

Bombardier_Crj200_WHITE.skp - SketchUp Pro
File
Edit
View
Camera
Draw
Tools
Window
Help
Pick two points to rotate
Length

Figure 3.1 Screen shot of SketchUP

3.2.3 Google Earth

According to the Google Earth website, Google Earth allows users to travel the world through a virtual globe and to view satellite imagery, maps, terrain, 3D buildings, and much more. With the rich geographical contents of Google Earth, users are able to experience a more realistic view of the world. This software was originally developed by Keyhole, Inc., which was acquired by Google in 2004. Keyhole Viewer, the original name of Google Earth, was released on June 11, 2001. Since 2004, Google has released new versions of Google Earth every year or more often.

This internet-based virtual globe is utilized in the present work because it gives a user the ability to virtually fly over any location with a realistic view. The generated KML file from the system contains optimized scripts that allow animate flights in the virtual world. Some sample KML code is in Appendix C.

3.2.3.1 DirectX versus OpenGL

Google Earth supports two graphic modes, OpenGL and DirectX. According to the documentation of Google Earth, if a user experiences strange graphics issues, he or she should try running Google Earth in each mode to see if one works better for the user's system. The recommended and default rendering mode is OpenGL because it is the rendering software for most graphics cards that do not use system resources. If a graphics card uses system memory, such as RAM, DirectX may produce better performance of the 3D effects.

Direct3D is part of the Microsoft DirectX API. Direct3D only works in Microsoft windows; it does not work cross-platform. One of the main benefits is its low level graphical API that enables a user to manipulate visual 3D models. Because Direct3D can take advantage of hardware acceleration, the user may experience faster graphics rendering.

Both OpenGL and DirectX are well known APIs in the community of developers. OpenGL seems to be used more in specialized areas such as 3D film animations, scientific visualizations, and geographical simulations, whereas DirectX is used by the video game development industries. In the past, most graphics software was written using Integrated Raster Imaging System Graphics Library (IRIS GL). It was widely used because of its performance, and its low system memory and CPU power requirements. Later, many functionalities of IRIS GL were incorporated in OpenGL. OpenGL aims for the professional graphics software. For instance, OpenGL Volumizer is a high-level volume rendering API for the energy, manufacturing, medical, and science markets. It is designed for interactive, high quality, scalable visualization of large volumetric data sets such as diagnostic CT, MRI, and PET scans, seismic data, and other unstructured meshes common in computational fluid dynamics analysis. OpenGL provides a high-level interface to hardware to allow application writers and researchers to visualize multiple gigabytes of volumetric data (OpenGL, 2004). Figure 3.2 shows volumetric data with transparency and color for different organs. The size of the visible human body data set is 6.77 GB.

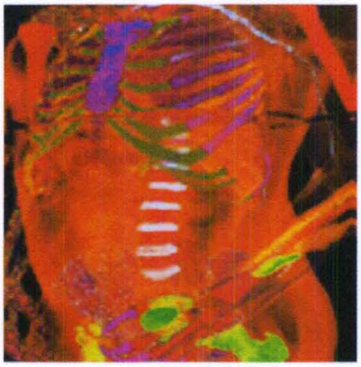


Figure 3.2 OpenGL Volumizer

On the contrary, DirectX is targeting for gaming developers by designing low-level, high performance games for end users who own PC's with affordable consumer priced graphics hardware. DirectX contains a set of tools such as DirectInput, DirectPlay, and DirectSound along with Direct3D. In fact, gaming not only shows 3D graphics but also plays sound during interactions with other network users.

3.2.3.2 Terrains and Accuracy

One of the most important features of Google Earth is its high resolution and realistic terrains. Using Google Earth, a user can visit famous cities with 3D buildings and can fly over the mountains that are really too hard to climb in one's lifetime. The following figure shows an example of terrains and 3D buildings in Google Earth.

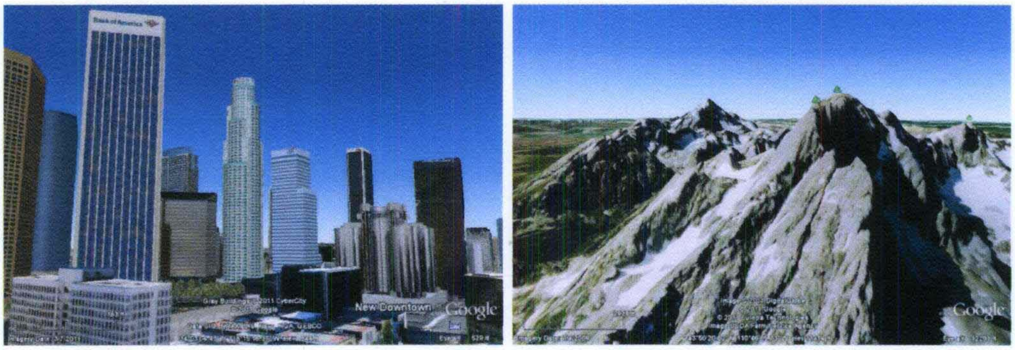


Figure 3.3 Google Earth's 3D buildings and Terrains

Since Version 4 was released, Google Earth had improved matching between the satellite and aerial images and the terrain. As of August 9, 2005, Google Earth typically covered the entire globe with satellite imagery at 15m resolution. Google was focused on large, US metropolitan areas initially and has expanded coverage to major international cities (Google Earth Coverage, 2005). Excluding Alaska and Hawaii, the continental US is covered with approximately 1m resolution. In this version, a user can select a new option called "Terrain Quality." If the computer system has a lot of memory and a good graphics card, a user may try to display with the maximum quality; otherwise he or she should slow down the update rate of the graphics card. One thing to keep in mind is that this option is only for enhancing 3D terrain visualization, but not the quality of imagery.

Google Earth uses Digital Elevation Model (DEM), data for the terrain collected by Shuttle Radar Topography Mission (SRTM). DEM is a digital model or 3D representation of cartographic information of terrain surfaces in a raster form. The U.S. Geological Survey, USGS, has been designated as a lead Federal agency for the collection and distribution of digital cartographic data (USGS, 1998). Because of rapidly changing technologies in the mapping industries, these DEM standards cover a broad range of collection systems and different accuracy levels. The data file contains the elevation of the surface at fixed grid intervals over the Earth, and these intervals in the grid are referenced by a geographical coordinate system. Note that the latitude and the longitude intervals may have different lengths. The following figure shows the basic structure of the DEM standards.

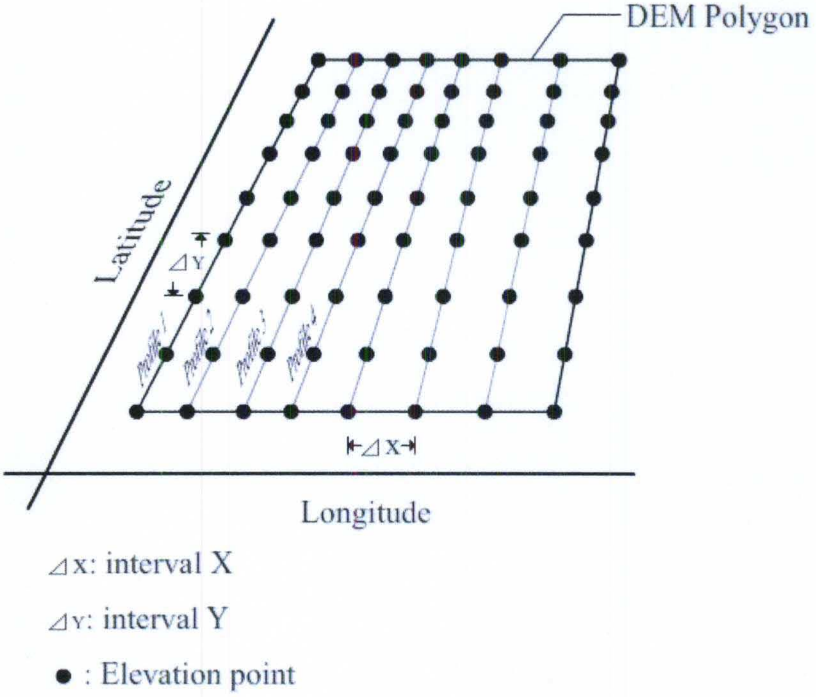


Figure 3.4 Standard of Digital Evaluation Model

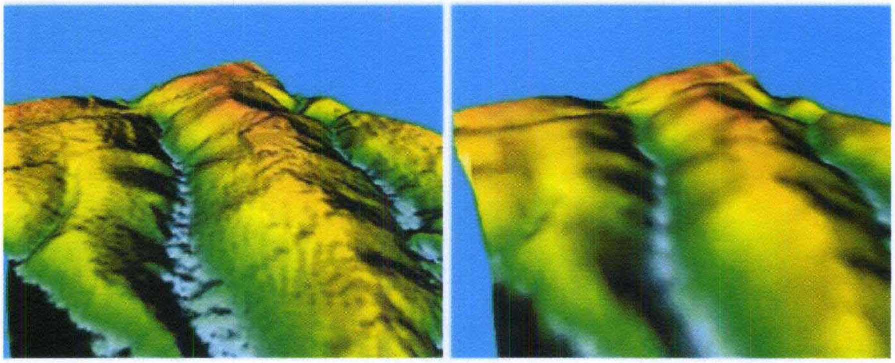
Each elevation point in each profile stores height information in a raster form. Based on the interval lengths, the USGS produced five primary types of DEM data.

- 7.5 minute DEM: up to 30 meter square grid spacing, cast on the Universal Transverse Mercator (UTM) projection. The horizontal grid spacing allows for integers between one and 30 meters.
- 30 minute DEM: 2 by 2 arc second data spacing. Two 30 minute DEM's provide the same coverage as a standard USGS 30 by 60 minute quadrangles.

- 1 degree DEM: 3 by 3 arc second data spacing. The ground spacing between grid points is 3 arc seconds, which is roughly 90 meters, dependent on the latitude.
- 7.5 minute Alaska DEM: 1 by 2 arc second data spacing, latitude by longitude. It provides coverage similar to a 7.5 minute DEM, except that the longitudinal cell limits vary from 10 minutes at the southernmost latitude of Alaska to 18 minutes at the northernmost latitude limits of Alaska.
- 15 minute Alaska DEM: 2 by 3 arc second data spacing, latitude by longitude. Its coverage is 15 minutes of latitude by 20 minutes of longitude at the southernmost latitude of Alaska, to 36 minutes of longitude at the northernmost latitude limit of Alaska.

A DEM file consists of a series of three record types: A, B, and C. The type A record contains header information, which includes name, boundaries, units of measurement, minimum and maximum elevations, the number of B records, and projection parameters. Each type B record is made up of data from one-dimensional arrays called profiles, one per line of elevation data. The type C record contains statistics on the accuracy of the data in the file. The logical record size of file is 1,024 bytes and more than one record is usually required to store a single type B record.

According to the fact sheet of USGS, the accuracy of DEM data depends on the source and resolution of the data samples. DEM data accuracy is derived by comparing linear interpolation elevations in the DEM with corresponding map location elevations and computing the statistical standard deviation or root mean square error (RMSE). The RMSE is used to describe the DEM accuracy. For 7.5 minute DEM's derived from a photogrammetric source, 90 percent of the elevation data have a vertical accuracy of 7 meter RMSE or better, and 10 percent are in the 8 to 15 meter range. The 1 degree DEM data have an absolute accuracy of 130 meters horizontally and 30 meters vertically (USGS, 2000). Benker *et al.* (Benker, 2011) compared virtually traced positions against high precision (< 1m) field measurements along three stratigraphic unconformity sub-sections in the Big Bend region to determine the current positional accuracy for the Google Earth terrain model. A horizontal position accuracy of 2.64m RMSE was determined for the Google Earth terrain model with the mean offset distance being 6.95m. A vertical position accuracy of 1.63m RMSE with the mean offset distance of 2.66m was also calculated for the terrain model. Results suggest data extracted from the Google Earth terrain model could plausibly be used in future studies. However, the authors urge caution in using Google Earth data due to limited information disclosures by developers (Benker, 2011).

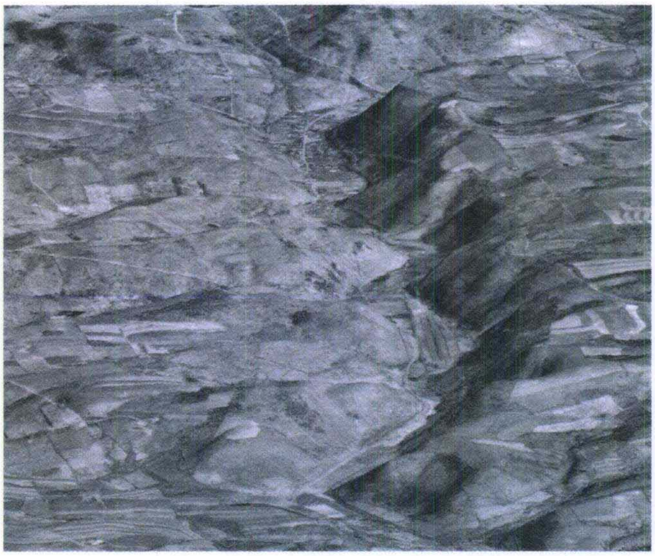


1m DEM *7m DEM*

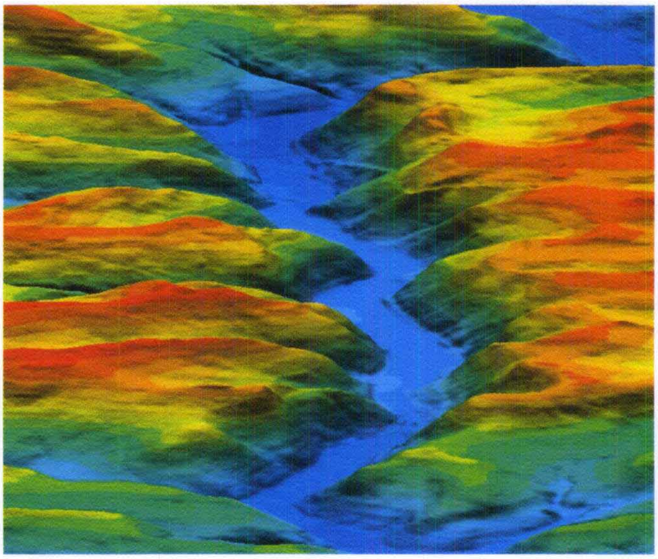
Figure 3.5 Example of 1m and 7m DEM

(http://www.spatialenergy.com/products_digital.html)

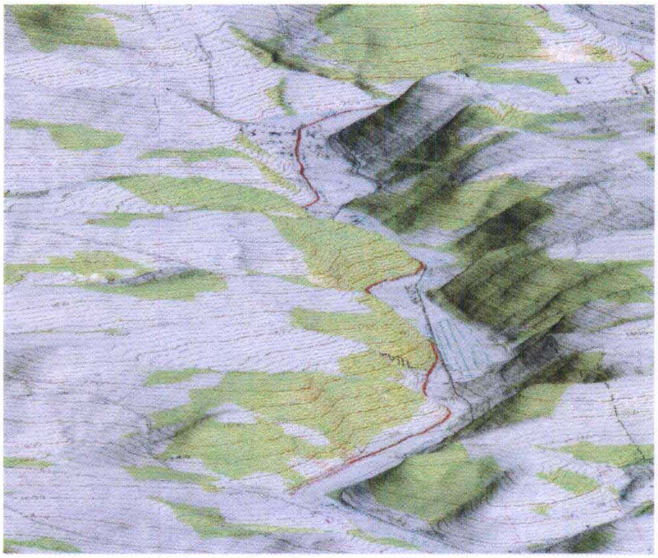
In the left image in Figure 3.5, the spacing is 5 meters with the vertical accuracy in the open flat terrain of 1m (RMSE). In the right image, the spacing is 30 meters with the vertical accuracy in the open flat terrain of 7m (RMSE). The more spacing is used in DEM, the more image is blurred. The following figures are also examples of DEM.



3D DOQ 1m 10m DEM; Pennsylvania, USA



3D Pseudo Color 10m DEM; Pennsylvania, USA

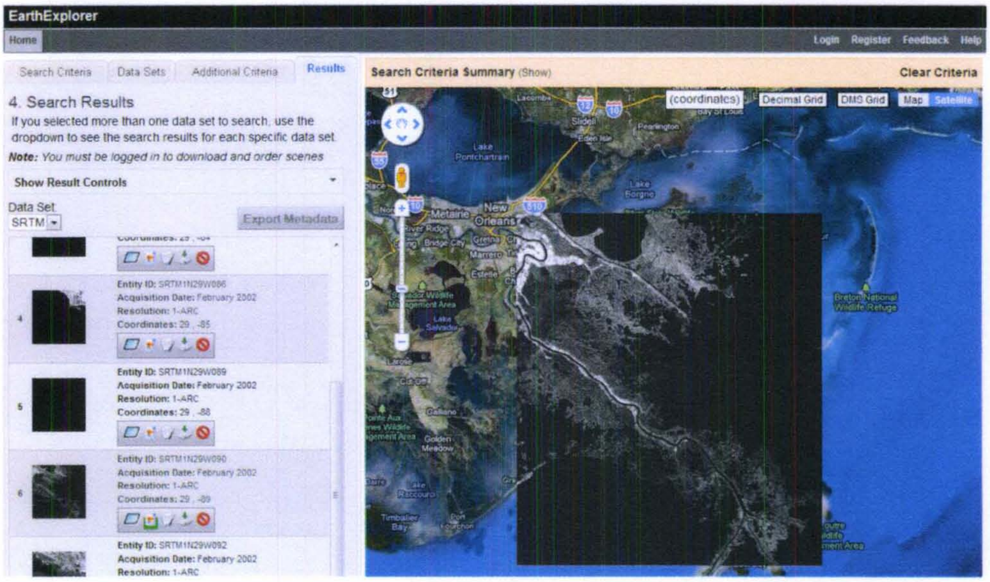


3D Topomap 10m DEM; Pennsylvania, USA

Figure 3.6 DEM at Pennsylvania, USA

(http://www.satimagingcorp.com/gallery-dem.html)

Google Earth's DEM data are mostly based on images from NASA's Shuttle Radar Topography Mission (SRTM). SRTM obtained elevation data on a near-global scale to generate the most complete high resolution digital topographic database of the Earth. SRTM consisted of a specially modified radar system that flew onboard the Space Shuttle Endeavour during an 11-day mission in February of 2000 (NASA, 2009). Endeavour orbited Earth 16 times each day during the 11-day mission, for a total of 176 orbits. SRTM successfully collected radar data over 80% of the Earth's land surface between latitude 60 degrees north and 56 degrees south with data points posted every arc second, which is approximately 30 meters. EarthExplorer can be used to search, preview, and download the finished grade SRTM elevation data. The collection is located under the Digital Elevation category (USGS, 2011).



4. Search Results

If you selected more than one data set to search, use the dropdown to see the search results for each specific data set

Note: *You must be logged in to download and order scenes*

Data Set: SRTM

4
Entity ID: SRTM1N29W086
Acquisition Date: February 2002
Resolution: 1-ARC
Coordinates: 29 , -85

5
Entity ID: SRTM1N29W089
Acquisition Date: February 2002
Resolution: 1-ARC
Coordinates: 29 , -88

6
Entity ID: SRTM1N29W090
Acquisition Date: February 2002
Resolution: 1-ARC
Coordinates: 29 , -89

Entity ID: SRTM1N29W092
Acquisition Date: February 2002
Resolution: 1-ARC

Figure 3.7 EarthExplorer: New Orleans region SRTM data

(http://edcsns17.cr.usgs.gov/NewEarthExplorer)

3.2.4 System Architecture

The system architecture for Flight Visualization is shown in Figure 3.8.

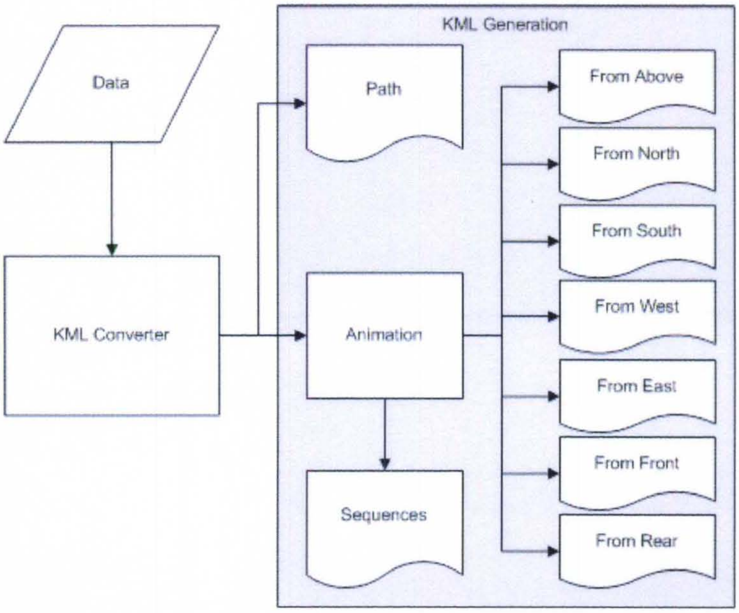
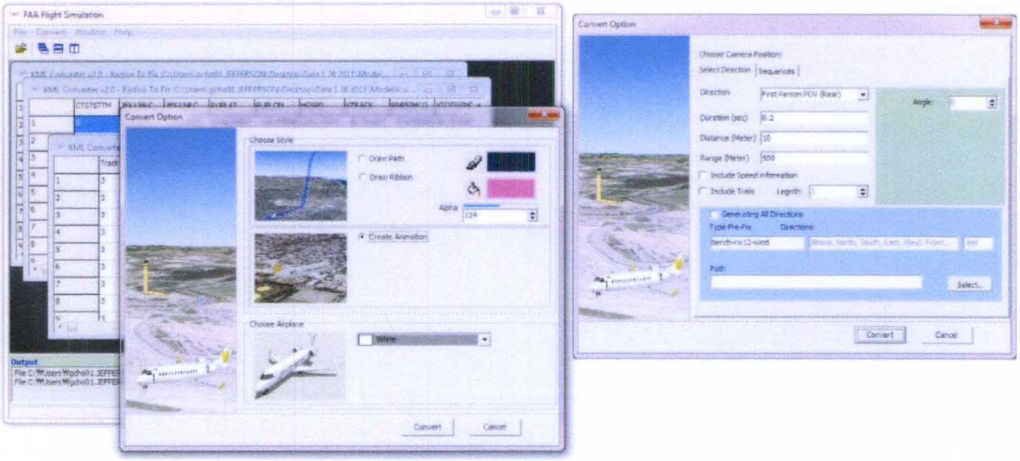


Figure 3.8 System Architecture

Basically, the system loads data and gets properties of the camera from a user. As in Figure 3.8, it loads the raw data file first. With this data, the system calculates the position, heading, and angle of camera by using geographical functions. After the user sets all properties, it is ready to generate optimized KML code.



Alpha 124

● Create Animation

Choose Airplane: White

Direction: First Person POV (Rear)

Duration (sec): 0.2

Distance (Meter): 10

Range (Meter): 500

☐ Include Speed Information

☐ Include Trails

○ Generating All Directions

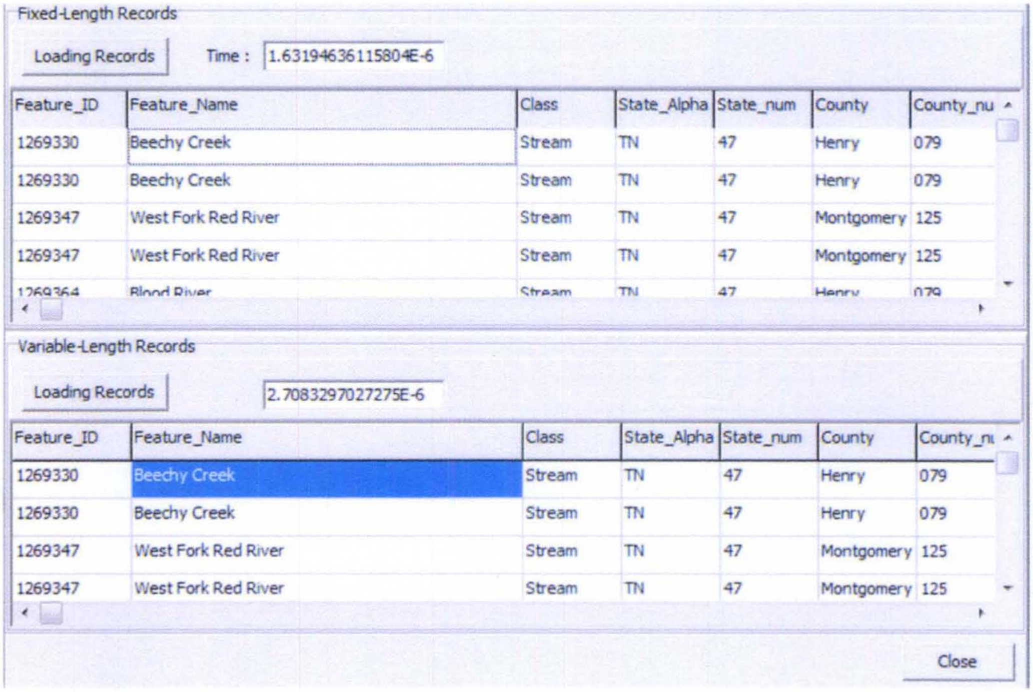
Type Pre-Fix: bench-rw12-wind

Directions: Above, North, South, East, West, Front...

Figure 3.9 Screen shot of an application

3.2.4.1 Raw Data

The data used in this system is tab delimited and variable-length data type, where fields can vary in length. Usually, for large amounts of data, a fixed-length format is used because of its high loading speed. When loading fixed-length data, the algorithm gets fields based on the lengths by using the sub-string function with specific starting and ending locations. In case of the variable-length data type, the algorithm searches the next delimiter to get fields, and thus it takes more time than the fixed-length data type. The following experiment is conducted in order to test loading speeds between fixed-length and variable-length data types. The data used for this experiment are from U.S. Geological Survey at http://geonames.usgs.gov/domestic/download_data.htm. The data are extracted from the geographic names information system. They contain primary feature attributes of Kentucky. Both of the KY_DECI_Fixed.txt and KY_DECI_Variable.txt files contains 10,000 records and 9 attributes. All attributes except the second one are 20 bytes in length for fixed-length data; the second attribute, Feature_Name, is 70 bytes in length. The vertical bar "|" character is used as a delimiter for variable-length data.



1.63194636115804E-6

Feature_ID	Feature_Name	Class	State_Alpha	State_num	County	County_nu
1269330	Beechy Creek	Stream	TN	47	Henry	079
1269330	Beechy Creek	Stream	TN	47	Henry	079
1269347	West Fork Red River	Stream	TN	47	Montgomery	125
1269347	West Fork Red River	Stream	TN	47	Montgomery	125
1269364	Blood River	Stream	TN	47	Henry	079

2.7083297027275E-6

Feature_ID	Feature_Name	Class	State_Alpha	State_num	County	County_nı
1269330	Beechy Creek	Stream	TN	47	Henry	079
1269330	Beechy Creek	Stream	TN	47	Henry	079
1269347	West Fork Red River	Stream	TN	47	Montgomery	125
1269347	West Fork Red River	Stream	TN	47	Montgomery	125

Figure 3.10 Fixed-length versus Variable-length data type

As shown in Figure 3.10 above, loading fixed-length data is about 1.7 times faster than loading variable-length data. Although fixed-length is faster, the system uses variable-length data for two reasons. The first reason is its compact size. In this experiment, the size of the file KY_DECI_Fixed.txt is 2,149 KB whereas it is 691KB for the file KY_DECI_Variable.txt. If the format for the whole flight data is fixed-length, its size will be enormous. The other reason is that the system does not have to read the whole data at once. The system loads the first record for setting the initial position and gets the next two records. By calculating the distances among three points, our system tries to remove outlier for the best animation. The system keeps repeating the above steps till the end of the flight record.

The raw flight data are typically collected at 1Hz. The fields in the raw data that will be used in the flight visualization include latitude, longitude, ground speed, magnetic heading, wind direction, aircraft heading (corrected), pressure altitude, roll angle, and pitch angle. Different sources of raw data may have different fields. Some sample data are in Appendix B. The system needs at least four mandatory fields; they are the number of fields, latitude, longitude, and altitude. The number of fields is needed to avoid out of index error, and the other three fields are used to define the location of the aircraft. If heading is not provided, the system will estimate the heading by deriving it from the latitude and longitude of the successive positions, which will not incorporate the effects of the wind on the heading angle. The range of heading in the raw file usually is between 0 and 360 degrees. However, the range in Google Earth is between -180 and 180 degrees. Thus heading must be converted to the angle range of Google Earth during loading.

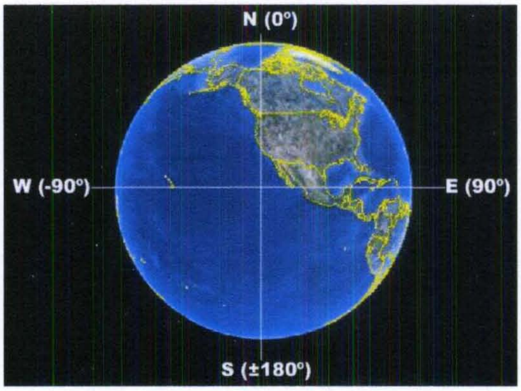
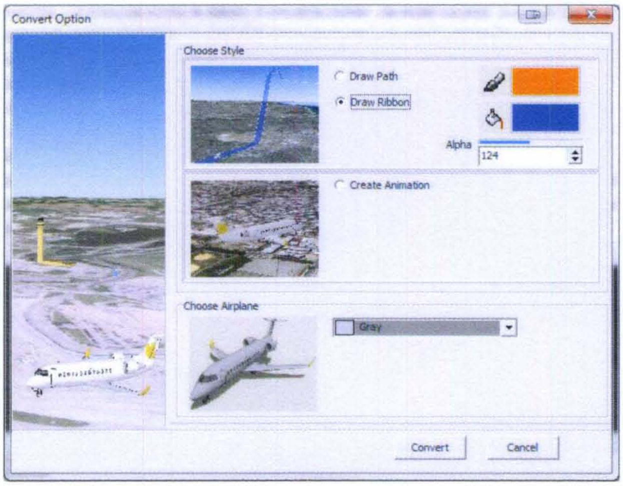


Figure 3.11 Orientation in Google Earth

3.2.4.2 Convert Options

There are mainly two options for the conversion; one option is to create an animation, and the other option is to draw a flight path or a ribbon. A user can select these options with a simple GUI. There are also nine types of aircraft models to be selected.



○ Draw Path
● Draw Ribbon
Alpha 124
○ Create Animation
Gray

Figure 3.12 Convert Option Screen

Path and Ribbon

For the flight path option, it has only one parameter, the color of the path. It generates a KML file that draws the fight path on the surface of the Earth, and it animates the camera from above and along the path. For the ribbon option, it needs the color of the ribbon, and the transparency of the color. The color can be defined by a hexadecimal number (between 0x00 and 0xff) in the order of transparency, blue, green, and red. For example, 0xff00ff00 is opaque green. After setting the color of the ribbon, the system draws a red line on the left side and a green line on the right side of the aircraft as navigation lights that are used to signal the position and status of the aircraft. Commonly, the navigation lighting system helps two aircrafts on a collision course determine who has right of way.

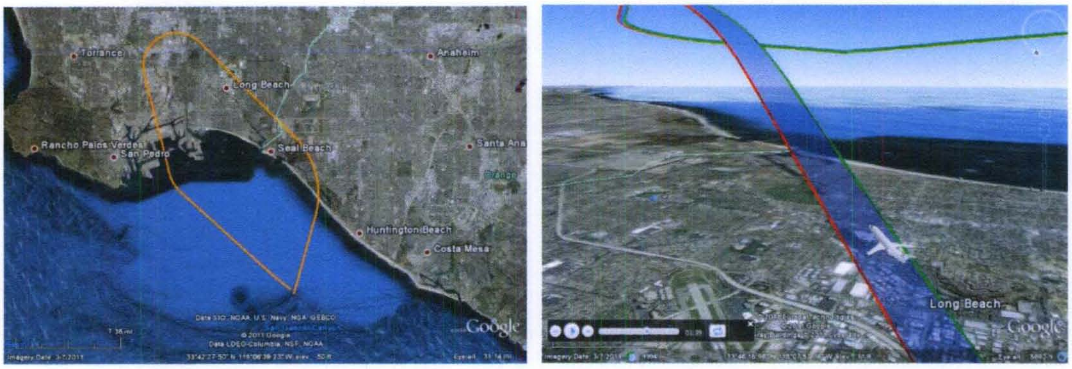


Figure 3.13 Path and Ribbon Options

In KML, paths and ribbons are objects that must have identifiers. For the path option, the system passes an array of aircraft (center of mass) positions in the format of longitude, latitude, and optionally altitude by using the `Linestring` element under the `Geometry` class. This implies that the system does not need to calculate the aircraft positions at all. However, for the ribbon option, the system needs to calculate the positions of wing tips using the aircraft (center of mass) position, heading, roll angle, and pitch angle. The system uses geometric functions and great earth distances in the calculation because KML does not support these kinds of methods.

The first step is to find the locations of wing tips using the great circle calculation. In the system, it returns longitude, latitude, altitude, heading, roll angle, and pitch angle of aircraft's nose.

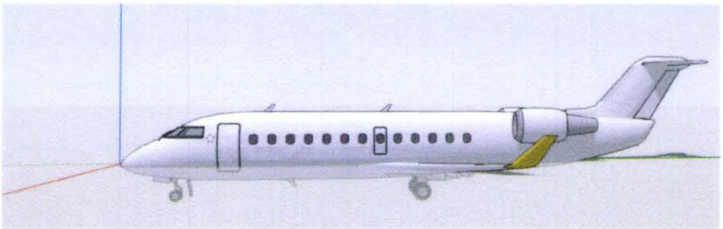


Figure 3.14 Origin of the aircraft from data

Based on the pitch angle, the calculation needs to change the origin of the coordinate system to the wingtip. In Figure 3.15, let us assume that *d* is the distance from the nose to the mid-point of wingtips, θ_p is the pitch angle, and h is the height based on the pitch angle.

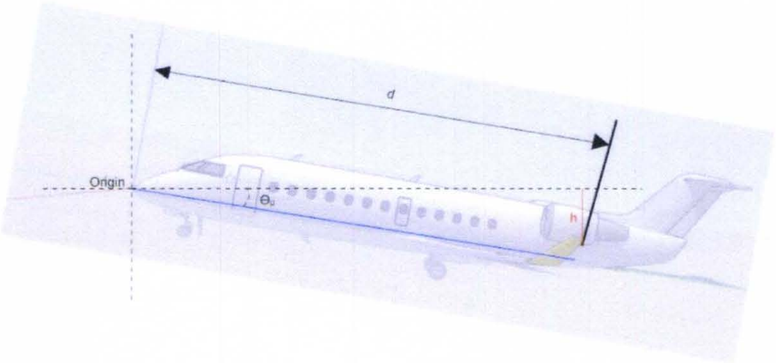


Figure 3.15 Changing origin based on distance and pitch angle

Now, the system uses the Great Circle calculation and trigonometry functions. Equation 3.13 and 3.14 are Great Circle calculations to get longitude and latitude of the new origin and Equations 3.15, 3.16, and 3.17 are used to set the altitude of the new origin.

Great Circle is also known as *Haversine* formula. Let us consider the Earth as a sphere. Then the shortest path between two points is calculated by the Great Circle distance, which corresponds to an arc linking two points on the sphere like Figure 3.16

(Rodrigue 2011). Because of the distortions caused by projection of the globe on a flat sheet of paper, a straight line on a map is not necessarily the shortest distance.

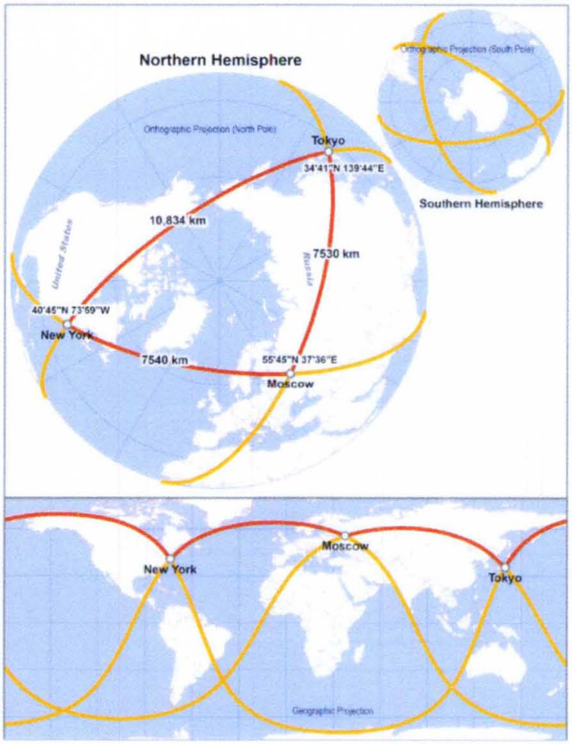


Figure 3.16 The shortest path on Earth

The *Versine* or versed sine of angle θ is 1 – cos(θ), and the *Haversine* is half the versine, or (1-cos(θ))/2. From the meaning of *Haversine*, the formula is Equation 3.1.

$$hav(\theta) = \frac{(1-\cos(\theta))}{2} = \sin^2\left(\frac{\theta}{2}\right) \tag{3.1}$$

In order to prove the formula, let us draw a circle as in Figure 3.17. *O* is the center of the circle, A and B are on the circle and θ_{AB} is the angle of AOB. The angle of AOS is $\theta_{AB}/2$.

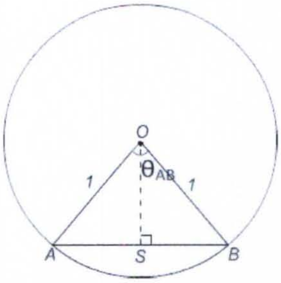


Figure 3.17 Circle of radius 1

The length of AS is $\sin(\theta_{AB}/2)$. So, the length of AB is AS+SB, that is $2\times\sin(\theta_{AB}/2)$.

In the case of the 3D space, as in Figure 3.18, A and B are opposite vertices of an isosceles trapezoid, ACBD with additional vertices C and D. Points E and F are the points where the longitude lines longitude A and longitude B meet the equator, respectively.

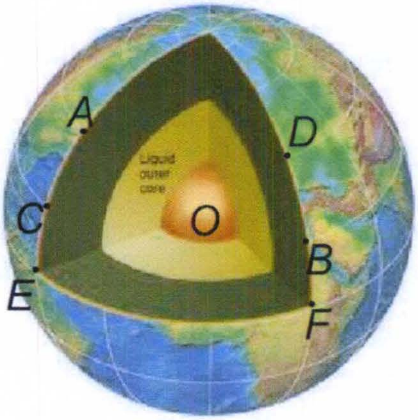


Figure 3.18 Sphere of radius 1

Then, the angle of AOC is the difference between latitude A, Lat_A and latitude C, Lat_C. Therefore, the length of AC is $2\times\sin((Lat_A - Lat_C)/2)$ or $2\times\sin((Lat_A - Lat_B)/2)$ because latitude C is equal to latitude B. The length of BD is the same as AC.

$$AC = 2\times\sin\left(\frac{(Lat_A - Lat_C)}{2}\right) = 2\times\sin\left(\frac{(Lat_A - Lat_B)}{2}\right) \tag{3.2}$$

$$BD = 2\times\sin\left(\frac{(Lat_D - Lat_B)}{2}\right) = 2\times\sin\left(\frac{(Lat_A - Lat_B)}{2}\right) \tag{3.3}$$

The length of EF is 2×sin((Lon_E - Lon_F)/2) by using the same formula.

Points A and D are on a circle of constant latitude Lat_A. In order to get the radius of this circle, consider Figure 3.19. Point G is perpendicular from A to OE.

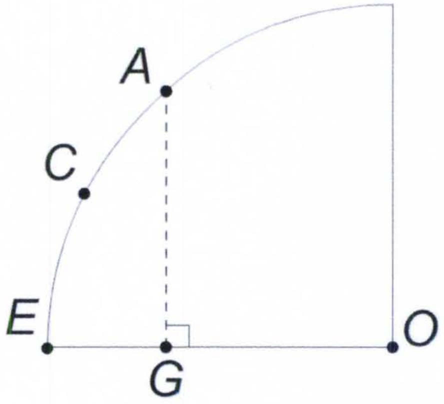


Figure 3.19 Section of the Point A, C, E, G, and O

The angle of AOE is Lat_A. So, the length of OG is cos(Lat_A) and it is also the radius of the circle of constant latitude Lat_A. Therefore, the length of AD is 2×sin((Lon_E - Lon_F)/2) ×cos(Lat_A) and the length of CB is 2×sin((Lon_E - Lon_F)/2) ×cos(Lat_B). Longitude E is equal to longitude A, and longitude F is equal to longitude B.

$$AD = 2\times\sin\left(\frac{(Lon_E - Lon_F)}{2}\right)\times\cos(Lat_A) = 2\times\sin\left(\frac{(Lon_A - Lon_B)}{2}\right)\times\cos(Lat_A) \tag{3.4}$$

$$CB = 2\times\sin\left(\frac{(Lon_E - Lon_F)}{2}\right)\times\cos(Lat_B) = 2\times\sin\left(\frac{(Lon_A - Lon_B)}{2}\right)\times\cos(Lat_B) \tag{3.5}$$

In Figure 3.20, let us see the trapezoid of ABCD in 2D again. Point H is perpendicular from A to CB.

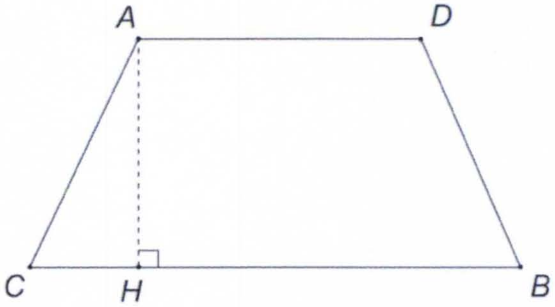


Figure 3.20 Trapezoid of ABCD

The length of CH is (CB-AD)/2, and the length of HB is (CB+AD)/2. By the Pythagorean Theorem, we can get the following equations.

$$(AH)^2 = (AC)^2 - (CH)^2 = (AC)^2 - \frac{(CB-AD)^2}{4} \tag{3.6}$$

$$(AB)^2 = (AH)^2 + (HB)^2 = (AC)^2 - \frac{(CB-AD)^2}{4} + \frac{(CB+AD)^2}{4} = (AC)^2 + (CB)\times(AD) \tag{3.7}$$

Now, we can substitute Equation 3.2, 3.4, and 3.5 to Equation 3.7.

$$(AB)^2 = 4\times\left(\sin^2\left(\frac{(Lat_A - Lat_B)}{2}\right) + \cos(Lat_A)\times\cos(Lat_B)\times\sin^2\left(\frac{(Lon_A - Lon_B)}{2}\right)\right) \tag{3.8}$$

Let *a* be the square of half the chord length AB, then *a* is

$$\begin{aligned} a &= \left(\frac{(AB)}{2}\right)^2 \\ &= \sin^2\left(\frac{(Lat_A - Lat_B)}{2}\right) + \cos(Lat_A)\times\cos(Lat_B)\times\sin^2\left(\frac{(Lon_A - Lon_B)}{2}\right) \end{aligned} \tag{3.9}$$

Now, let us go back to Figure 3.17 and find the angle AOB. The length of AS is $\sqrt{a}$, and we need to find the length of OS using Equation 3.10.

$$OS = \sqrt{(OA)^2 - (AS)^2} = \sqrt{1-a} \tag{3.10}$$

So, $\tan(\theta_{AB}/2)$ is AS/OS. Let c be the angle of AOB, then

$$c = 2 \times \arctan\left(\frac{AS}{OS}\right) = 2 \times \arctan\left(\frac{\sqrt{a}}{1-\sqrt{a}}\right) \tag{3.11}$$

Finally, we can get the distance, d, using the following equation.

$$d = R \times c \tag{3.12}$$

R is the Earth radius (Rick, 2011). Since the Earth is not strictly a sphere, there are small errors in using spherical geometry; the Earth is actually roughly ellipsoidal (or more precisely, oblate spheroidal) with a radius varying between about 6,378 km (equatorial) and 6,357 km (polar), and local radius of curvature varying from 6,336 km (equatorial meridian) to 6,399 km (polar). 6,371 km is the generally accepted value for the Earth's mean radius. This means that errors from assuming spherical geometry might be up to 0.55% near the equator, although generally below 0.3%, depending on the latitude and the direction of travel. An accuracy of better than 3 m in 1 km is mostly good enough, but if a user wants greater accuracy, the system could use the Vincenty formula for calculating geodesic distances on ellipsoids, which gives results accurate to within 0.5 mm (Veness, 2010).

Up to this point, we discussed how to calculate the great circle distance between two points. Now if the system is given one point with latitude Lat_1 and longitude Lon_1, a heading θ_H, and a distance d, the latitude Lat_2 and longitude Lon_2 of the destination point can be calculated by the inverse of Equations 3.1 to 3.12 (Garcia-Castelloanos 2007).

$$Lat_2 = \arcsin\left(\sin(Lat_1) \cdot \cos\left(\frac{d}{R}\right) + \cos(Lat_1) \cdot \sin\left(\frac{d}{R}\right) \cdot \cos(\theta_H)\right) \tag{3.13}$$

$$Lon_2 = Lon_1 + \arctan\left(\frac{\sin(\theta_H)\sin\left(\frac{d}{R}\right)\cos(Lat_1)}{\cos\left(\frac{d}{R}\right) - \sin(Lat_1 \cdot \sin(Lat_2))} \right) \tag{3.14}$$

To get the bearing angle to the mid-point of the wingtips, the reverse angle of the heading is used. Actually, the previous geographical calculation does not consider altitude. It returns only the location with latitude and longitude. Now it needs to add altitude information with pitch angle, θ_P.

$$d' = d \times \cos(\theta_P) \tag{3.15}$$

$$h = d' \times \sin(\theta_P) \tag{3.16}$$

$$\begin{aligned} &\text{If pitch} > 0 \quad Altitude_{newOrigin} = Altitude_{oldOrigin} - h \quad . \\ &\text{If pitch} = 0 \quad Altitude_{newOrigin} = Altitude_{oldOrigin} \quad . \\ &\text{If pitch} < 0 \quad Altitude_{newOrigin} = Altitude_{oldOrigin} + h \end{aligned} \tag{3.17}$$

Let Lat_1 and Lon_1 be the latitude and longitude of the aircraft, Lat_2 and Lon_2 be those of the wing tip, *d* be the length of the wing, R be the radius of the Earth (in meters), and θ be the heading of the aircraft. The ratio *d/R* is the angular distance of the wing span. From these quantities, Equations 3.13 and 3.14 calculate the position of the right wing tip (Veness, 2010).

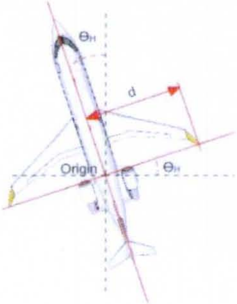


Figure 3.21 Positions of wingtips relative to the aircraft center of mass

Most trigonometry library functions have domains or ranges between -π and +π (that is, between -180° and +180°), whereas navigation uses compass bearings in the range between 0° and 360°. Thus conversion to degrees and modulo arithmetic (the % operator) of (θ+360) % 360 are needed. All lengths are measured in meters in order to have the same metric system as Google Earth.

The next step is to find the altitude for each wingtip and at the same time to correct the longitude and latitude calculated by Equations 3.13 and 3.14 based on the roll angle. This procedure is what makes a ribbon twisted based on the roll angle. Let *h* be the elevation of the wingtip relative to the aircraft center of mass (Figure 3.22). However, the system passes a modified distance value using Equation 3.18 in order to get the value of d'.

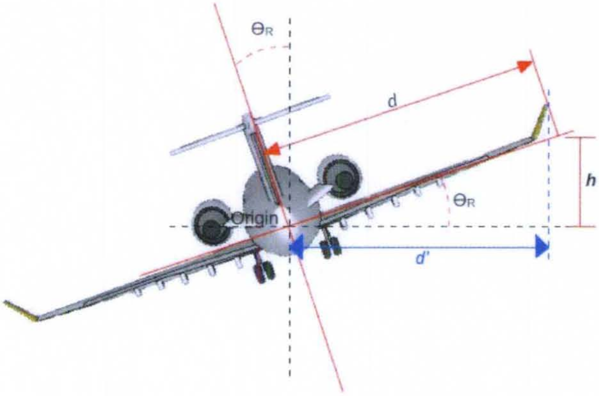


Figure 3.22 Adjustment to wingtip position due to aircraft roll angles.

$$d' = d \times \cos(\theta) \tag{3.18}$$

Then, when the system takes into consideration of the roll angle, it needs to find the adjusted position of each wingtip by adding the height of each wingtip. With this calculation, the system adjusts positions of wingtips. Then, it finally sets up the altitude.

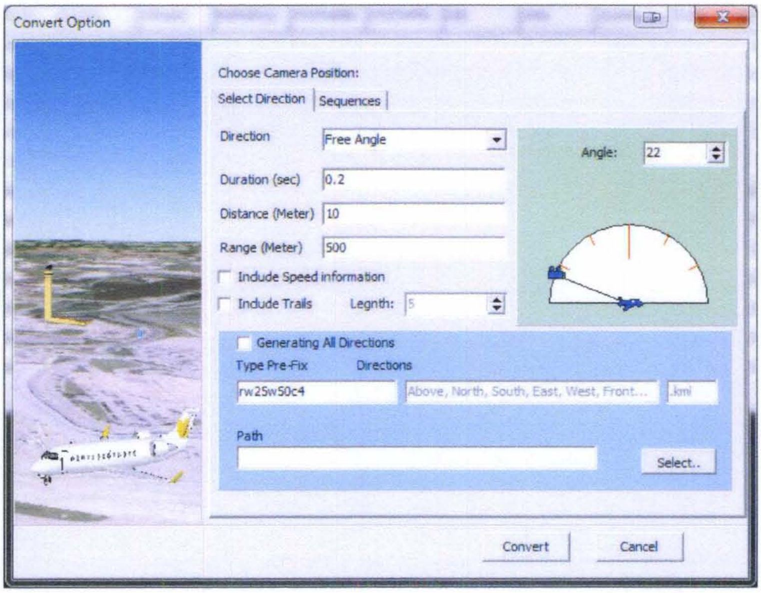
If the roll angle is positive, it means the right wing is higher than the left and the system adds this value using Equation 3.20.

$$h = d' \times \sin(\theta) \tag{3.19}$$

$$Altitude_{RightWing} = Altitude_{newOrigin} + h \tag{3.20}$$

Animation

For generating animation with a 3D model of the aircraft, the user can choose the *Create Animation* option. Choosing this option leads to the next page to set up the camera properties with several choices.



Direction: Free Angle
Duration (sec): 0.2
Distance (Meter): 10
Range (Meter): 500
☐ Include Speed information
☐ Include Trails Length: 5
Angle: 22
☐ Generating All Directions
Type Pre-Fix: rw25w50c4
Directions: Above, North, South, East, West, Front... .kml
Path:

Figure 3.23 Select Direction tab in Animation Option

The first option the user can set is direction. In Figure 3.23, the user selects camera's directions from the aircraft. Table 3.1 lists the directions and descriptions.

Above	The camera is located above the aircraft and follows the aircraft.
North	The camera is located to the North of the aircraft and follows the aircraft.
South	The camera is located to the South of the aircraft and follows the aircraft.
West	The camera is located to the West of the aircraft and follows the aircraft.
East	The camera is located to the East of the aircraft and follows the aircraft.
Front	The camera is located in front of the aircraft and follows the aircraft.
Rear	The camera is located to the rear of the aircraft and follows the aircraft.
Free Angle	The user selects an angle between 0 and 180 degrees; 0 degree is the same as Rear; 180 degrees is the same as Front; 90 degrees is the same as Above.

Table 3.1 Camera's directions

By selecting one of the these options in Table 3.1, the system creates the virtual camera with its position relative to the Earth's surface as well as the viewing direction to the aircraft. To set up the position, Equations 3.13 and 3.14 are used in the system except the *Above* option because the position of camera is the same as the aircraft except the altitude. In the case of the *Free Angle* option, the system uses Equations 3.19 and 3.20 additionally. The user adds a value, D_{camera}, for the distance between the aircraft and the virtual camera in meters. Similar to the calculation of actual position of a wingtip, D_{camera} is used in place of the distance of the wingtip in Equations 3.13 and 3.14. The θ in the equations will be the direction from the aircraft. These four equations are enough to set the position of the virtual camera. After setting up the position of the virtual camera, the next step is to set the view direction of the camera. Figure 3.24 shows the orientation of the view in Google Earth.

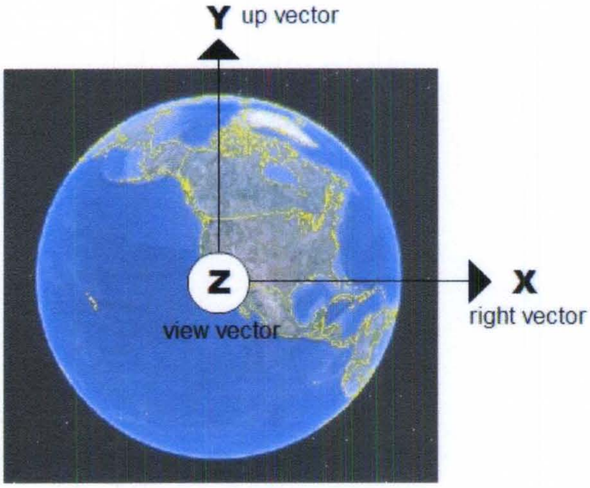


Figure 3.24 Directions in Google Earth

The X axis points toward the right of the virtual camera, and the "tilt" of the camera rotates around this axis. If the value is 0 degree, the camera aims straight down toward the earth. The Above option (Table 3.1) uses this value to have a downward view of the aircraft from the above. The North, South, East, West, Front, and Rear options use 90 degrees of tilt so that the camera aims toward the horizon. The Y axis, the up vector, defines the up direction relative to the screen. The Z axis points from the center of the screen toward the view point, and the "heading", direction (azimuth) of the camera, rotates around this axis. The inverse direction of the virtual camera is the heading of the view point.

Zooming in and out is controlled by the range value, a distance in meters from the position of the virtual camera. This element is used to see a model where its altitude is affected by `altitudeMode` in Google Earth. There are three modes of altitude: clamp to ground, relative to ground, and absolute.

- CLAMP TO GOURND: This ignores the altitude of object and places it on the ground.
- RELATIVE TO GROUND: This interprets the altitude as a value in meters above the ground.
- ABSOLUTE: It interprets the altitude as a value in meters above the sea level.

Figure 3.25 introduces the range, orientation, and altitude elements.

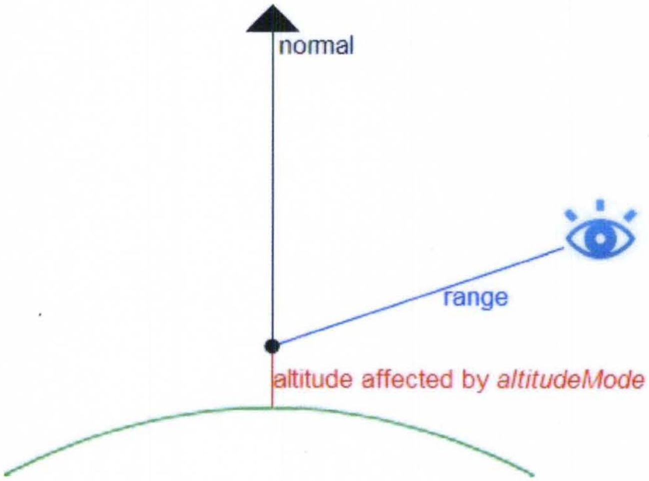
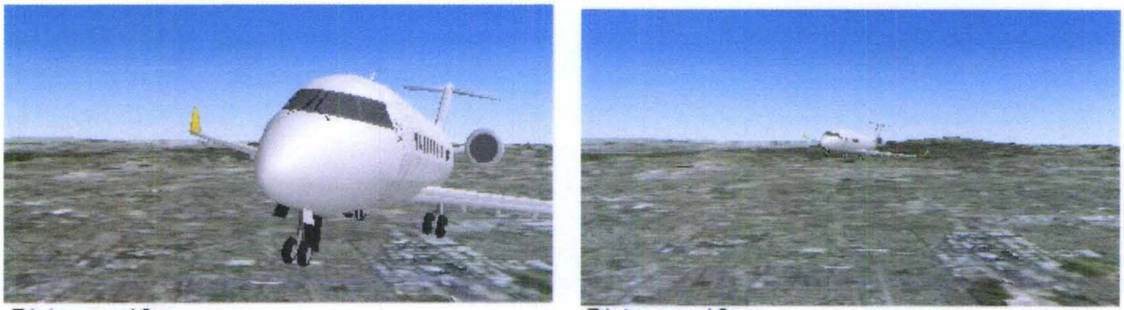
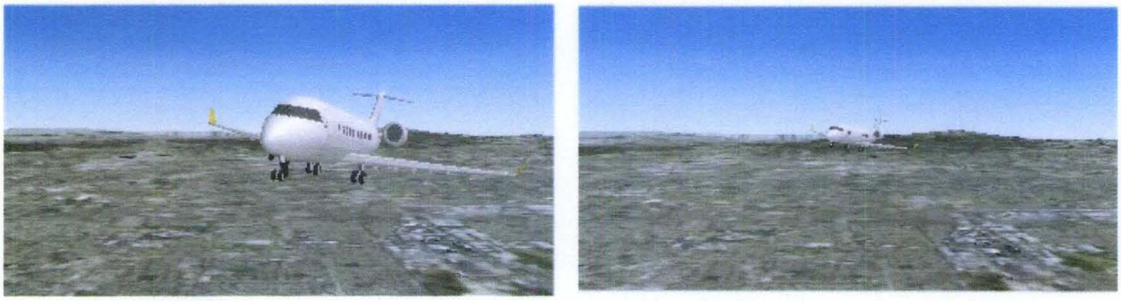


Figure 3.25 Range, orientation, and altitude in Google Earth



Distance: 10m
Range: 50m

Distance: 10m
Range: 500m



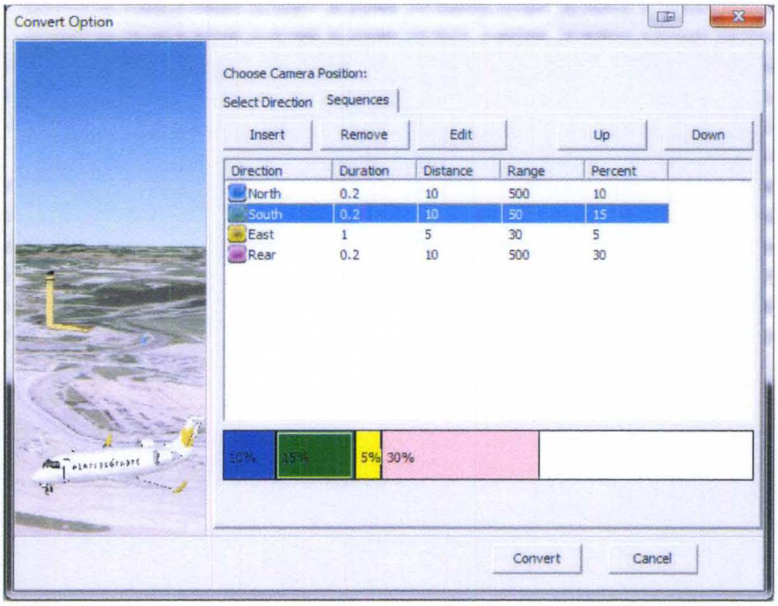
Distance: 100m
Range: 50m

Distance: 100m
Range: 500m

Figure 3.26 Distance vs. Range

When a user selects the *Generating All Directions* option, it will generate all directions without asking file names to save. However, the user has to define the folder and the prefix of file names. The *Include Speed Information* option can be added into the KML file to display the current speed at the selected frame. The *Duration* field is for specifying a time span for events. It specifies the length of time that the browser takes to fly from the previous point to the next point.

When the system loads each record, it generates KML code using options that the user chooses. Between KML code for each record, the system adds the `AnimateUpdate` tag to control changes of the model during a tour and the `FlyTo` tag to move the camera in parallel with the model. This should contain the duration and method of flight, respectively. The smooth mode allows for an unbroken flight from point to point. An unbroken series of smooth modes will begin and end at zero velocity, and will not slow down at each point. In the bounce mode, the camera bounces from waypoint to waypoint, beginning and ending at zero velocity at each waypoint.



Direction	Duration	Distance	Range	Percent
North	0.2	10	500	10
South	0.2	10	50	15
East	1	5	30	5
Rear	0.2	10	500	30

Figure 3.27 Sequences Option

In the *Sequences* option, the user can select directions in sequences by choosing percent of the whole data to generate. The user may insert as many sequences as he wants if the total of percent is not over 100 percent. Sequences can be changed by clicking up and down button or drag and drop selected sequences.

CHAPTER 4

RESULTS AND COMPARISONS

4.1 Introduction

This chapter shows the result of the 3D in-flight visualization and comparisons among other systems related with flight visualization. Because of the limitation of the Google Earth API, the system creates several effects by its own methods or ways. For the flight data, the system is also able to remove the noisy data in order to analyze the flight. At the end of this chapter, comparisons with other system are provided.

4.2 Effects

After setting up the camera properties, the user is also able to choose other visual effects: visibility, and the cloud effect. The flight data used in the system does not contain weather information. However, for realistic flight visualization, the system offers these visual effects.

4.2.1 Visibility

The visibility effect is similar to the fog effect. In the KML API, the way of placing picture is called overlay that is the base type for image overlays drawn on the surface of the Earth or on the screen. Typically, this technique is used to draw pie or bar charts on the surface of the Earth to show the statistic information by GIS. Depending on how a chart or image is placed on the surface, two properties are used, Clamped to ground or Absolute. Clamped to ground is the same as the ground level and Absolute is relative to the sea level. Figure 4.1 shows the different between two overlay modes.

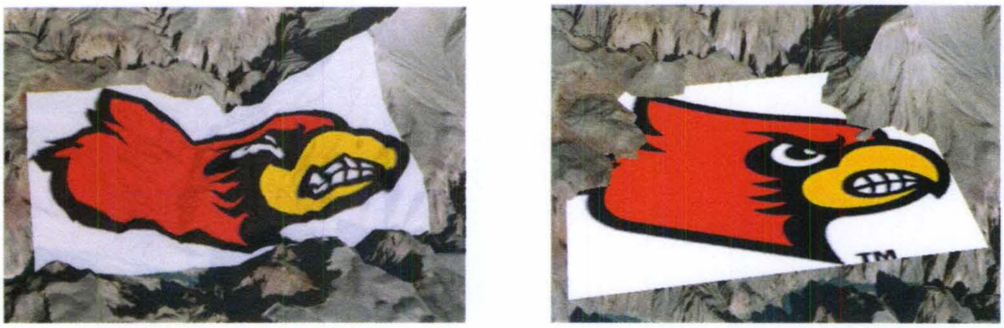
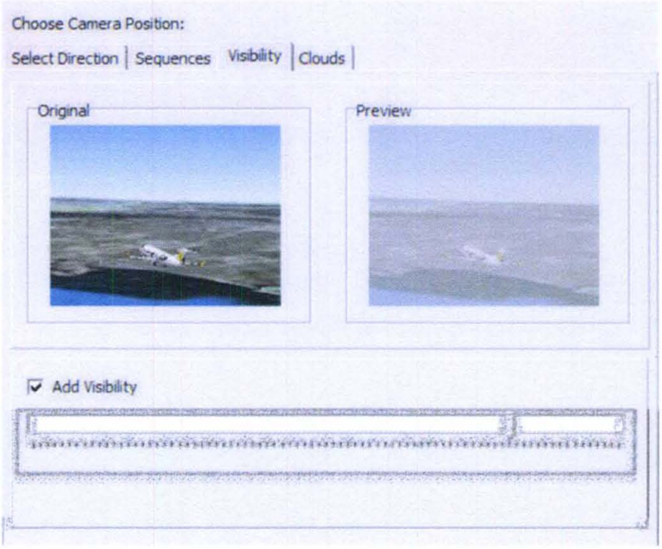


Figure 4.1 Clamped to ground and Absolute mode

In order to place an image or draw a color on the screen, screen overlay is used. In the system, the foreground color is light gray and a user can define an opaque value for the overlay. The following dialog is for setting the opaque value and the user can confirm the visibility by the preview screen. Figure 4.3 shows the result with different opaque values.



☑ Add Visibility

Figure 4.2 Visibility Option dialog

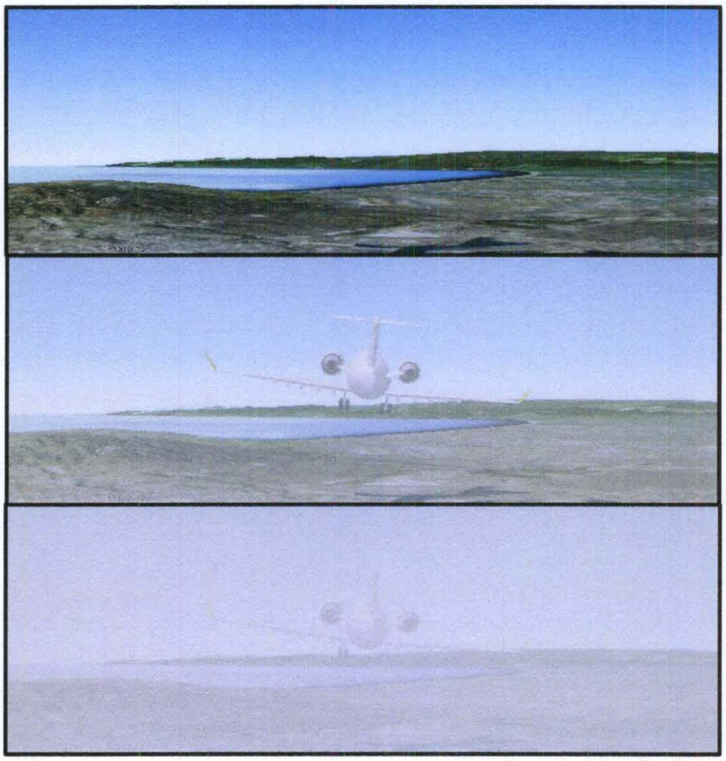
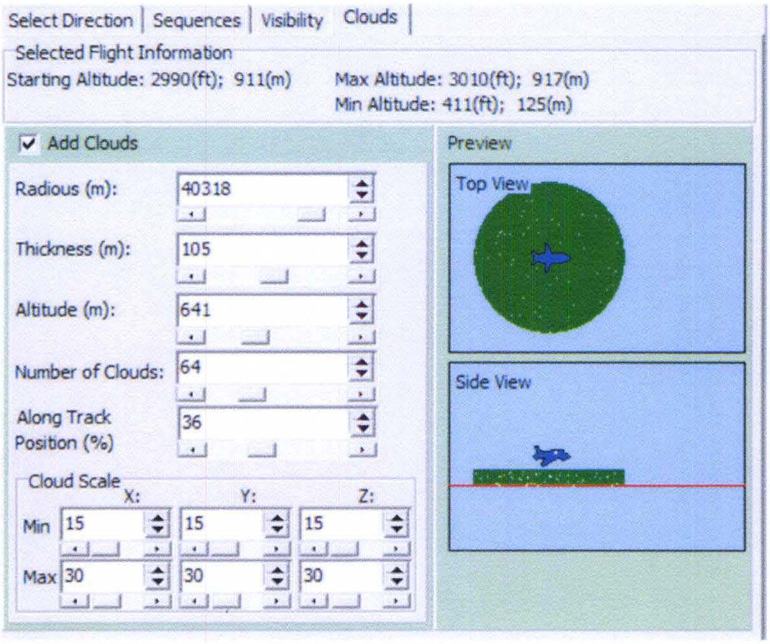


Figure 4.3 Results of different opaque values

4.2.2 Clouds Effects

Google Earth renders remarkable terrains and detailed models of the streets. However, it displays only a clear sky without any cloud at all. The system also generates a KML code to put clouds in the virtual world. These clouds are placed in the sky randomly with the user's parameters such as the radius of area in meters from the aircraft starting point, altitude from the ground level, thickness of the region that contains clouds, the number of clouds, along track position in percent and ranges of scales along the X, Y, and Z axes in the model's coordinate space to define clouds' sizes. For better understanding, the cloud dialog window in the system shows the top and side preview screens of the flight. In Figure 4.4, the cloud dialog window provides the user several options for generating clouds.



Selected Flight Information
Starting Altitude: 2990(ft); 911(m)
Max Altitude: 3010(ft); 917(m)
Min Altitude: 411(ft); 125(m)

☑ Add Clouds

Radious (m): 40318
Thickness (m): 105
Altitude (m): 641
Number of Clouds: 64
Along Track Position (%): 36

Cloud Scale

	X:	Y:	Z:
Min	15	15	15
Max	30	30	30

Figure 4.4 Cloud option dialog

A cloud is composed of tiny water droplets or ice crystals that are suspended in the air. During cold days, people can see their breath because when they breathe out, the warm air includes moisture in the form of water vapor, and when this warm air collides with cold air, it dries the air outside. This is a basic idea how a cloud forms. Each cloud has a different shape and size. There are many kinds of clouds such as Cumulus, Cirrus and Stratus. Cumulus clouds are puffy. They have flat bottoms and are low in the sky. Cumulus clouds usually mean fair weather. If they grow tall, they can become thunderheads and bring rain. Cirrus clouds are the highest clouds. They look white and feathery. Stratus clouds are the low clouds. Fog is a stratus cloud at the ground level. They look like a low gray blanket. Stratus clouds bring rain or snow (Meyerhorn, 2001). Figure 4.5 shows common types of cloud.

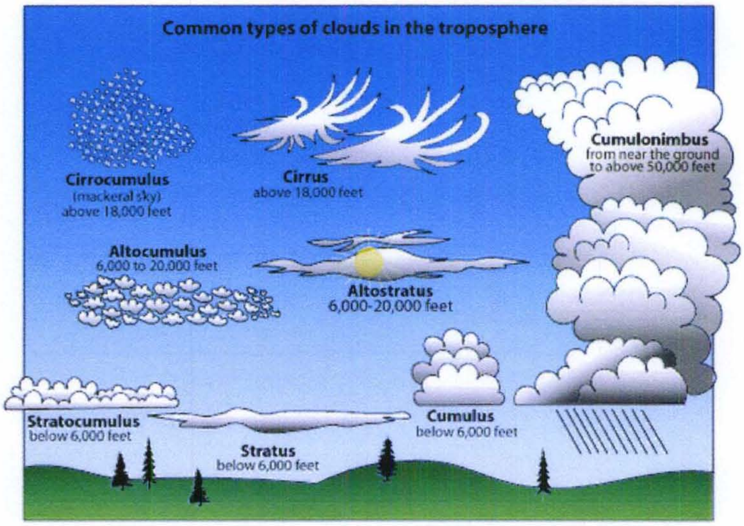


Figure 4.5 Common types of cloud (http://eo.ucar.edu/webweather/cloud3.html)

The system uses only one cloud model that looks like the Cumulus type. In order to render the model with animation faster, the size of the model is also an important factor. Currently the file size of the cloud model is 249 KB. However, with this cloud model, it is possible to render other types of cloud or combination of the types by choosing appropriate scales and opaque values. In Figure 4.6, a cloud model has X, Y, and Z axes to define the size of a cloud.

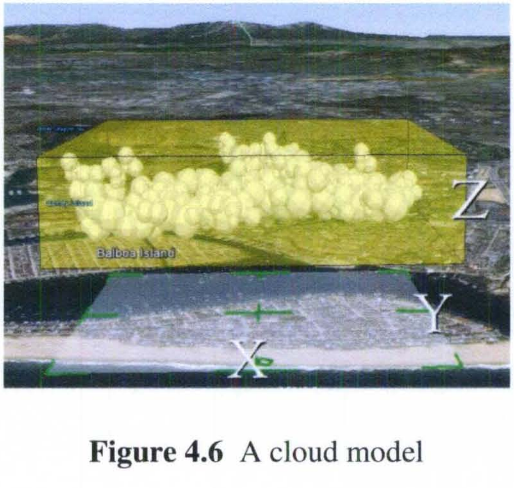


Figure 4.6 A cloud model

In the cloud option window, if the user sets up the range of the X and Y values lower than the Z value, the type of the cloud looks like a Cumulonimbus cloud. In the other case, the cloud is similar to a Stratocumulus cloud. With an opaque value and a smaller value of Z, the Stratocumulus cloud converts into a Stratus cloud. With wide ranges of X, Y, and Z values, the system will generate combinations of the cloud types. Figure 4.7 shows that a combination of cloud types is generated, and Figure 4.8 also shows the screen shot of the result.

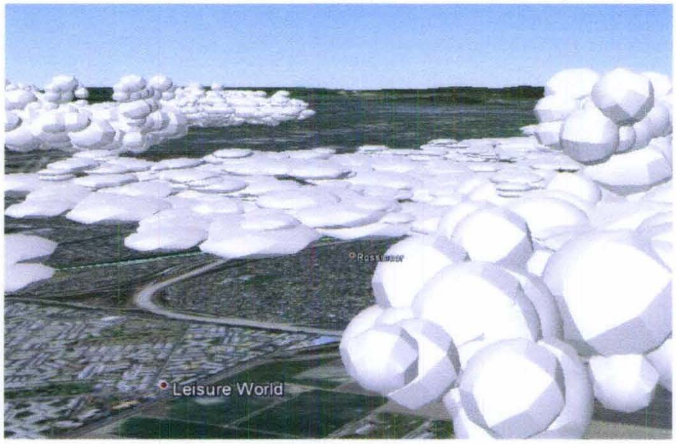


Figure 4.7 Combination of cloud types generated.

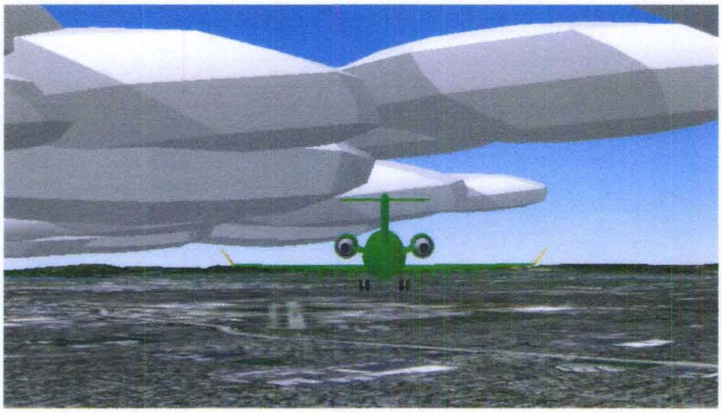


Figure 4.8 Result of flight visualization with clouds

4.3 Analysis

The flight visualization system has a function to create a 3D flight animation that may attract regular users. However, the main feature of the system is drawing a path or ribbon of flight for researchers or flight analysts.

4.3.1 Noisy Data Removal

Currently, flight analysis uses 2D chart graphs. The system uses the path option that draws the path of flight on the surface of the Earth. A lot of the flight data may be noisy. Sometimes the aircraft would have traveled more than 1NM in 1 second according to the recorded latitude and longitude. Figure 4.9 contains an example of noisy data.

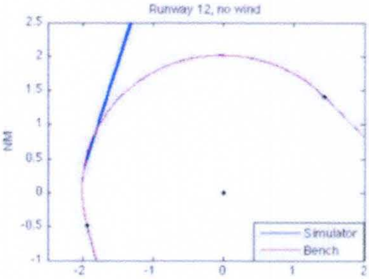


Figure 4.9 An example of noisy data

The system has a data cleaning procedure to properly analyze the data like Figure 4.10 with the cleaned data.

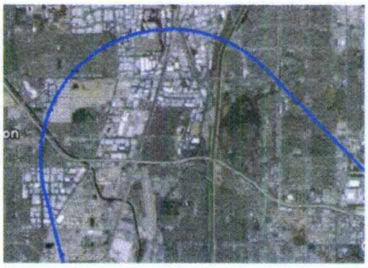


Figure 4.10 Cleaned data

At the initial part of the some flight data, the records may not have correct information. For example, both longitude and latitude values are 0 at the first record. The system tries to ignore these values for the purpose of animation or analysis. Like the way of the interpreter, the system loads data record by record and writes KML code according to the user's parameters. When the system loads a record, it also reads next two records. Then, it calculates the distances between the first and the second position and between the second and third position. If the distance of the former calculation is twice than the distance of the latter calculation, the system considers that the second position is a noisy record. This step is repeated until the second record from the last. Last two records are compared with average of the distance that is stored in previous steps because last two records do not have next two more records.

4.3.2 Ribbon Representation

The most significant work in the system that has never been done before is the ribbon representation. It visualizes ribbons of multiple flights and compares the flight data in the 3D space instead of 2D. With the system, one can see the flights in three dimensional space from every possible angle, and one can freely zoom in and out, and the flights can be played forward and backward at various speeds. Figure 4.11 shows the current way of analyzing flight data with ground trace, altitude, roll angle, and so on.

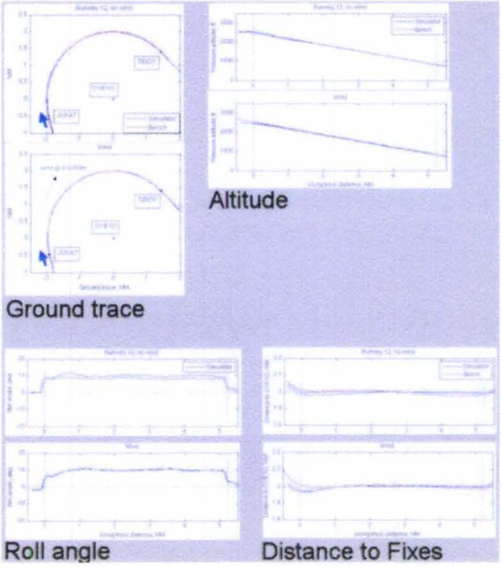


Figure 4.11 Analyzing flight data in 2D

In Figure 4.11, one flight will produce lots of charts. Although ground traces and altitude can be read together to gain some insight to the flights in the three dimensional space, they do not capture the flights in motion in space. The system makes it possible to show multiple flight data in one scene. Figure 4.12 displays 17 flights in one scene.

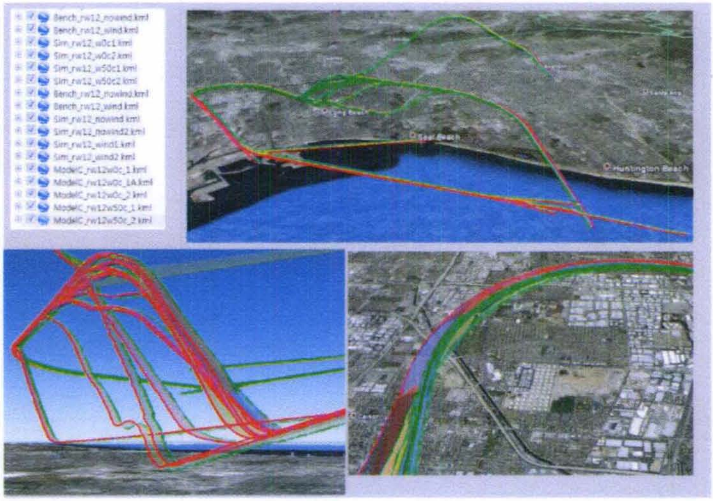


Figure 4.12 3D Visualization of ribbon of 17 flights

4.3.3 Path with Altitude

Previously, the drawing path option is to draw a path on the surface of the Earth. The system also introduces a new option called Drawing Path with Altitude. As in Figure 4.13, it can draw multiple paths with altitude. Each vertical line represents the interval when data is recorded. The analyst can see that the missing vertical line indicates the time when noisy data was recorded and the system cleaned it up.

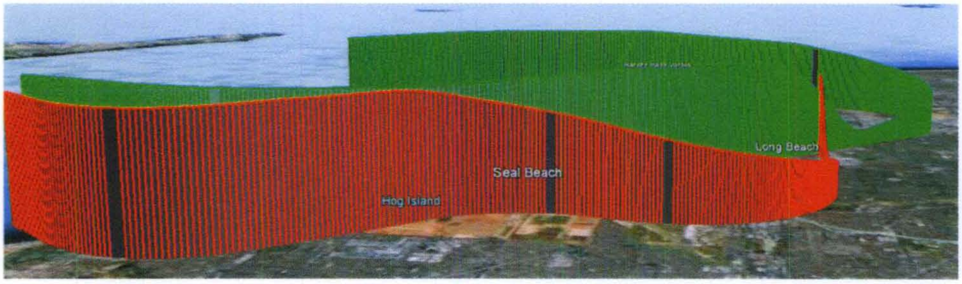
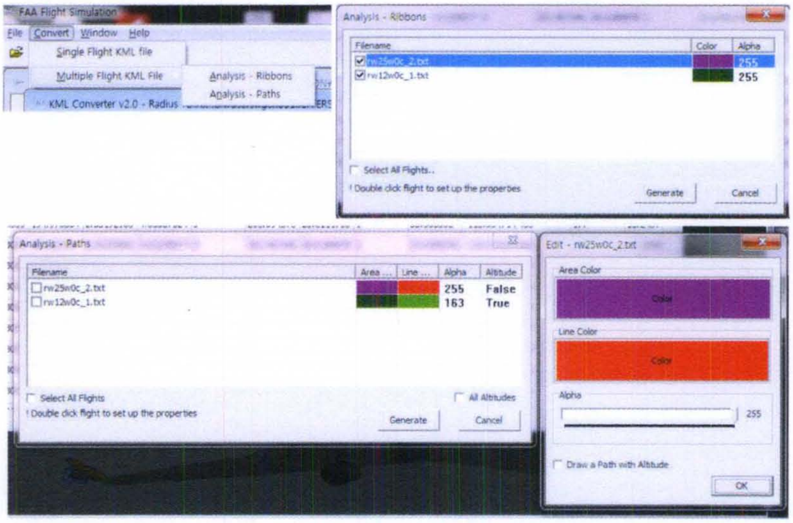


Figure 4.13 3D Visualization of paths with altitudes

In Figure 4.14, the way of creating these analysis visualization is to choose one of multiple flight analysis menu, either ribbon or path, and then to select options.



Analysis - Ribbons

Filename	Color	Alpha
☑ rw25w0c_2.txt	🟪	255
☑ rw12w0c_1.txt	🟩	255

☐ Select All Flights..

! Double click flight to set up the properties

Analysis - Paths

Filename	Area ...	Line ...	Alpha	Altitude
☐ rw25w0c_2.txt	🟪	🟥	255	False
☐ rw12w0c_1.txt	🟩	🟩	163	True

☐ Select All Flights

☐ All Altitudes

! Double click flight to set up the properties

Edit - rw25w0c_2.txt

Area Color: 🟪

Line Color: 🟥

Alpha: 255

☐ Draw a Path with Altitude

Figure 4.14 Multiple Flight Analysis menu

After generation of the flight from the system, a user can choose a flight of interest to compare with other flight data and can see any angle of the flight from the scene.

4.4 Comparisons

4.4.1 Flight Graphical User Interface

C. Santiago (2006) implemented Flight Graphical User Interface (GUI) tools in order to analyze flight conflict for the Federal Aviation Administration's (FAA) Conflict Probe Assessment Team (CPAT). The flight GUI tools are based on the Java platform and interact with an Oracle relational database that contains actual flight data from the Air Route Traffic Control Center (ARTCC). The tools can visualize three modes: single flight mode, encounter mode, and alert mode. The flight GUI can display the flight information on the several visual methods, such as the Visualization Window, the Compass Window, the Tabular Data Window, and the TZ Graphs Window. The Visualization Window shows the plots of the flights' trace data along with a plot of the ARTCC boundary. This plot representation is the same concept as the path tracking of the flight visualization of this dissertation. However, Santiago's flight GUI tools only support the 2D representation of flights.

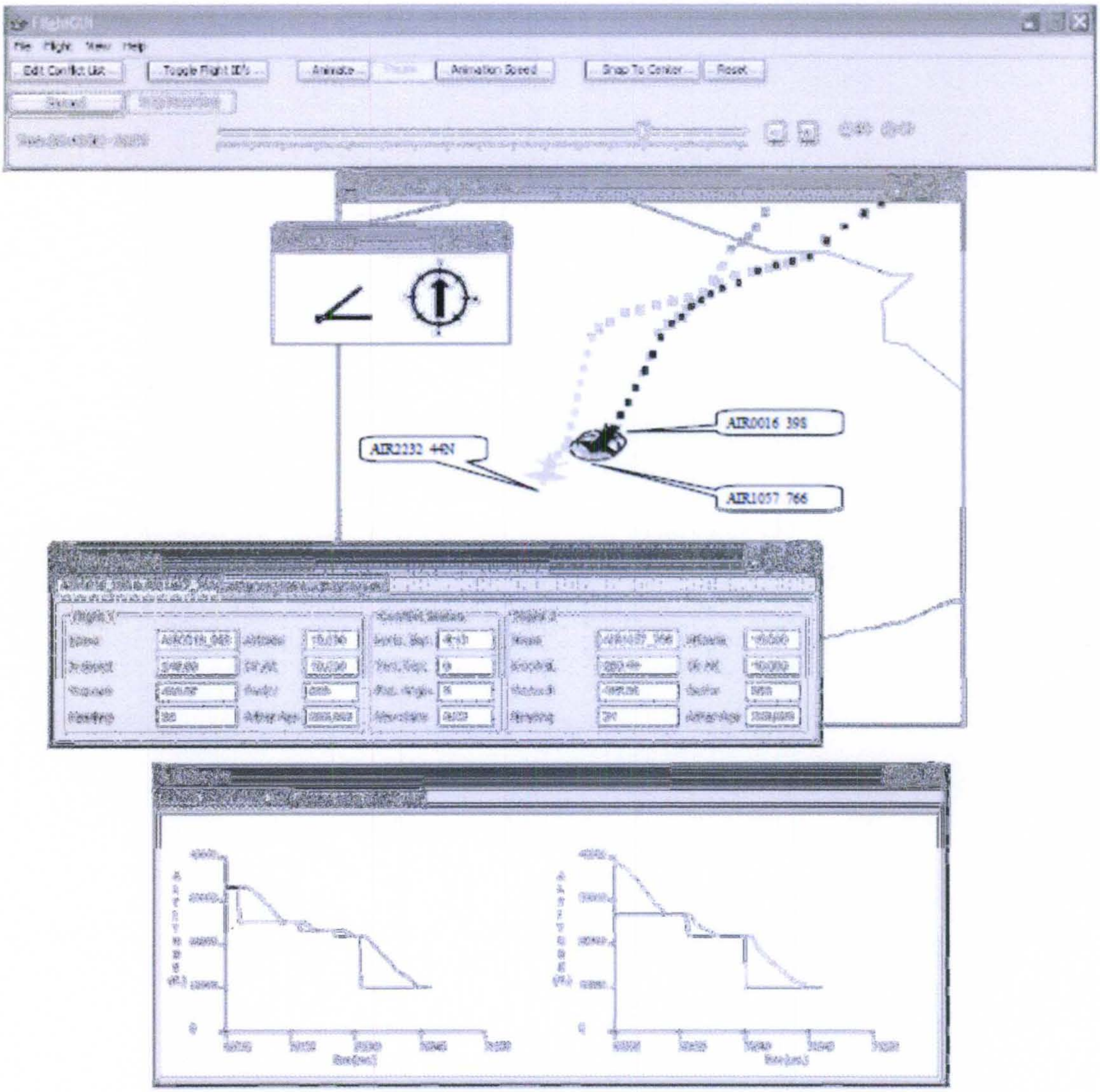


Figure 4.15 FlightGUI user interface

4.4.2 Autonomous Quadrotor Flight Test

R. Goel et al. (2009) used the Flightgear simulator for 3D visualization of the result of autonomous quadrotor flight tests. They developed the Rotary-wing Unmanned Mini Aerial Vehicles (RUMAV) robot and collected flight data during the flight test. For 3D visualization, the Flightgear is used with Matlab/Simulink. Flightgear is enabled to choose various flight dynamics models and has accurate real world scenery data. It contains over 20,000 real world airports with the full environment and correct runway markings and placement. Goel's team used the Flightgear for the 3D visualization of their flight simulation. Although the Flightgear is good for the 3D visualization, it requires additional graphics power for 3D modeling.

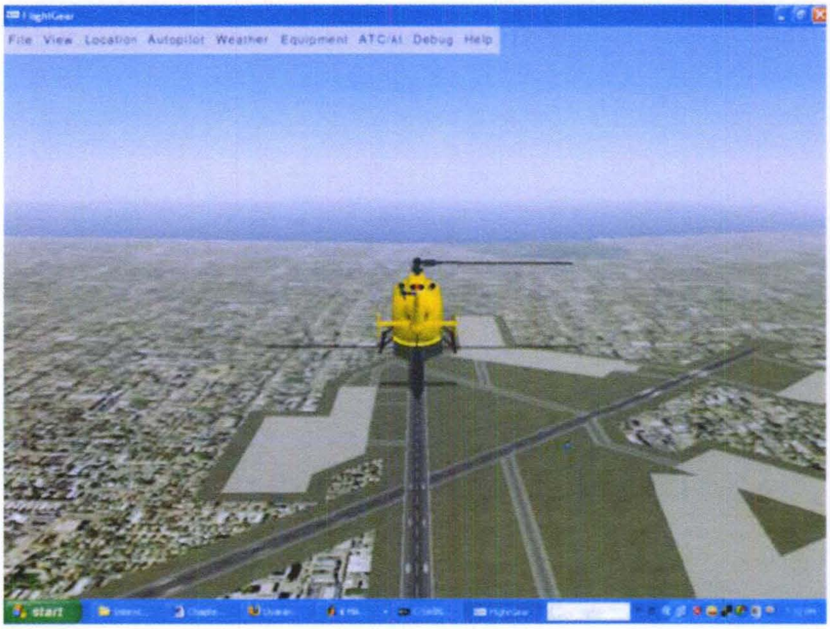

FlightGear
File View Location Autopilot Weather Equipment ATC/AI Debug Help
start

Figure 4.16 Snapshot of simulation in Flightgear

4.4.3 Flight Visualization using the Health and Usage Monitoring System

A. Singh *et al.* (2011) introduce the flight visualization using the Health and Usage Monitoring System (HUMS) of US military aircrafts. HUMS data consists of the core parameters and useful parameters. Core parameters are Airspeed, Pitch Attitude, Roll Attitude, Pitch Rate, Roll Rate, Yaw Rate, Vertical Acceleration, Vertical Velocity, Engine Torque, and Weight on Wheels, Rotor Speed, Fuel Quantity, Pilot Stick Positions, Altitude, Outside Air Temperature, Gross Weight, and Rotor Brake. Useful parameters are Lateral and Longitudinal Acceleration, Ground Speed, Ground Track, Radar Altitude, Swashplate Tilt, Rotor Torque, Parking Brake, and Rotor Flapping. Some of the parameters are the same as the present flight visualization system. Their aim is to check the health state of the aircraft.

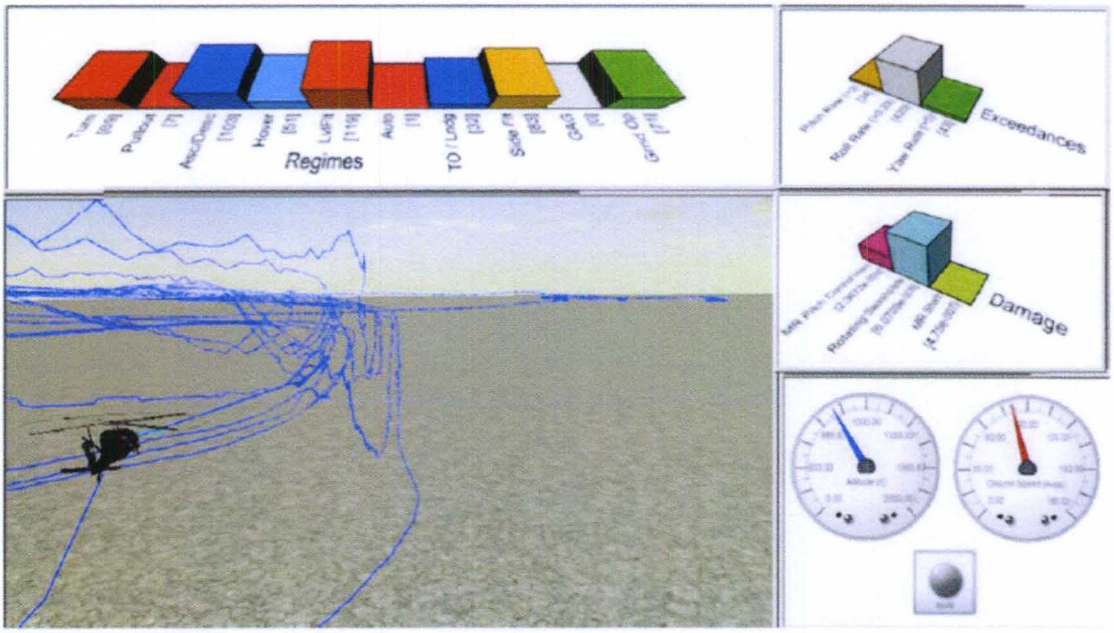


Figure 4.17 Full screen shot display of flight visualization

CHAPTER 5

FUTURE WORKS

5.1 Introduction

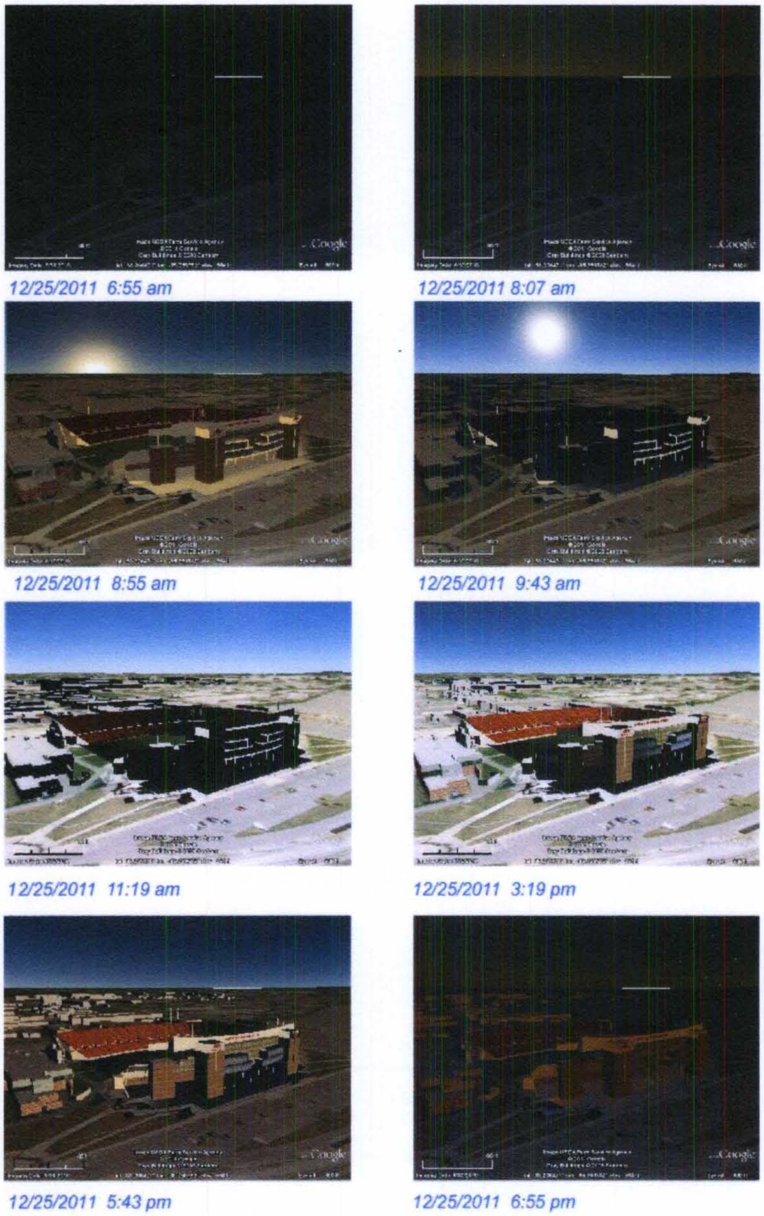
This chapter describes some aspects and future work to enhance the system. The future work is to test the system with entire flight data (from take-off to landing), to find a way to incorporate weather conditions in Google Earth, and to conduct a survey of geographical calculations.

5.2 Flight Data

Currently, the system have data for 54 flights and they consist of 750 records on the average that they lead to about 12 minute and 30 seconds of animation if the sample rate of data is 1 Hz and the duration is 1 second during playback. Without any options such as speed information, trail, and so on, the system generates about 18,000 lines of KML code from the data and the file size is about 850 KB. During the 12.5 minutes of the flight time, the aircraft moved about 25 miles near the landing airport. The next procedure is to test data which contains whole flights from take-off to landing, covering a thousand miles or more in distance. From this work, the maximum size of KML file to execute flight visualization in Google Earth can be found out, and problems need to be solved if they exist.

5.3 Weather Conditions

In Google Earth, a user can time travel for certain day and time. It does not mean to travel to the future or the past to see what happens on the street. There is a feature called *sun* in Google Earth. It displays the sun and sunlight across the landscape. Figure 5.1 shows images of different time on December 25th, 2011 in Google Earth.



12/25/2011 6:55 am

12/25/2011 8:07 am

12/25/2011 8:55 am

12/25/2011 9:43 am

12/25/2011 11:19 am

12/25/2011 3:19 pm

12/25/2011 5:43 pm

12/25/2011 6:55 pm

Figure 5.1 Different time in Google Earth

The user can set up the time of the flight. However, there are no functions for weather such as wind, rain, snow, fog, and visibility. One direction of future work can be incorporating the weather conditions into a KML file.

Currently, the system can generate up to 200 clouds at a time. It is described in the Chapter 4. The next plan is to create lightning effects. Each cloud model has its own size and altitude. A lightning model can be incorporated, and it will link from the ground to the bottom of the randomly selected clouds.

As Chapter 4 introduced, the system can generate the visibility (fog) effect. However, it uses screen overlay. It blurs the whole screen based on the opacity value so that the user may think it is fog. Enhanced visibility effect could be one of the future directions too. By the International Civil Aviation Organization, visibility for aeronautical purposes is the greater of:

- the greatest distance at which a black object of suitable dimensions, situated near the ground, can be seen and recognized when observed against a bright background
- the greatest distance at which lights in the vicinity of 1000 candelas can be seen and identified against an unlit background.

The two distances have different values in air of a given extinction coefficient, and the latter varies with the background illumination. The former is represented by the Meteorological Optical Range. Like these definitions, the visualization should show the objects clearly when they are close to the camera; otherwise, the user can not discern the different distances of objects.

The important factor of the visibility is the interaction of light and particles. A photon is said to be scattered when it is received by a particle and re-radiated at the same wavelength in any direction (Malm, 1999). The author gives us an example that an atmosphere containing nitrogen dioxide (NO_2) will tend to deplete the number of blue photons through the absorption process. The eye detects relative differences in brightness rather than the overall brightness level and it measures contrast between adjacent objects or between an object and its background. Contrast of an object is simply the percent difference between object luminance and its background luminance.

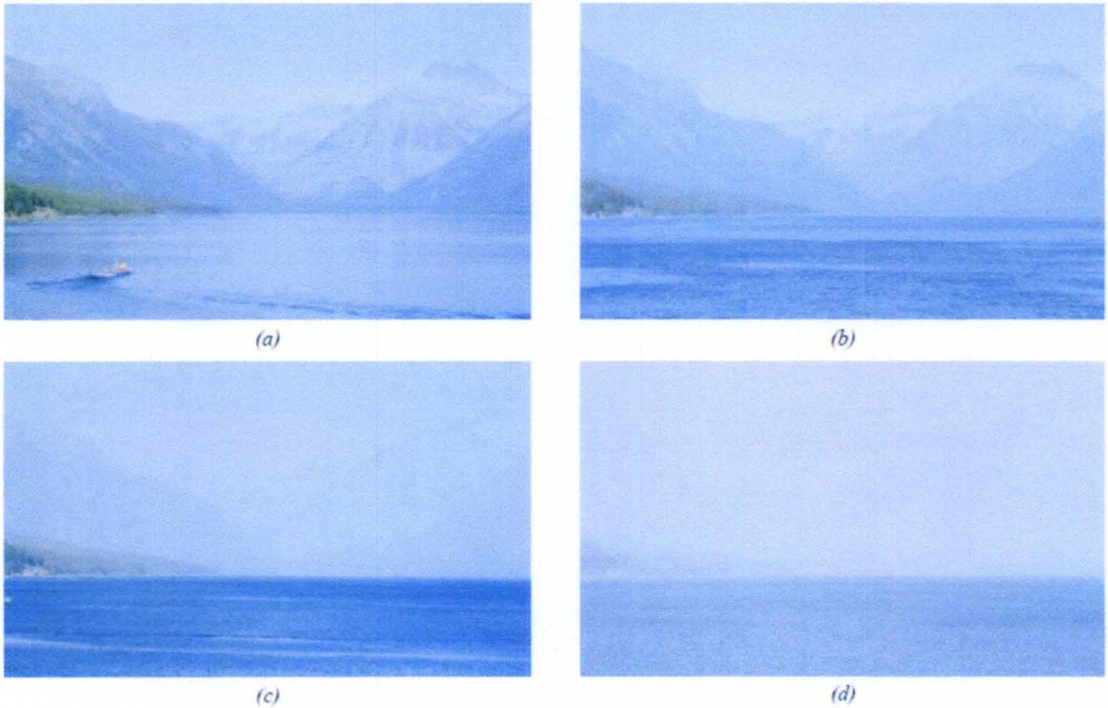


Figure 5.2 Regional or uniform haze on a Glacier National Park (Malm, 1999)

In Figure 5.2, the view is the Garden Wall from across Lake McDonald. Atmospheric particulate concentrations associated with photographs (a), (b), (c), and (d) correspond to 7.5, 12.0, 21.7 and 65.3 μg/m^3 respectively.

The goal of the present work is to provide the user with enhanced visibility effects by inputting the value of the visual range. From the camera position, the user is able to see the objects within the visual range.

5.4 Collision Detection for Turbulence Effect

By the selection of the user, the system can generate 200 clouds in the scene each time with the single animation option. Each cloud model is randomly generated with x, y, and z of the model scale and location of the model such as longitude, latitude, and altitude. With this information of the cloud and the current location in the 3D space of the aircraft, another future direction is to support the turbulence effect by shaking the camera or moving the location of aircraft when the aircraft and a cloud collide.

In Google Earth or KML, it does not support a collision detection API or library for developers. Before the system creates a KML file, an optimized algorithm for detecting collision is needed. According to Jimenez et al. (2000), many applications in Computer Graphics require fast and robust 3D collision detection algorithms and many collision detection algorithms have been proposed in recent years within the fields of Computational Geometry, Robotics, and Computer Graphics. In Computer Graphics, the emphasis is placed on the possibility of detecting collisions in real-time, especially for computer animation applications, even if speed is gained at the expense of precision, thus allowing the adaptation of output precision to the limited computing time.

In the case of Flight Visualization, the area for collision detection is huge, the number of models is luxuriantly increased each time, and the scale of each cloud model is much larger than the sizes of any other models in computer animations or games. Also, the models are in the virtual world that is using a geographic location system. This means geographical calculation is needed to get the distances among the models. Most collision algorithms use space partitioning techniques such as k-d trees, Octrees, and BSP (Binary Space Partition) trees, which can not be adapted directly in the present system because this system needs more update rates in real time.

Teschner et al. (2003) proposed a new approach to collision detection of dynamically deforming objects that consist of tetrahedrons. According to their result, the algorithm can detect collisions and self-collisions in environments of up to 20k tetrahedrons in real time. Using a hash table, the presented algorithm is integrated in a physically based environment. All cells are traversed from the discretized minimum to the discretized maximum of the AABB, axis-aligned bounding boxes, and all vertices found at the corresponding hash table bucket are tested for intersection.

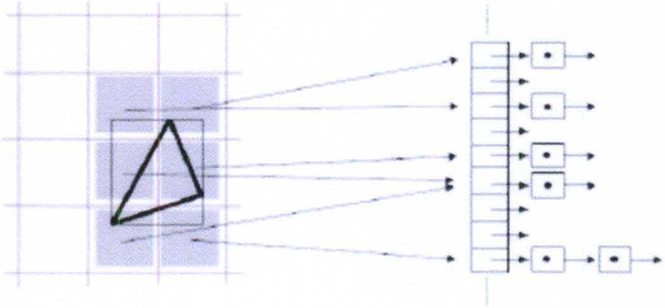


Figure 5.3 Teschner's collision detection algorithm

One of the future directions is to implement a collision detection algorithm that is suitable for Flight Visualization. Teschner's idea using hash functions will be adapted because it is very efficient. However, BSP trees will be added into the algorithm too. Instead of calculations of all distances between the aircraft and clouds, the system gets the area of interest to calculate the distances. Because the aircraft in the data could not turn itself more than 90 degree in both directions, the system will have a threshold degree from the heading angle to ignore the distances to the clouds that are located off to the side by using BSP trees. The system is in the 3D space so that another set of BSP trees is needed for the altitude; another threshold value is needed for the altitude angle range. For instance, let us assume that an aircraft is flying somewhere in Kentucky. The system does not need to get the distance from the aircraft to a cloud that is in Georgia. This threshold value can reduce the number of calculations in the hash table. The system will use a global bounding box for each model instead of calculating AABBs for the following two reasons. One is that the space is too big compared with the models, so that small errors can be accepted. The size of the hash table will be gradually increased because hash values are computed for all cells affected by the AABB of a tetrahedron. The system can generate 200 clouds at a time, and there will be numerous tetrahedrons from the models.

When the collision detection algorithm is developed, it will be possible to detect collision with other models such as buildings, trees, or terrains such as mountains in Google Earth. It can show the user when an aircraft crashes into the objects.

5.5 Survey of Geographical Calculations

As discussed in Chapter 2, there are many different formulas for geographic calculations. However, the system uses only one set of formulas, the Great Circle Calculation. Based on each position of the aircraft, this calculation returns the position of the camera. The next plan is to add other geographical calculations to compare the result of the camera position and to see how they affect the visualization of the flight.

5.6 Stereoscopic Visualization

It is proposed to present the visualization by stereoscopic vision, which will provide a realistic visual representation of the flights. Stereoscopic vision is the next generation of visualization techniques that two images and high-tech wireless glasses give 3D perception and depth information. Human eyes are located in front of the head side by side. When the user looks at some object at the scene, each eye sends a different image to the brain, and the brain combines two images from both eyes. Because of the slight difference in the two images, depth is unconsciously inferred. By using this concept, many 3D products, such as TV, game devices, and video cards, are in the market place. The goal is to provide flight visualization in stereoscopic vision so that a user may perceive the depth of the images and see the flight as if he or she is flying with the aircraft.

REFERENCES

Benkera, S., Langforda, R., and Pavlis, T. (2011), *Positional Accuracy of the Google Earth Terrain Model Derived from Stratigraphic Unconformities in the Big Bend Region, Texas, USA*, Department of Geological Sciences, University of Texas, El Paso, TX.

Burtch, B.(2004), *Direct and Inverse Geodetic Problem; Surveying Engineering Department*, Ferris State University; p41-47.

Chiaro, T. (2010), *Sneak Preview: Delphi 2011 is Delphi XE*, Delphi Insider, http://delphi-insider.blogspot.com/2010_08_01_archive.html.

Veness, C. (2010), Calculate Distance, Bearing and more between Latitude and Longitude Points, Movable Type Scripts, http://www.movable-type.co.uk/scripts/latlong.html

Eick, S. (1994), *graphically display text.* Journal of Computational and Graphical Statistics, 3:127-142.

Embarcadero, *Rapid Application Development Studio*, http://www. embarcadero.com/products/rad-studio.

European Organization for the Safety of Air Navigation (EUROCONTROL) Brussels, Belgium, and Institute of Geodesy and Navigation (IfEN) University FAF Munich, Germany(1998), *WGS 84 Implementation Manual.*

Federal Communications Commission (FCC) (2010), *Reference Points and Distance Computations.* 47 CFR Section 73.208.

Friendly, M. (2009), *Milestones in the History of Thematic Cartography, Statistical Graphics, and Data Visualization.*

Flurchick, K., and Veltman, T. (2001). *Introduction to Scientific Visualization*, National Computational Science Education Consortium.

Garcia-Castelloanos, Daniel; Lombardo, Umberto; *Poles of Inaccessibility: A Calculation Algorithm for the Remotest Places on Earth*; Scottish Geographical Journal; Sep 2007, Vol. 123 Issue 3, p227-233.

Goel, R., Shah, S., Gupta, N., and Ananthkrishnan, N. (2009), *Modeling, Simulation and Flight Testing of an Autonomous Quadrotor*, ICEAE

Google Earth Coverage (2005), *Google Earth High Resolution Imagery Coverage (USA).*

Google SketchUp, http://sketchup.google.com/.

International Civil Aviation Organization (2007), *Annex 3 to the Convention on International Civil Aviation; Meteorological Service for International Air Navigation*

Jean-Paul Rodrigue (2011), *the Geography of Transport Systems*, Department of Global Studies and Geography, Hofstra University, http://people.hofstra.edu/geotrans/eng/ch1en/conc1en/greatcircle.html

Jimenez, P., Thomas, F., and Torras, C. (2000). *3D Collision Detection: A survey*, Comissionat per a Universitats i Recerca, Barcelona, Spain.

Malm, W. (1999), *Introduction to Visibility*, Air Resources Division, National Park Service, Cooperative Institute for Research in the Atmosphere (CIRA), NPS Visibility Program, Colorado State University.

Maor, E. (1998). *Trigonometric Delights*, Princeton University Press, Princeton, New Jersey: Chapter 13 A Mapmaker's Paradise.

Meyerhorn, A. (2001). *Wonders in Weather*, Dondero School, City of Portsmouth, NH, http://www.cityofportsmouth.com/school/dondero/msm/weather/index.html

Moore, L. (1997). *Transverse Mercator Projections and U.S. Geological Survey Digital Products*. U.S. Geological Survey.

National Aeronautics and Space Administration (NASA) (2009), *Shuttle Radar Topography Mission (SRTM): the Mission to Map the World*, Jet Propulsion Laboratory, California Institute of Technology, http://www2.jpl.nasa.gov/srtm/.

OpenGL (2004), *OpenGL Volumizer 2.7 adds Support for Windows*, http://www.opengl.org/news/comments/opengl_volumizer_27_adds_support_for_windows.

Rick (1999), *The Math Forum: Deriving the Haversine Formula*, http://mathforum.org/library/drmath/view/51879.html

Santiago, C., Oaks, R., Paglione, M., and Rusu, A. (2006), *Flight Graphical User Interface: A Visualization Application for Analyzing Conflict Probe Tools*, Air Traffic Control Association's 51st Annual Conference Proceedings.

Singh, A., Iyyer, M., and Phan, N. (2011), *Flight Visualization using HUMS Data*, Condition Based Maintenance (CBM) meeting 2011

Teschner, M. (2003), *Optimized Spatial Hashing for Collision Detection of Deformable Objects*, Vision Modeling and visualization 2003

U.S. Geological Survey (USGS) (1998), *Standards for Digital Elevation Models.*

U.S. Geological Survey (USGS) (2000), *US GeoData Digital Elevation Models*, Fact Sheet 040-00: Reston, Virginia.

U.S. Geological Survey (USGS) (2011), *Shuttle Radar Topography Mission (SRTM) – Finished*, http://eros.usgs.gov/#/Find_Data/Products_and_Data_Available/SRTM.

Veness, C. (2010), Movable Type Scripts: Calculate Distance, Bearing and More between Latitude / Longitude Points, http://www.movable-type.co.uk/scripts/latlong.html.

Vincenty, T. (1975), *Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations.* Survey review, Directorate of Overseas Surveys of the Ministry of Overseas Development: Kingston Road, Tolworth, Surrey.

APPENDICES

APPENDIX A: GLOSSARY

AABB: Axis Aligned Bounding Box

Computer graphics: a process of producing pictures or images using computers.

Datum: the position of the spheroid related to the center of the Earth.

DEM: Digital Elevation Model is a digital model or 3D representation of cartographic information of terrain surfaces in a raster form

FCC: Federal Communications Commission is an organization that regulates interstate and international communications by radio, television, wire, satellite and cable in all 50 states, the District of Columbia and U.S. territories.

Geographic coordinate system: a reference system to define the location of points on the Earth specified by a set of numbers such as latitude and longitude.

Information visualization: generally applied to the visual representation of large-scale collections of non-numerical information, such as files and lines of code in software system, library and bibliographic databases, networks of relations on the internet, and so forth.

KML: an XML grammar and file format for modeling and storing geographic features such as points, lines, images, polygons, and models for display in Google Earth or Google Maps.

Mercator projection: a cylindrical map projection that assumes a cylinder is wrapped around the globe so that its surface touches the equator; the meridians of longitude can be projected onto the cylinder as equally spaced straight lines perpendicular to the equator.

NASA: National Aeronautics and Space Administration is the federal agency that institutes and administers the civilian programs of the U.S. government that deals with aeronautical research and the development of launch vehicles and spacecraft.

SketchUP: a software tool that combines a tool-set with an intelligent drawing system for modeling of 3D objects.

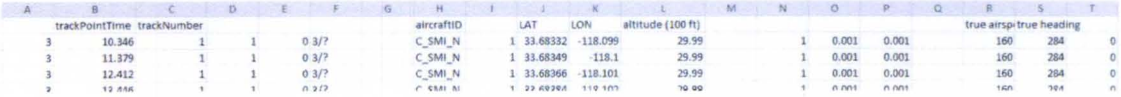
USGS: The U.S. Geological Survey is a science organization that provides impartial information on the health of our ecosystems and environment, the natural hazards that threaten us, the natural resources we rely on, the impacts of climate and land-use change, and the core science systems that help us provide timely, relevant, and useable information.

Visualization: the creation of information to show the important matters and is used as an effective way of the communication.

3D computer graphics: a three dimensional representation of geometric data.

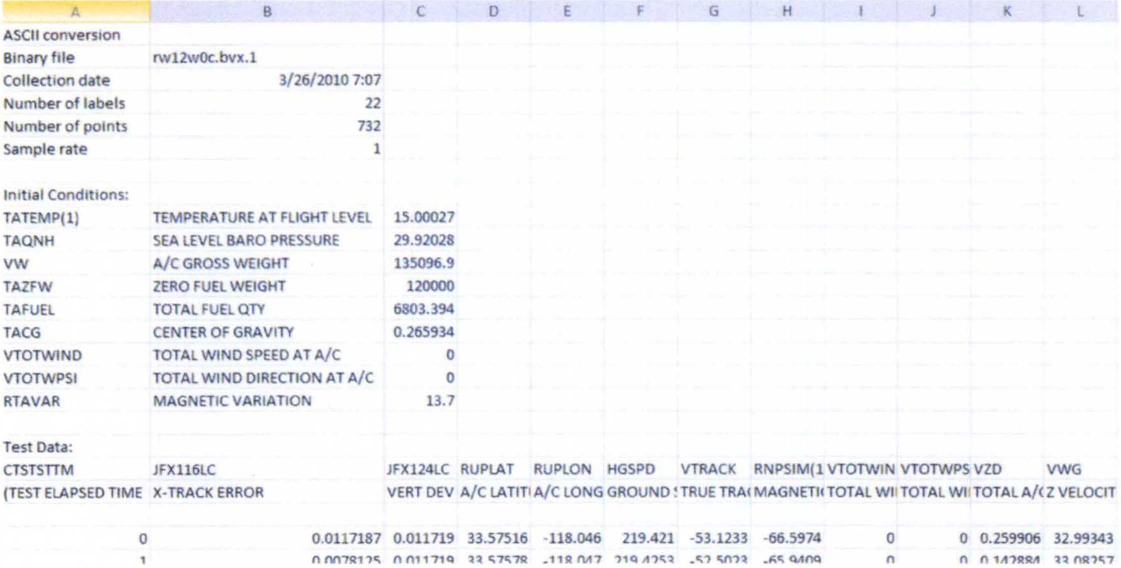
APPENDIX B: SAMPLES OF IN-FLIGHT RECORDED DATA

Model A with Benchmark



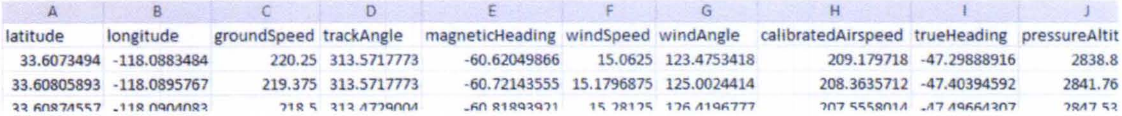
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	trackPointTime	trackNumber					aircraftID		LAT	LON	altitude (100 ft)						true airsp	true heading	
3	10.346	1	1	0	3/?		C_SMI_N	1	33.68332	-118.099	29.99		1	0.001	0.001		160	284	0
3	11.379	1	1	0	3/?		C_SMI_N	1	33.68349	-118.1	29.99		1	0.001	0.001		160	284	0
3	12.412	1	1	0	3/?		C_SMI_N	1	33.68366	-118.101	29.99		1	0.001	0.001		160	284	0

Model A with Simulation



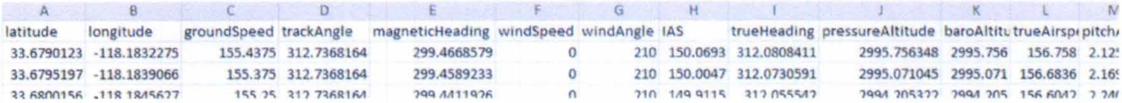
A	B	C	D	E	F	G	H	I	J	K	L
ASCII conversion											
Binary file	rw12w0c.bvx.1										
Collection date	3/26/2010 7:07										
Number of labels	22										
Number of points	732										
Sample rate	1										
Initial Conditions:											
TATEMP(1)	TEMPERATURE AT FLIGHT LEVEL	15.00027									
TAQNH	SEA LEVEL BARO PRESSURE	29.92028									
VW	A/C GROSS WEIGHT	135096.9									
TAZFW	ZERO FUEL WEIGHT	120000									
TAFUEL	TOTAL FUEL QTY	6803.394									
TACG	CENTER OF GRAVITY	0.265934									
VTOTWIND	TOTAL WIND SPEED AT A/C	0									
VTOTWPSI	TOTAL WIND DIRECTION AT A/C	0									
RTAVAR	MAGNETIC VARIATION	13.7									
Test Data:											
CTSTSTTM	JFX116LC	JFX124LC	RUPLAT	RUPLON	HGSPD	VTRACK	RNPSIM(1	VTOTWIN	VTOTWPS	VZD	VWG
(TEST ELAPSED TIME	X-TRACK ERROR	VERT DEV	A/C LATIT	A/C LONG	GROUND	TRUE TRA	MAGNETI	TOTAL WI	TOTAL WI	TOTAL A/	Z VELOCIT
0	0.0117187	0.011719	33.57516	-118.046	219.421	-53.1233	-66.5974	0	0	0.259906	32.99343

Model B with Benchmark



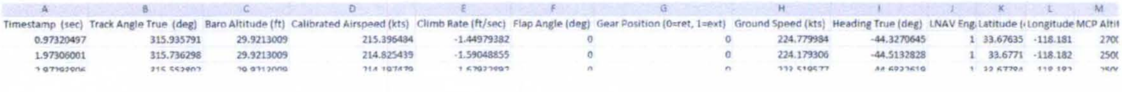
A	B	C	D	E	F	G	H	I	J
latitude	longitude	groundSpeed	trackAngle	magneticHeading	windSpeed	windAngle	calibratedAirspeed	trueHeading	pressureAltit
33.6073494	-118.0883484	220.25	313.5717773	-60.62049866	15.0625	123.4753418	209.179718	-47.29888916	2838.8
33.60805893	-118.0895767	219.375	313.5717773	-60.72143555	15.1796875	125.0024414	208.3635712	-47.40394592	2841.76

Model B with Simulation



A	B	C	D	E	F	G	H	I	J	K	L	M
latitude	longitude	groundSpeed	trackAngle	magneticHeading	windSpeed	windAngle	IAS	trueHeading	pressureAltitude	baroAltitu	trueAirsp	pitch
33.6790123	-118.1832275	155.4375	312.7368164	299.4668579	0	210	150.0693	312.0808411	2995.756348	2995.756	156.758	2.12
33.6795197	-118.1839066	155.375	312.7368164	299.4589233	0	210	150.0047	312.0730591	2995.071045	2995.071	156.6836	2.16

Model C



A	B	C	D	E	F	G	H	I	J	K	L	M
Timestamp (sec)	Track Angle True (deg)	Baro Altitude (ft)	Calibrated Airspeed (kts)	Climb Rate (ft/sec)	Flap Angle (deg)	Gear Position (0=ret, 1=ext)	Ground Speed (kts)	Heading True (deg)	LNAV Eng	Latitude (	Longitude	MCP Altit
0.97320497	315.935791	29.9213009	215.396484	-1.44979382	0	0	224.779984	-44.3270645	1	33.67635	-118.181	270
1.97306001	315.736298	29.9213009	214.825439	-1.59048855	0	0	224.179306	-44.5132828	1	33.6771	-118.182	250

APPENDIX C: SAMPLES OF GENERATED KML CODES

Basic structure

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns:gx="http://www.google.com/kml/ext/2.2" xmlns:kml="http://www.opengis.net/kml/2.2"
xmlns:atom="http://www.w3.org/2005/Atom">
  <Document>
 ....
 ....
 ....
  </Document>
 </kml>
```

Define a model (aircrafts, clouds, trails and so on)

```
 ....
    <Model id="Aircraft_Model1">
     <altitudeMode>relativeToGround</altitudeMode>
     <Location id="planeLocation">
      <longitude>-118.211433</longitude>
      <latitude>33.700943</latitude>
      <altitude>912.621056136</altitude>
     </Location>
     <Orientation id="planeOri">
      <heading>132.964016</heading>
      <tilt>4.71051025</tilt>
      <roll>6.20395184</roll>
     </Orientation>
     <Scale>
      <x>6</x>
      <y>6</y>
      <z>6</z>
     </Scale>
     <Link>
      <href>White_Jazz_Bombardier.dae</href>
      <refreshMode>
      </refreshMode>
      <refreshInterval>
      </refreshInterval>
      <viewRefreshMode>
      </viewRefreshMode>
      <viewRefreshTime>
      </viewRefreshTime>
     </Link>
    </Model>
 ....
```

Path with Altitude

```
....
   <Style>
    <LineStyle>
     <color>FF8000FF</color>
     <width>3</width>
    </LineStyle>
    <PolyStyle>
      <color>40D7300D</color>
    </PolyStyle>
   </Style>
   <LineString>
    <altitudeMode>absolute</altitudeMode>
    <tessellate>1</tessellate>
    <extrude>1</extrude>
    <coordinates>
    -118.06311,33.6427994,911.901621456
    -118.062698,33.6439285,912.097778544
    ....
    </coordinates>
   </LineString>
....
```

Visibility

```
....
<ScreenOverlay>
        <name>Visibility</name>
        <color>a0c0c0c0</color>
        <overlayXY x="0" y="1" xunits="fraction" yunits="fraction"/>
        <screenXY x="0" y="1" xunits="fraction" yunits="fraction"/>
        <rotationXY x="0" y="0" xunits="fraction" yunits="fraction"/>
        <size x="1" y="1" xunits="fraction" yunits="fraction"/>
</ScreenOverlay>
....
```

Animation

```
....
  <gx:Tour>
   <name>PlayMe</name>
   <gx:Playlist>
    <!-- Fly to our start location -->
    <gx:FlyTo>
     <gx:duration>0.2</gx:duration>
     <LookAt>
      <altitudeMode>relativeToGround</altitudeMode>
      <longitude>-118.211433</longitude>
      <latitude>33.7010329321606</latitude>
      <altitude>912.621056136</altitude>
      <range>500</range>
      <heading>0</heading>
      <tilt>0</tilt>
     </LookAt>
    </gx:FlyTo>

    <!-- Move Model  -->
    <gx:AnimatedUpdate>
     <gx:duration>0.2</gx:duration>
     <Update>
      <targetHref />
      <Change>
       <Location targetId="planeLocation">
        <longitude>-118.212349</longitude>
        <latitude>33.701664</latitude>
        <altitude>912.401981136</altitude>
       </Location>
       <Orientation targetId="planeOri">
        <heading>133.546875</heading>
        <roll>9.08416939</roll>
        <tilt>4.73684216</tilt>
       </Orientation>
      </Change>
     </Update>
    </gx:AnimatedUpdate>

    <!-- Move the camera in parallel with the model -->
    <gx:FlyTo>
     <gx:duration>0.2</gx:duration>
     <gx:flyToMode>smooth</gx:flyToMode>
     <LookAt>
      <altitudeMode>relativeToGround</altitudeMode>
      <longitude>-118.212349</longitude>
      <latitude>33.7017539321606</latitude>
      <altitude>912.621056136</altitude>
      <range>500</range>
      <heading>0</heading>
      <tilt>0</tilt>
     </LookAt>
    </gx:FlyTo>

   .... Repeat <-Move Model -> and <!-- Move the camera in parallel with the model --> part until the end of the data
   </gx:Playlist>
  </gx:Tour>
  <Style id="Style_-220478772">
   <BalloonStyle>
    <bgColor>FFFFFFFF</bgColor>
    <textColor>FF000000</textColor>
   </BalloonStyle>
  </Style>
....
```

CURRICULUM VITAE

Gyuchoon Cho

Personal Information

Date of Birth: February 9th, 1975
Place of Birth: DaeJeoun, Korea
Citizenship: Korea
E-mail: g0cho001@louisville.edu
Phone Number: 502-779-0803

ACADEMIC TRAINING

Aug. 2009- Present	Ph.D., Computer Science and Engineering University of Louisville, KY, U.S.A
Aug. 2007- May. 2009	M.S., Computer Science Western Kentucky University, KY, U.S.A
Aug. 2002- May. 2006	B.S, Computer Science Armstrong Atlantic State University, GA, U.S.A

TEACHING EXPERIENCE (Graduate Teaching Assistant)

- Fall 2010: **Introduction to Programming**, Department of Computer Engineering and Computer Science, University of Louisville, KY.
- Summer 2010: **Computer Compiler**, Department of Computer Engineering and Computer Science, University of Louisville, KY.
- Spring 2009: **Computer Information System Design**, Department of Computer Science, Western Kentucky University, KY.
- Spring 2009: **Computer Science I Laboratory**, Department of Computer Science, Western Kentucky University, KY.
- Spring 2009: **Computer Science I**, Department of Computer Science, Western Kentucky University, KY.
- Spring 2009: **Introduction to Computing**, Department of Computer Science, Western Kentucky University, KY.
- Fall 2008: **Computer Systems Hardware and Software II**, Department of Computer Science, Western Kentucky University, KY.
- Fall 2008: **Database Management System**, Department of Computer Science, Western Kentucky University, KY

PROFESSIONAL EXPERIENCE

December 2010 – Present
Jefferson County Public Schools
Louisville, KY in USA
User Interface Analyst: Develop applications for the end-user and manage database.

January 2010 – December 2010
Department of Computer Engineering and Computer Science
University of Louisville, KY in USA
Teaching Assistance

January 2010 – April 2010
Resources for Academic Achievement (REACH)
University of Louisville, KY in USA
Computer Tutor: Provides help with basic Computer Programming and other selected courses. Stays up to date with required computer knowledge

September 2008 – May 2009
A.I. Lab
Western Kentucky University
A.I. Lab Manager (Graduate Assistant): Assist students with projects and introduce them to robot programming, demonstrate robots and programs to visitors and faculty, and personally build, design, program, and test robots.

January 2004 – December 2005
Office of Video Conferencing
Armstrong Atlantic State University (Savannah, GA)
Facilitator: Served on 10-member team of facilitators providing technical support for academic classes, conferences, interviews, and meetings that used video conferencing.

November 2000 – December 2000
Inzen, Inc (Seoul, Korea)
Delphi Programmer: Participated on development team creating GUIs and Internet Security Modules.

November 1999– November 2000
Nilex, Inc (Seoul, Korea)
Delphi programming Team Leader: Recruited to lead 5-member development team managing projects, and timelines to ensure timely project delivery.

August 1999 – November 1999

MultiClass, Inc (Dea-Jeon, Korea)

Delphi Programmer: Recruited as sole programmer to create educational program.

PUBLICATIONS

[1]. Roman V. Yampolskiy, Gyuchoon Cho, Richard Rosenthal and Marina L. gavrilova, "Evaluation of Face Recognition Algorithms on Avatar Face Datasets" Cyberworlds 2011 International Conference on Cyberworlds, Oct. 4-6, 2011

[2]. Gyuchoon Cho, "Flight simulation Using Google Earth" The 2010 Kentucky Academy of Science (KSA) Annual Meeting, Bowling Green, KY, Nov 12-13, 2010. (3rd place)

[3]. Ahmed Emam and Gyuchoon Cho, "Real-Time Driver Safety System" The 2009 International Conference on Embedded Systems and Applications, Las Vegas NV, ESA'09: July 13-16, 2009.

[4]. Gyuchoon Cho and Ahmed Emam, "Driver Safety System", 39th Annual WKU Student Research Conference, Western Kentucky University, February 21, 2009.

[5]. Gyuchoon Cho and Ahmed Emam, "Real-time Visual Motion Detection System", 38th Annual WKU Student Research Conference, Western Kentucky University, April 12, 2008 .

[6]. Gyuchoon Cho and Ahmed Emam, "Real-time Visual Motion Detection System", ACM Southeast Conference, ACMSE 2008, Auburn University, March 28-29 2008.

RESEARCH PROJECTS

- Development of algorithms for "Medical Operation Simulation Using Wii" P.I.: Ming Ouyang, Adel Elmaghraby (Development language: C#)
- Development of algorithms for "Automatic Identification of Artificial Entities such as Robots, Bots, and Avatars." Research fellow, P.I.: Roman V. Yampolskiy (Development language: C++, Delphi).
- Development of algorithms for "Online ER-1 Robot Control System." Research fellow, P.I.: Art Shindhelm (Development language: Java, Delphi).
- Development of algorithms for "Real Time Drowsy Eye Detection." Research fellow, P.I.: Ahmed Emam (Development language: Delphi).
- Development of algorithms for "Xbox Game Development with Face Control." Research fellow, P.I.: Dar-jen Chang (Development language: C#).

- Development of algorithms for "Real-Time Motion Detection System." TechFest 2006 3rd place. (Development language: Delphi)
- Development of algorithm for "Mobile Game" TechFest 2005 2nd place (Development language: J2ME, MathFP)

PERSONAL ACHIEVEMENTS

- 3rd place at The 2010 Kentucky Academy of Science (KSA) Annual Meeting
- ACM Programming Contest 2005: Florida Institute of Technology
- AASU Student Writing Award: 2005
- 2nd place at Tech-Fest (www.TheCreativeCoast.org) : 2005 (Mobile game : J2ME, MathFP)
- 3rd place at Tech-Fest (www.TheCreativeCoast.org) : 2006 (Motion detection program using Webcam : Delphi)
- Awarded the first excellent degree by Korean Small and Medium Business Administration (9/7/2000)