A Google Earth-Based Framework for Visualization of the Chesapeake Bay Operational Forecast System

Gary Lawson
*Old Dominion University*

Follow this and additional works at: [https://digitalcommons.odu.edu/msve_etds](https://digitalcommons.odu.edu/msve_etds)

Part of the Computational Engineering Commons, Computer Sciences Commons, Geographic Information Sciences Commons, and the Meteorology Commons

**Recommended Citation**

Lawson, Gary. "A Google Earth-Based Framework for Visualization of the Chesapeake Bay Operational Forecast System" (2011). Master of Science (MS), Thesis, Computational Modeling & Simulation Engineering, Old Dominion University, DOI: 10.25777/7vg0-nc18

[https://digitalcommons.odu.edu/msve_etds/99](https://digitalcommons.odu.edu/msve_etds/99)

This Thesis is brought to you for free and open access by the Computational Modeling & Simulation Engineering at ODU Digital Commons. It has been accepted for inclusion in Computational Modeling & Simulation Engineering Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.
A GOOGLE EARTH-BASED FRAMEWORK FOR VISUALIZATION
OF THE Chesapeake Bay OPERATIONAL FORECAST
SYSTEM

by

Gary Lawson
B.S. May 2010, Old Dominion University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
MODELING AND SIMULATION
OLD DOMINION UNIVERSITY
December 2011

Approved by:

Yuzhong Shen (Director)

Frederic D. McKenzie (Member)

Jiang Li (Member)
ABSTRACT

A GOOGLE EARTH-BASED FRAMEWORK FOR VISUALIZATION OF THE CHESAPEAKE BAY OPERATIONAL FORECAST SYSTEM

Gary Lawson
Old Dominion University, 2011
Director: Dr. Yuzhong Shen

For the persons who live near and travel the waters of the Chesapeake Bay, the data provided by the Chesapeake Bay Operational Forecast System (CBOFS) is invaluable. The information provided includes measurements and forecasts of surface wind velocity, water current velocity, salinity levels, water level, and temperature. Currently, this information is freely available on the CBOFS website hosted by the National Oceanographic and Atmospheric Administration (NOAA). It is offered as Nowcast, measured data, and Forecast data and is visualized using 2D images which describe a subset of the data in an easy to read chart. However, if the data were made available in a 3D environment, it would not only provide viewers with the general information currently available on the CBOFS website, but it would also provide users with a means to explore the data in the rich context of the surrounding environment. Viewers would have the ability to look at the data from any viewpoint, zoom in to see individual markers, and receive actual measurements in moments. This would considerably increase the use of the data measured by CBOFS as it would allow those who rely on this information to still receive the same data in a similar format, but it would also provide those interested in the Chesapeake Bay a means to learn more about the bay and how it functions under the water's surface.
This thesis proposes a framework for developing interactive visualizations of CBOFS data in Google Earth by exporting the CBOFS data into KML files which may be loaded into Google Earth from a standalone application or web plugin. Generation of these KML files involves retrieving the data files provided from OPeNDAP servers which host the CBOFS data files in NetCDF format. The pertinent raw data is then extracted from these data files using the Unidata Java library. The raw data is further processed based on the grids utilized by CBOFS, and different visualization methods are employed to visualize different CBOFS variables. The final visualization results are then converted to KML files for efficient rendering in Google Earth. Finally, the KML files are organized into a carefully designed hierarchical structure and archived on a server for easy retrieval by the website designed to display the CBOFS data using Google Earth. The proposed framework provides several advantages over the current CBOFS visualization at the NOAA website: 1) it produces visualizations that are not available in the current CBOFS visualization; 2) it produces visualizations with resolution and accuracy much higher than that of the current CBOFS visualization; and 3) it provides users interactive capabilities that essentially do not exist in the current CBOFS visualization. Researchers, educators, students, and the general public will benefit greatly from the proposed framework.
This thesis is dedicated to someone whom I highly respect and look up to, my father.
ACKNOWLEDGMENTS

Throughout this thesis I have received many people’s support, time, and expertise which have guided me towards its completion; these people deserve special recognition. First, I’d like to extend much gratitude and many thanks to my advisor, Dr. Yuzhong Shen, for his guidance, expertise, and patience throughout my studies and editing of this manuscript. He has always been there when I needed him, and I extend my sincerest appreciations for his efforts. I’d also like extend my appreciations to Dr. Rick McKenzie and Dr. Jiang Li for serving as my thesis committee members. They have offered me their time, attention, and advice to review this thesis, and for that I am sincerely grateful. Finally, I’d like to extend many thanks to my family for their love and support. Without them I would have never made it this far.
TABLE OF CONTENTS

LIST OF FIGURES........................................................................................................ viii
LIST OF TABLES........................................................................................................... xiii

1. INTRODUCTION ........................................................................................................ 1
   1.1 Motivation........................................................................................................... 1
   1.2 Completed Work................................................................................................ 5
   1.3 Thesis Organization............................................................................................ 10

2. BACKGROUND AND RELATED WORK .................................................................. 11
   2.1 Numerical Ocean Modeling .............................................................................. 11
   2.2 Chesapeake Bay Operational Forecast System ................................................. 13
   2.3 Google Earth ..................................................................................................... 17
   2.4 Google Earth’s KML File ................................................................................... 22

3. CBOFS DATA PROCESSING ....................................................................................... 25
   3.1 Data Formats and Servers ................................................................................. 25
   3.2 Raw Data Extraction .......................................................................................... 27
   3.3 Data Preparation ................................................................................................ 32
   3.4 Data Processing .................................................................................................. 41

4. VISUALIZATION ....................................................................................................... 47
   4.1 Current CBOFS Data Visualization ................................................................... 47
   4.2 Visualization in Google Earth ............................................................................ 53
   4.3 CBOFS Data Representation in Google Earth .................................................... 60

5. IMPLEMENTATION, RESULTS, AND DISCUSSIONS .............................................. 67
   5.1 Implementation .................................................................................................... 67
   5.2 Results ............................................................................................................... 69
   5.3 Discussions ........................................................................................................ 80

6. CONCLUSIONS AND FUTURE WORK .................................................................. 82

REFERENCES ............................................................................................................. 83

APPENDIXES
   A. Salinity Mesh ..................................................................................................... 87
   B. Temperature Mesh .............................................................................................. 92
   C. Water Current ..................................................................................................... 97

VITA .......................................................................................................................... 102
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Structure of the proposed CBOFS data visualization framework</td>
<td>6</td>
</tr>
<tr>
<td>2. The curvilinear grid used by the Chesapeake Bay Operational Forecast System</td>
<td>15</td>
</tr>
<tr>
<td>3. Bathymetry of the Chesapeake Bay</td>
<td>16</td>
</tr>
<tr>
<td>4. Planet Earth (a) and the amphitheater in Rome, Italy (b) as seen in Google Earth Google Earth’s standalone application</td>
<td>18</td>
</tr>
<tr>
<td>5. Google Earth’s standalone application</td>
<td>19</td>
</tr>
<tr>
<td>6. Examples of some of Google Earth’s uses: (a) Architecture &amp; Engineering (b) Real Estate &amp; Insurance (c) Media (d) Energy &amp; Utilities</td>
<td>21</td>
</tr>
<tr>
<td>7. A look at Old Dominion University from two different levels of detail: (a) overlooks Norfolk (b) Old Dominion University Campus</td>
<td>22</td>
</tr>
<tr>
<td>8. Simple KML Example</td>
<td>23</td>
</tr>
<tr>
<td>9. Example KML objects in Google Earth</td>
<td>23</td>
</tr>
<tr>
<td>10. Panoply displaying Surface Wind information</td>
<td>28</td>
</tr>
<tr>
<td>11. Example data for the U component of Surface Wind</td>
<td>29</td>
</tr>
<tr>
<td>12. THREDDS Catalog Server Variable List</td>
<td>33</td>
</tr>
<tr>
<td>13. Directory Hierarchy Diagram</td>
<td>34</td>
</tr>
<tr>
<td>14. The CBOFS KML Exporter GUI</td>
<td>36</td>
</tr>
<tr>
<td>15. CBOFS KML Exporter Application with Options Shown</td>
<td>38</td>
</tr>
<tr>
<td>16. Vector Components</td>
<td>40</td>
</tr>
<tr>
<td>17. CBOFS Curvilinear Grid with Reference Angle</td>
<td>42</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>18. Angle Quadrants and their relation to U and V</td>
<td>43</td>
</tr>
<tr>
<td>19. Orientation in Google Earth</td>
<td>44</td>
</tr>
<tr>
<td>20. ‘u’ and ‘v’ component offset from normal grid</td>
<td>46</td>
</tr>
<tr>
<td>21. Water Levels Nowcast</td>
<td>48</td>
</tr>
<tr>
<td>22. Surface Winds Nowcast</td>
<td>49</td>
</tr>
<tr>
<td>23. Field Surface Water Temperature Nowcast</td>
<td>50</td>
</tr>
<tr>
<td>24. Field Surface Water Salinity Nowcast</td>
<td>51</td>
</tr>
<tr>
<td>25. Field Surface Water Currents Nowcast</td>
<td>52</td>
</tr>
<tr>
<td>26. Placemark Point Icons with Names</td>
<td>54</td>
</tr>
<tr>
<td>27. Orientation of a Model in Google Earth</td>
<td>56</td>
</tr>
<tr>
<td>28. Example KML file</td>
<td>59</td>
</tr>
<tr>
<td>29. Wireframe Mesh</td>
<td>63</td>
</tr>
<tr>
<td>30. Color smoothing example part 1</td>
<td>64</td>
</tr>
<tr>
<td>31. Color smoothing example part 2</td>
<td>65</td>
</tr>
<tr>
<td>32. Color smoothing example part 3</td>
<td>65</td>
</tr>
<tr>
<td>33. Implementation Overview</td>
<td>67</td>
</tr>
<tr>
<td>34. Website for CBOFS visualization in Google Earth</td>
<td>69</td>
</tr>
<tr>
<td>35. Water Level</td>
<td>71</td>
</tr>
<tr>
<td>36. Surface Wind Velocity</td>
<td>72</td>
</tr>
<tr>
<td>37. Temperature</td>
<td>73</td>
</tr>
<tr>
<td>38. Temperature Mesh</td>
<td>74</td>
</tr>
<tr>
<td>39. Salinity</td>
<td>75</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>40. Salinity Mesh</td>
<td>76</td>
</tr>
<tr>
<td>41. Water Current</td>
<td>77</td>
</tr>
<tr>
<td>42. Exploring Water Current</td>
<td>78</td>
</tr>
<tr>
<td>43. Exploring Salinity</td>
<td>79</td>
</tr>
<tr>
<td>44. Exploring Temperature Mesh</td>
<td>80</td>
</tr>
<tr>
<td>45. Salinity Mesh Depth -0.025</td>
<td>87</td>
</tr>
<tr>
<td>46. Salinity Mesh Depth -0.075</td>
<td>87</td>
</tr>
<tr>
<td>47. Salinity Mesh Depth -0.125</td>
<td>87</td>
</tr>
<tr>
<td>48. Salinity Mesh Depth -0.175</td>
<td>87</td>
</tr>
<tr>
<td>49. Salinity Mesh Depth -0.225</td>
<td>88</td>
</tr>
<tr>
<td>50. Salinity Mesh Depth -0.275</td>
<td>88</td>
</tr>
<tr>
<td>51. Salinity Mesh Depth -0.325</td>
<td>88</td>
</tr>
<tr>
<td>52. Salinity Mesh Depth -0.375</td>
<td>88</td>
</tr>
<tr>
<td>53. Salinity Mesh Depth -0.425</td>
<td>89</td>
</tr>
<tr>
<td>54. Salinity Mesh Depth -0.475</td>
<td>89</td>
</tr>
<tr>
<td>55. Salinity Mesh Depth -0.525</td>
<td>89</td>
</tr>
<tr>
<td>56. Salinity Mesh Depth -0.575</td>
<td>89</td>
</tr>
<tr>
<td>57. Salinity Mesh Depth -0.625</td>
<td>90</td>
</tr>
<tr>
<td>58. Salinity Mesh Depth -0.675</td>
<td>90</td>
</tr>
<tr>
<td>59. Salinity Mesh Depth -0.725</td>
<td>90</td>
</tr>
<tr>
<td>60. Salinity Mesh Depth -0.775</td>
<td>90</td>
</tr>
<tr>
<td>61. Salinity Mesh Depth -0.825</td>
<td>91</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>62. Salinity Mesh Depth -0.875</td>
<td>91</td>
</tr>
<tr>
<td>63. Salinity Mesh Depth -0.925</td>
<td>91</td>
</tr>
<tr>
<td>64. Salinity Mesh Depth -0.975</td>
<td>91</td>
</tr>
<tr>
<td>65. Temperature Mesh Depth -0.025</td>
<td>92</td>
</tr>
<tr>
<td>66. Temperature Mesh Depth -0.075</td>
<td>92</td>
</tr>
<tr>
<td>67. Temperature Mesh Depth -0.125</td>
<td>92</td>
</tr>
<tr>
<td>68. Temperature Mesh Depth -0.175</td>
<td>92</td>
</tr>
<tr>
<td>69. Temperature Mesh Depth -0.225</td>
<td>93</td>
</tr>
<tr>
<td>70. Temperature Mesh Depth -0.275</td>
<td>93</td>
</tr>
<tr>
<td>71. Temperature Mesh Depth -0.325</td>
<td>93</td>
</tr>
<tr>
<td>72. Temperature Mesh Depth -0.375</td>
<td>93</td>
</tr>
<tr>
<td>73. Temperature Mesh Depth -0.425</td>
<td>94</td>
</tr>
<tr>
<td>74. Temperature Mesh Depth -0.475</td>
<td>94</td>
</tr>
<tr>
<td>75. Temperature Mesh Depth -0.525</td>
<td>94</td>
</tr>
<tr>
<td>76. Temperature Mesh Depth -0.575</td>
<td>94</td>
</tr>
<tr>
<td>77. Temperature Mesh Depth -0.625</td>
<td>95</td>
</tr>
<tr>
<td>78. Temperature Mesh Depth -0.675</td>
<td>95</td>
</tr>
<tr>
<td>79. Temperature Mesh Depth -0.725</td>
<td>95</td>
</tr>
<tr>
<td>80. Temperature Mesh Depth -0.775</td>
<td>95</td>
</tr>
<tr>
<td>81. Temperature Mesh Depth -0.825</td>
<td>96</td>
</tr>
<tr>
<td>82. Temperature Mesh Depth -0.875</td>
<td>96</td>
</tr>
<tr>
<td>83. Temperature Mesh Depth -0.925</td>
<td>96</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>84. Temperature Mesh Depth -0.975</td>
<td>96</td>
</tr>
<tr>
<td>85. Water Current Mesh Depth -0.025</td>
<td>97</td>
</tr>
<tr>
<td>86. Water Current Mesh Depth -0.075</td>
<td>97</td>
</tr>
<tr>
<td>87. Water Current Mesh Depth -0.125</td>
<td>97</td>
</tr>
<tr>
<td>88. Water Current Mesh Depth -0.175</td>
<td>97</td>
</tr>
<tr>
<td>89. Water Current Mesh Depth -0.225</td>
<td>98</td>
</tr>
<tr>
<td>90. Water Current Mesh Depth -0.275</td>
<td>98</td>
</tr>
<tr>
<td>91. Water Current Mesh Depth -0.325</td>
<td>98</td>
</tr>
<tr>
<td>92. Water Current Mesh Depth -0.375</td>
<td>98</td>
</tr>
<tr>
<td>93. Water Current Mesh Depth -0.425</td>
<td>99</td>
</tr>
<tr>
<td>94. Water Current Mesh Depth -0.475</td>
<td>99</td>
</tr>
<tr>
<td>95. Water Current Mesh Depth -0.525</td>
<td>99</td>
</tr>
<tr>
<td>96. Water Current Mesh Depth -0.575</td>
<td>99</td>
</tr>
<tr>
<td>97. Water Current Mesh Depth -0.625</td>
<td>100</td>
</tr>
<tr>
<td>98. Water Current Mesh Depth -0.675</td>
<td>100</td>
</tr>
<tr>
<td>99. Water Current Mesh Depth -0.725</td>
<td>100</td>
</tr>
<tr>
<td>100. Water Current Mesh Depth -0.775</td>
<td>100</td>
</tr>
<tr>
<td>101. Water Current Mesh Depth -0.825</td>
<td>101</td>
</tr>
<tr>
<td>102. Water Current Mesh Depth -0.875</td>
<td>101</td>
</tr>
<tr>
<td>103. Water Current Mesh Depth -0.925</td>
<td>101</td>
</tr>
<tr>
<td>104. Water Current Mesh Depth -0.975</td>
<td>101</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. KML file data types and their appropriate variables</td>
<td>8</td>
</tr>
<tr>
<td>2. Array sizes for Variables ‘u’ and ‘v’</td>
<td>45</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

This chapter provides an introduction to this research and describes its motivation, completed work, and contributions. The chapter wraps up by providing an overview of the organization of the thesis.

1.1 Motivation

Human civilization has been dependent on Mother Nature from its very conception. Despite the profound power that has been achieved and demonstrated by human civilization, human beings are often helpless and at the mercy of Mother Nature in many natural disasters, such as tsunamis, hurricanes, and earthquakes. Many disciplines have been established to study natural phenomena, such as meteorology and oceanography, to provide guidance and forecast for better and safe utilization of natural environments. With advances in computing hardware and software, numerical modeling is playing a more and more important role in many disciplines to acquire quantitative insight of natural environments and phenomena. In the field of oceanography, many ocean circulation models have been developed using different numerical methods (e.g. finite difference, finite element, and finite volume) and vertical discretization methods (e.g. layered models and terrain-following vertical coordinates) [1]. Among them, the Regional Ocean Model System (ROMS) is a free-surface, hydrostatic, primitive equation ocean model that uses stretched, terrain-following coordinates in the vertical and orthogonal curvilinear coordinates in the horizontal [1]. As its name indicates, ROMS is widely used for modeling of relatively small regions of the ocean water, such as the Chesapeake Bay. The Chesapeake Bay Operational Forecast System (CBOFS) uses
ROMS to generate measured and forecasted oceanographic data about Chesapeake Bay, the largest estuary in the United States [2] with a drainage basin covering Washington D.C., New York, Pennsylvania, Delaware, Maryland, Virginia, and West Virginia [3].

As is common in scientific studies, numerical ocean modeling produces a large amount of data that need to be analyzed and disseminated. For example, CBOFS utilizes a curvilinear grid that contains more than 90,000 points, and each point has five variables representing water level, wind, water temperature, salinity, and currents, which change at every hour. There are other numerous (tens of) other internal variables used for computing these outputs. How to interpret and understand the large amount of data produced in scientific studies presents a challenge due to the sheer amount of data. For example, a tabular print of one million numerical values on paper not only overwhelms the reader but also is an enormous waste of paper. For many scientific and engineering applications, visualization is the key approach for understanding results, if not the only way. Without visualization methods, the data and measurements would be rather difficult to comprehend quickly as people would have difficulty imagining the meaning and relationship between all of these points. Visualization assists in the comprehension of data; it allows those who analyze it to quickly pick out patterns and correlations between individual data points, as well as provide some sort of intuitive meaning for the values. While this may not provide a complete and precise analysis of the data, it would provide a quick method for intuitive explanations which cannot be obtained from lists of individual data points. Visualization is important in learning and analyzing measurements and forecasts which are the result of studying natural phenomenon; it is also important in
learning and analyzing the results of the Chesapeake Bay Operational Forecast System (CBOFS).

The data produced by ROMS is packaged into the Network Common Data Format (NetCDF) and hosted on the CBOFS website using both the THREDDS catalog servers and OPeNDAP data servers [4]. On the CBOFS website, users are provided information about recent and future conditions of the Chesapeake Bay including surface wind speeds, temperature, salinity, water currents, and water level. To many, this information is invaluable as it provides mariners and U.S. port authorities the information needed to keep them and their crews safe.

Currently, the CBOFS data are approximated and displayed as 2D images to fit within the bounds of a webpage. The CBOFS data contain approximately 20,000 points which describe the conditions of the bay on an hourly basis. For scalar data such as salinity, temperature, and water level, it’s feasible to display this data using color maps condensed and overlaid on a small image of the Chesapeake Bay; however, for vector data such as surface wind and water current, the data must be condensed to such an extreme as to only provide several hundred data points to describe the vast Chesapeake Bay. This provides a quick reference to draw conclusions; however, the sample of data is too small to be accurate for specific locations. Furthermore, it would be expected that with the size of the bay, the user might be able to zoom in to obtain a more complete representation of the circumstances within the bay at the selected time. However, currently there is very limited user interaction with the images. Users may view individual time periods in the form of a slide show animation but may not zoom in to see a finer grid from which more intuitive and beneficial conclusions may be drawn. In addition, some information
provided by CBOFS is not available in the current CBOFS visualization, such as salinity and temperature at different layers.

From the above discussion, it is clear that the visualization capabilities of CBOFS are insufficient and should be expanded to provide its users more complete data with higher resolution and accuracy in a more interactive way. It is also desirable to visualize CBOFS data in a 3D environment that allows the user to immerse oneself in the display and provides users with the ability to explore the data and make a stronger connection between physical locations and measured or theoretical data. The new visualization capabilities would provide users the ability to zoom in and out, as well as look at the location from an infinite number of different vantage points. The data may be viewed as a whole or as a single data point. Google Earth, a rich 3D environment that provides a plethora of geographical data and visualization capabilities, is ideally suited to accommodate the new CBOFS visualizations to be developed in this thesis [5].

Google Earth provides a unique experience with the virtual Earth. It allows data, such as that provided by CBOFS, to be visualized in 3D while simultaneously providing a connection between physical location and individual data points. Furthermore, Google Earth institutes an intuitive navigation control system involving use of the mouse or icons on the screen to control the camera. This camera provides the portal between the user and the data; it allows the user to explore and study the data provided. Google Earth is very popular with wide use in engineering, the media, and education, to name just a few. Ease of use and geographical accuracy make Google Earth a well-defined framework for visualization of data with geographical significance. In addition to these capabilities, Google Earth is supported as a standalone, web, or mobile application. When using
Google Earth on the web or as a standalone application, users may import models, icons, images, lines, or KML files which contain code to import any of the previously mentioned objects. Using KML files to import objects can be an easy method for incorporating data in Google Earth and representing it graphically using objects native to Google Earth, such as icons, lines, polygons, and images [5].

Visualizing the data provided by CBOFS in a more interactive, accurate, and complete manner is the goal of this thesis. The final result will be CBOFS data visualized using Google Earth. The completeness of the CBOFS data combined with the intuitive user interface of Google Earth contained in one easy to use package is the purpose of this thesis. Researchers, educators, students, and the general public will find varied use of the work in this thesis and greatly benefit from it.

1.2 Completed Work

The overall goal of the research presented by this thesis is to provide a more involved and encompassing visualization platform for viewing CBOFS data. The input is the CBOFS data file in NetCDF format; the outputs are KML files for each of the data types contained in the CBOFS file, which will be the input to the CBOFS Data Visualization Website. The overall structure of the proposed framework is illustrated in Figure 1; each component will be described briefly in the subsequent sections.
1.2.1 CBOFS Raw Data Extraction

CBOFS collects data from the Chesapeake Bay as well as the atmosphere to provide accurate measurements of the most recent conditions experienced within the bay, as well as to provide accurate predictions about conditions in the future. The data collected is compiled, processed, and exported, using the ocean modeling system ROMS, into NetCDF files for educational and scientific use [6]. To provide a higher resolution data set, the measurements are interpolated to approximate areas where measurements could not be taken. Once the data set is complete, the data is packaged into a Network Common Data Format (NetCDF) file [7]. The NetCDF file format is a collection of multidimensional data arrays [8] that is widely used in earth sciences which require the use of dense arrays of information that needs to be packaged and transferred for use by other educators and scientists [9].

The NetCDF files provided by CBOFS are distributed via the OPeNDAP server and THREDDS catalog server [4]. These servers host several days’ worth of data at any given time. Each file contains 7 hours’ worth of data, and there are 4 files for each day. The final hour in each file overlaps the first hour in the following file. The size of these files
varies depending on the type of data contained in the file. Data which had been measured at designated locations called stations has a file size of 7.7 MB, whereas forecasted data at these stations has a file size of 60.5 MB. Measured data across the entire Chesapeake Bay has a file size of 553.6 MB, whereas the forecasted data across the entire bay has a file size of 3.7 GB. This data is free to anyone for download and is also visualized using 2D images which approximate the data contained in the files for quick reference [7]. The raw data in the CBOFS files are extracted by an application program developed by the author based on the NetCDF-Java library [10].

1.2.2 KML File Generation

Google Earth allows users to add several different types of objects such as icons, lines, images, and models. These items may be added using commands present in the standalone application, by use of special javascript commands used by the web application, or by KML files which may be handled by both the standalone application and web application.

A Keyhole Markup Language (KML) file is essentially an XML file with specific keywords designated by Google for use with their Google Earth client. An XML file is a text file that uses tags to describe data contained within the file. The purpose of the XML format is to provide high readability; however, these files are generally much larger due to formatting and use of ASCII text for readability.

After the raw data are extracted from the CBOFS NetCDF files, further processing and calculations are needed to convert the original data based on a staggered grid in ROMS to new values in the geographical coordinate system used by Google Earth. Other
processing includes vector magnitude and direction calculation and unit conversions. Table 1 shows the list of CBOFS variables to be visualized in the KML files and other relevant variables required in order to read and process the final output variables.

<table>
<thead>
<tr>
<th>KML File Data Type</th>
<th>Variables Required for KML File</th>
<th>Calculations Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surface Wind</td>
<td>Uwind, Vwind, angle, lat_rho, lon_rho, mask_rho</td>
<td>Yes</td>
</tr>
<tr>
<td>Water Current</td>
<td>u, v, angle, lat_rho, lon_rho, mask_rho</td>
<td>Yes</td>
</tr>
<tr>
<td>Water Level</td>
<td>zeta, lat_rho, lon_rho, mask_rho</td>
<td>Yes</td>
</tr>
<tr>
<td>Salinity</td>
<td>salt, lat_rho, lon_rho, mask_rho</td>
<td>No</td>
</tr>
<tr>
<td>Temperature</td>
<td>temp, lat_rho, lon_rho, mask_rho</td>
<td>Yes</td>
</tr>
</tbody>
</table>

After all the variables are converted to formats that conform to Google Earth, appropriate visualization methods are selected to visualize different types of variables. The visualization process is substantiated in the generation of the KML files, which contain the graphical objects representing the CBOFS variables. Generating KML files beforehand has huge advantages over visualizing the CBOFS data on the fly because
instantaneous objects are drawn in Google Earth using Java Script, which presents performance issues due to its scripting nature.

1.2.3 CBOFS Data Visualization Website

Finally, the CBOFS data is visualized using the Google Earth web plugin or standalone application. The final result of this thesis is a website dedicated to visualizing the data exported by the application created by this research. This website will allow users to select from a list of dates, time periods, data types, and depths (if applicable) to visualize the CBOFS KML files using the Google Earth Web plugin. Once a user has selected the appropriate data file, it will load into the Google Earth viewer, and the user will be automatically teleported to view the data contained in the file. The loading process only takes a few moments as the data files are reasonably sized and formatted to optimize loading times. If another file is selected, the previous file is removed and the new file is loaded. This process only takes as long as the loading process as KML file removal is reasonably fast in Google Earth. Below the viewer and user interface will be brief information concerning CBOFS, Old Dominion University and the Department of Modeling, Simulation, and Visualization Engineering.

1.2.4 Contributions

The contributions of the proposed thesis research include an application which produces 3D models of CBOFS oceanographic data and a website for viewing and exploring this data using Google Earth. In particular, this thesis provides the following contributions:
An original method to generate 3D models of CBOFS data in the KML format from data provided by CBOFS in NetCDF format. This method provides 3D models of surface winds, water level, water current, salinity, and temperature using one or more visualization methods where applicable. All of these models may be visualized using Google Earth.

An original website for viewing the KML files generated by the proposed method in this thesis. The website allows users to view and explore CBOFS data in a virtual world to analyze and learn from the data as well as use it for general navigation purposes.

1.3 Thesis Organization

The remainder of this thesis is organized into the following chapters. Chapter 2 provides a detailed description of the Chesapeake Bay Operational Forecast System and Google Earth. Chapter 3 discusses data processing and extraction techniques. Chapter 4 details the visualization methods and techniques. Chapter 5 describes the implementation and results as well as the problems in the current visualization methods. Finally, Chapter 6 concludes this thesis and discusses future additions.
CHAPTER 2
BACKGROUND AND RELATED WORK

This chapter discusses the Chesapeake Bay Operational Forecast System and Google Earth in more detail. The chapter begins with the numerical ocean modeling system used by CBOFS to analyze oceanographic and atmospheric information and produce ocean models for surface winds, water currents, water level, salinity, and temperature for both previously measured data and forecast estimates, as well as the administration that designed and produced this invaluable resource, the National Oceanographic and Atmospheric Administration (NOAA). This chapter will discuss Google Earth, including both the standalone application and web plugin, and its capabilities and usage. Finally, the native KML file format used to represent data in Google Earth is described.

2.1 Numerical Ocean Modeling

Numerical ocean modeling is modeling of ocean climate and characteristics [11]. Amongst the field of ocean general circulation modeling is many different ocean models, each loosely characterized by their approach to spatial discretization and vertical coordinate treatment [1]. The ocean model of interest in this research, however, is a free-surface, hydrostatic, primitive equation ocean model by the name of ROMS, Regional Ocean Model System [12]. ROMS is an example of a layered and sigma-coordinate model as depth is broken down into several individual layers and elevation is represented by sigma [13]. Sigma represents depth as a fraction of the total depth between the sea floor and sea level [1]. Therefore, ROMS is described as terrain-following in the vertical and orthogonal curvilinear coordinates of the horizontal [12].
ROMS originated from algorithms based on the S-coordinate Rutgers University Model (SCRUM) created in 1994 but was rewritten to improve its numeric accuracy and efficiency [12]. ROMS was optimized to run efficiently on both single and multi-core architectures and was expanded to include several new features, such as high-order advection schemes and data assimilation [12]. Below provides some examples for the equations of motion used by ROMS to generate the ocean models. The balance of momentum in the x- and y- directions can be represented by the following [14]

\[
\frac{\partial u}{\partial y} + \nabla \cdot \mathbf{u} = -\frac{\partial \phi}{\partial x} - \frac{\partial}{\partial z} \left( u' w' - v \frac{\partial u}{\partial z} \right) + F_u + D_u
\]

\[
\frac{\partial v}{\partial y} + \nabla \cdot \mathbf{v} = -\frac{\partial \phi}{\partial y} - \frac{\partial}{\partial z} \left( v' w' - u \frac{\partial v}{\partial z} \right) + F_v + D_v
\]

where \( u, v, \) and \( w \) are the \((x, y, z)\) components of the vector velocity \( \mathbf{v} \), \( F \) is a forcing term with \( u \) and \( v \) components, \( D \) is a diffusive term with \( u \) and \( v \) components, \( \nu \) is the molecular viscosity and diffusivity, and \( f(x,y) \) is the Coriolis parameter.

The time evolution of a scalar concentration for salinity, temperature, and nutrients can be written as

\[
\frac{\partial C}{\partial t} + \nabla \cdot \mathbf{C} = -\frac{\partial}{\partial z} \left( C' w' - \nu_\theta \frac{\partial C}{\partial z} \right) + F_C + D_C
\]

where \( C(x, y, z, t) \) denotes the scalar concentration field and \( \nu_\theta \) is the molecular viscosity and diffusivity. It's also important to note that an overbar represents a time average and a prime represents a fluctuation about the mean.

The ROMS modeling software may be configured for open, closed, or periodic horizontal boundaries [15]. The model domain is rectangular; however, boundary conditions are masked, and boundaries may be either no-slip or free-slip walls [16]. When biharmonic friction is present, a higher order boundary condition must be
provided; however, the values for the biharmonic terms are calculated. The boundary conditions preserve the property of no gain or loss of volume-integrated momentum or scalar concentration [16].

2.2 Chesapeake Bay Operational Forecast System

Chesapeake Bay Operational Forecast System (CBOFS) is a real-time operational system which produces daily forecasts of water level, temperature, salinity and ocean currents in the Chesapeake Bay. This forecasting system exists to aid U.S. port authorities and mariners in efficiently navigating the Chesapeake Bay without compromising safety. This goal is accomplished by providing best estimates for expected water levels and speeds which are based on NOAA’s Tide Tables, surface winds, atmospheric pressure, and river flow. The tide tables are accurate predictions of the astronomical tide, the changes in water level due to the gravitational effects of the moon and sun as well as the rotation of the Earth [7]. CBOFS is the result of a joint project between the Center for Operational Oceanographic Products and Services (CO-OPS), the Office of Coast Survey, and the National Centers for Environmental Prediction (NCEP) Central Operations (NCO) using Rutgers University’s Regional Ocean Modeling System (ROMS) [7].

2.2.1 CBOFS Data

In an effort to provide better information to the maritime community, CBOFS was upgraded from providing strictly sea level data to include all data exported from ROMS. This change affected the information available to mariners as 19 additional layers of salinity, temperature, and water current information was now available where previously
only sea level measurements were available. Surface wind and water level are exceptions to the additional layers as they may only be measured at sea level. All of the measurement and forecast data exported from ROMS is based on a timescale of 1 hour increments [7] which is the same as the NOAA timescale for updating forecasting information [17]; therefore, CBOFS data is also based on a 1 hour timescale.

Currently, CBOFS provides Nowcast and Forecast Guidance models for the measured and computed oceanographic information: water level, surface wind velocity, salinity, temperature, and water current. Nowcast incorporates near real-time observed and calculated meteorological, oceanographic, and/or river flow data into a large-scale numerical model. The Nowcast covers a period of time of a couple days starting in the present and extending back to previous time periods. It also interpolates the data to fill in locations where observable data is not available. Forecast Guidance incorporates meteorological, oceanographic, and/or river flow rate forecasts to generate a large-scale forecast model. This model begins at the current state of the Nowcast and forecasts a couple days into the future [4].

2.2.2 CBOFS Curvilinear Grid System

Nowcast and Forecast Guidance models are based on a $332 \times 291$ curvilinear grid which is generated by CBOFS to display the model data for the Chesapeake Bay as shown in Figure 2. The grid is composed of varying resolutions. The finest resolution for x and y directions are $34 \text{ m}$ and $29 \text{ m}$ respectively, whereas the coarsest resolutions are $4895 \text{ m}$ and $3380 \text{ m}$ respectively [4].
Figure 2: The curvilinear grid used by the Chesapeake Bay Operational Forecast System
Applications of this grid include modeling the bathymetry of the Chesapeake Bay, as well as modeling the attributes of the bay: wind, water level, water current, salinity, and temperature. Figure 3 shows a model of the bathymetry of the Chesapeake Bay from CBOFS.

Figure 3: Bathymetry of the Chesapeake Bay [4]
CBOFS is part of the National Oceanographic and Atmospheric Administration’s (NOAA’s) operational forecast systems, which handle daily weather forecasts, severe storm warnings, and climate monitoring for the United States. The weather data provided by NOAA is updated on an hourly basis and it includes the max/min temperature, probability of precipitation, general weather forecast, dewpoint, sky cover, amount of precipitation, snow amount, ice accumulation, apparent temperature, relative humidity, wind speed and direction, and wind gusts occurring throughout the country [17]. Cutting-edge research and high-tech instrumentation assist scientists with measuring oceanographic and atmospheric conditions and to compile and generate high fidelity images, maps, and charts. The generated data is then provided to the public: citizens, planners, mariners, emergency managers, and other decision makers who need reliable and accurate information [17].

2.3 Google Earth

Google Earth is a cloud-based service that allows users to view the entirety of planet Earth in a lightweight application. Users may look at the globe as seen from outer space, or they may zoom in to see continents, countries, regions, cities, and streets. Google Earth is available for free or for commercial use and is available to anyone with interests in exploring the globe or displaying information using the product. The commercial license must be purchased for any use of the product which is not publicly available [5]. Figure 4 shows the planet and an example of a 3D building as seen using Google Earth.
Google Earth provides advanced graphics rendering capabilities to display geographical data and satellite images to show terrain detail. Satellite images and aerial photography are superimposed onto 3D GIS data which provides Google Earth with the look and feel of a virtual Earth. This application is available on the phone, the web, and on personal computers as a standalone application. The standalone application is shown below in Figure 5.
The standalone application provides users with a large area to navigate and view the 3D model. In addition, there are several options for users to add placemarks, or icons denoting a specific location, lines, images, models, and even tours [5]. A tour is a virtual visit where the camera is panned along a specific path determined by the user. It allows for a virtual walkthrough of specific places, whether located at ground level or as seen from the sky. In addition to adding objects to Google Earth, the standalone application allows users to find places or even map out directions using an integrated version of Google Maps.

The Google Earth API (Application Programming Interface) is a JavaScript based library for embedding Google Earth into any web page [18]. The API provides most of the functionality of the desktop application in the convenience of a web page. It is important to note that Google Earth as viewed on the web does not include the tool bars
provided to those using the standalone application. Figure 4(a) shows an example of Google Earth as seen from a web page. Although the plugin does not provide tool bars similar to those found using the standalone application, developers may still add all of the objects allowed using the standalone application. Developers may draw markers, lines, and shapes; drape images over the terrain; include 3D models; and create tours [18]. These features allow users to build sophisticated 3D applications [18].

Google Earth is widely used by a wide range of industries, such as architecture and engineering, real estate and insurance, energy and utilities, media, government and defense, and nonprofits and education. Figure 6 demonstrates Google Earth's uses in a few of these industries.
In addition to its wide use, Google Earth comes equipped with high level rendering capabilities. By default, Google Earth supports level of detail. This technique allows objects, including the terrain, to increase or decrease in detail depending on the distance the camera is from the object. One example can be seen by just looking at terrain above Old Dominion University. As the camera zooms in closer, it is more apparent that there are roads, houses, rivers, and ports in this area. However, zooming in further reveals the actual university as it looked the day the photo was taken and provided to Google to include in their application. Figure 7 shows Old Dominion University in Google Earth.
Figure 7: A look at Old Dominion University from two different levels of detail: (a) overlooks Norfolk (b) Old Dominion University Campus

As shown in the figures above, Google Earth provides high quality satellite images and 3D models of buildings. Considering the power of Google Earth’s default capabilities and the ability to add objects to the environment to describe real data, Google Earth is an invaluable addition to this research. It would be nearly impossible to generate the aesthetic capabilities and user interface functionalities which would mimic those provided by Google Earth in the duration of this thesis research. Visualizing CBOFS data in a 3D virtual world is made possible because of the visuals and capabilities provided by Google Earth.

2.4 Google Earth’s KML File

KML is simply a file format used to display geographic data in Google Earth, Google Maps, and Google Maps mobile. It is a tag-based structure, based on the XML standards,
which provide nested elements and attributes to contain and describe data. KML files, such as the one in Figure 8, may be simple, displaying several objects, or may be more complex and include thousands of objects, each with different styles. Figure 9 displays the KML file as shown in Google Earth.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple Placemark</name>
    <description>Attached to the ground. Intelligently places itself at the height of the underlying terrain.</description>
    <Point>
      <coordinates>-122.0822035425683,37.4228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Figure 8: Simple KML Example [19]

Figure 9: Example KML objects in Google Earth
There are two methods for adding objects to the Google Earth web plugin: use of the Google Earth API and KML files. The API uses javascript functions to import objects to Google Earth, whereas KML files may be read directly by the Google Earth web or standalone application. Using javascript functions, however, drastically slows the process of importing objects to Google Earth because javascript relies on the web browser and it relies on the conversion between javascript functions and Google Earth implementations. For a small number of objects, added all at once, this may be hardly noticeable; however, when the number of objects being created exceeds approximately one hundred, the speed by which objects are added to Google Earth slows down. However, when using a KML file, the process of importing the objects is handed over to the Google Earth application completely. Without the middle man, the web browser, objects may be added to Google Earth much quicker as Google Earth natively reads and processes these data files.
CHAPTER 3
CBOFS DATA PROCESSING

This chapter begins with the NetCDF data format used by CBOFS, followed by a discussion of the servers hosting the CBOFS data. The chapter then discusses the methods for extracting data from the data files, using a viewer and the Unidata Java NetCDF library. It then describes the preparation of the data files including the directory hierarchy for completed KML files, the graphical user interface for the application created using Java, and methods for handling the data, including memory management. Finally, the chapter discusses how the data are processed and focuses on variables which require detailed explanation: the grid mask, angle, and water current.

3.1 Data Formats and Access

CBOFS exports its outputs into NetCDF format. NetCDF is an acronym for Network Common Data Format and is a self-describing and machine-independent data format specialized for array-oriented scientific data [20]. Unidata, a program at the University Corporation for Atmospheric Research (UCAR), provides tools to access and visualize earth-related data, as well as services which allow researchers and educators to retrieve this data. Unidata is the chief source for software, standards development, and updates pertaining to earth-related data [21]. This thesis uses the Unidata Java library to read NetCDF data files provided from CBOFS. NetCDF was derived from CDF (Common Data Format) which is a library toolkit developed by NASA. The software was built as an interface for the storage and manipulation of multi-dimensional data sets [8]. Newer versions of NetCDF are based on HDF5 (Hierarchical Data Format) which provides more optimized methods for storing and organizing datasets within a file [9]. There are two
methods for obtaining CBOFS data files: THREDDS catalog service or directly from CBOFS servers using OPeNDAP.

3.1.1 OPeNDAP

Open-source Project for a Network Data Access Protocol (OPeNDAP) is a widely accepted and used data transport architecture and protocol. Servers, such as the one hosted for CBOFS, can serve rather large collections of data in HDF, NetCDF, or user-defined formats [22]. These large collections of data files take advantage of the data access protocols defined by OPeNDAP which have become the recommended standard for HTTP requests and responses as recognized by NASA’s Earth Science Data Systems Standards Process Group [23].

3.1.2 THREDDS

Thematic Realtime Environmental Distributed Data Services (THREDDS) is a project with the goal of providing a bridge between data providers and data users in the scientific community [24]. THREDDS is created for students, educators, and researchers to publish, contribute, find, and interact with data relating to the Earth system in a convenient, effective, and integrated fashion [24]. THREDDS provides data in the form of catalogs. These catalogs contain datasets and metadata which describes the datasets within the catalog.

3.2 Raw Data Extraction

The data files collected and referenced in this project directly reference the CBOFS data server for data retrieval. Although THREDDS does provide the CBOFS data using
OPeNDAP [24], the most significant reason for using the CBOFS server directly was a more direct method for data retrieval. THREDDS, while convenient, requires a few extra inquiries concerning the data contained in the dataset whereas the CBOFS server provides the complete file. Once the data files have been collected, the data must be extracted from the file. There are two methods for extracting the data: NetCDF Viewers and the Unidata NetCDF Java Library.

3.2.1 NetCDF Viewers

There are several NetCDF viewers on the web. For instance, Panoply was developed by NASA and generates 2D plots of the data contained in a NetCDF file [25]. Some of these viewers, such as Panoply, may also export specific variables to text, binary, or CSV (Comma Separated Value) files. Figure 10 provides an example of Panoply displaying Surface Wind data.
Array 1, as shown in Figure 11, contains all of the information for the U component of the wind vector. Likewise, array 2 contains all of the V components.
3.2.2 The Unidata NetCDF Java Library

The Unidata NetCDF Java Library is a library which supports the creation and access of array-oriented data [10]. Similar libraries have been created for use with C and Perl; however, Java was selected because it is similar to C# which was used in the original implementation of this research. The NetCDF library is freely available and the latest version, 4.2, was utilized in this research [26].
The library contains five classes which are essential to reading the CBOFS data files: NetcdfFile, DODSNetcdfFile, Variable, Array, and Index [27]. The first, NetcdfFile, is the container which holds the NetCDF file to be read, created, or written to. In order to read in a NetCDF file, it must be opened using the DODSNetcdfFile.open() function which only requires a string containing the filename as its input. This function returns a NetCDF file. Variable holds all of the contents for any variable contained in the data file; this includes its name, description, dimensions, and a pointer to its contents. The Array class is a container which accesses the contents of a variable and returns the contents as an integer, float, double, single, or string. Finally, the Index class is used to access information within an array. The index has an input for every dimension of the array: 2 inputs for 2D arrays, and so on. The use of these classes provides easy access to the contents within the NetCDF file.

The Unidata NetCDF Java library was used over the NetCDF viewers for several reasons. The viewers could only export individual variables to text, binary, or CSV files. As the number of variables needed for the application increased, the number of files to be exported increased. In addition, it took a significant amount of time for the viewer to produce the necessary files and only one file could be created at a time. The viewer did allow a queue to form of files to create, but it was still too inconvenient when expanding the process to more than one data file. Finally, the viewer was not used because the files exported contained too much overhead information requiring specific functions to strip away for each file as all files were not created consistently.

The NetCDF viewer did play an important role, however, in the creation of the file application. It was difficult, when creating KML files, to determine if the exported files
matched the expected results. The viewer provided a method for comparing the results from the exported KML files and the expected results. Unfortunately, the viewer does not provide a customized color scale, but it still provides methods for determining if the KML file's wind vectors were all facing the correct directions and to determine if the magnitudes were as expected.

3.3 Data Preparation

The NetCDF Java library hosts several functionalities which enable the developed application to access and extract parts of the NetCDF data file. Before the file may be accessed, however, it must be archived for retrieval.

3.3.1 Directory Hierarchy

When designing the application, it was important to ensure that an intuitive directory hierarchy was implemented for the files to be generated by the project. Before the directory hierarchy is explained, it is important to understand the composition of the CBOFS data file. The CBOFS data files follow a naming convention. A file name is broken down as follows:
Each of these files contains a list of variables that can be viewed in three ways: by use of a NetCDF viewer, the online THREDDS catalog server, or by displaying a list of variables to the screen using the Unidata NetCDF Java Library. As shown in Figure 12, the online THREDDS catalog server provides a very detailed description of each variable contained in a data file. The web page referenced in Figure 12 is an OPeNDAP Dataset Access Form which allows users to choose which variables to include in the selected dataset and then download the data file in ASCII or Binary. This was another reason why direct access to the OPeNDAP server was used for retrieving data files.
NetCDF files contain two types of variables: scalars and arrays. In this application the scalar variables are ignored as the important information is found within the array variables. The array variables may be single or multidimensional arrays. The example in Figure 12 shows three 4D arrays, ‘u’, ‘v’, and ‘w’, as well as two 3D arrays, ‘Uwind’ and ‘Vwind’. These variables are important to the hierarchy of the directory because they contain a very important dimension, time. Each data file is a container holding variable
information that spans a 7 hour period. For any given day, 4 data files are generated that contain 6 new hour periods and 1 overlapping hour. The final hour contained in the data files is a duplicate of the first hour in the following data file. Now that the hierarchy of the data file has been explained, the directory hierarchy will be discussed.

Figure 13: Directory Hierarchy Diagram
The directory hierarchy, as shown in Figure 13, is fairly simple and intuitive. Currently, the application requires two directories to be present:

Cbofs Datasets/NetCDFs/

Cbofs Datasets/KMLs/.

The first directory holds all of the CBOFS data files, and the latter contains the generated KML files.

When the application begins exporting a data file, a directory is created within the KML folder if it doesn’t already exist. This directory is named after the year of the data file being processed followed by a directory named after the appropriate month and finally, a directory named after the appropriate day. Within this directory will be a list of 24 folders. As the count suggests, each folder is based off a one hour period in the day, starting with ‘00’ and ending with ‘23’. The generated KML files for the appropriate day and time step are stored in each of these folders. Finally, a folder is created for Water Current, Salinity, and Temperature as there are 20 layers for each of these data types, and each depth is stored in its own KML file. Each time step folder can contain 102 individual KML files.

The KML files generated by the application are split into two categories: single layer and multi-layer. The single layer KML files are Wind and Water Level and exist only on the water’s surface. The multilayer KML files are Salinity, Salinity Mesh, Temperature, Temperature Mesh, and Water Current which have 20 depth layers each. Salinity and Temperature are “special” in the sense that they can be visualized by two different methods. This will be further discussed in the visualization chapter.
3.3.2 Application Graphical User Interface

This application evolved through several renditions of programs. The first implementation of a KML exporter required use of a viewer. The viewer allowed selected variables to be exported as text files. Then a program developed in C++ would parse the text files and prepare them for a C# program which read the cleaned text files and export KMLs. This method worked well for proving that these data files could be converted to KMLs but otherwise required too many intermediate steps. The next rendition worked similarly to the first except the C# program was improved to handle extraction of the relevant data from the header information present in the text file output by the viewer. Finally, the entire project was ported from C# to Java to take advantage of Unidata’s NetCDF Java Library. This enhancement would completely phase out the need for a viewer and allow direct access to the CBOFS data files.

Figure 14: The CBOFS KML Exporter GUI
The current application interacts with the user through a custom graphical user interface (GUI) as shown in Figure 14. There are 3 features to note concerning this GUI. The first is the “Available Data Files” panel and the “Export” button. Here, the application lists all CBOFS data files contained in the directory CBOFS Datasets/NetCDFs/. From this list, the user may select one or more data sets they wish to export. It is also worth noting that the data sets selected by the user do not have to be adjacent to one another; for example, the user may choose to select every other data file. The “Export” button allows the user to export the selected files. When this button is selected, it is then disabled to prevent the user from initializing another export thread.

The next feature is the “Options” drop down menu as well as the “Overwrite Files” and “Output XML” checkbox options. First, the Options menu provides the user with a list of KML file types that they export. By default, all file types are selected. Figure 15 shows the GUI with the drop down menu displaying the available options.
Figure 15: CBOFS KML Exporter Application with Options Shown

Additionally, there are two checkbox options located below the “Available Data Files” panel. The first is the “Overwrite Files” option, and this allows the user to overwrite previously generated data files. By default, this option is deselected but may be enabled in the event that the user suspects that a data file was corrupt, or in the event that the user exits the program without allowing the program to finish writing the KML files. The final option is “Output XML”. This option allows the user to update or create the XML file responsible for updating the website of available KML files. More information regarding the website and this option will be discussed later.

The last feature includes two progress bars on the bottom of the GUI and the large panel on the right side of the window. The topmost progress bar allows the user to see that the application is working by showing a blue bar which bounces from side to side. If the application has stalled, or when the application has finished generating files this
progress bar will remove the blue bar from view. The second progress bar provides the user with a percentage of progress made while exporting the current file. For each CBOFS file to be exported, this progress bar will start a 0% and reach 100% when all KML files have been generated and the application releases the CBOFS data file. Finally, the right panel is an output log of events that occurred while exporting the data file. The application updates the output log for the following actions: initialization, reading variables, performing calculations, writing files, skipping existing files, and completing a data file.

3.3.3 Data Handling

In previous renditions of the CBOFS KML Exporter, all relevant data was first read and stored in local variables, followed by handling calculations, and, finally, exported all KML files sequentially. In C#, this method worked fine because a text file could be read into a string one line at a time and parsed to find the relevant data which would then be stored in a local variable. In the Java implementation, this method was no longer reasonable because data from the data file could no longer be read one line at a time. The Unidata NetCDF Library instead reads the entire array, or sections of the array into a local variable all at once. This method of retrieving information requires more memory to be used when handling the data, especially when higher dimension arrays are involved.

To reduce the amount of memory needed by the application, variables are read one at a time and stored in local variables which require less memory. The default data type for most multidimensional arrays is double. Variables such as angle, latitude, and longitude are then converted from double to float, where the higher precision isn’t necessary. Some variables, such as Uwind, Vwind, u, and v, should not be converted to a lower precision.
Before discussing how the Uwind, Vwind, u, and v variables are handled as to save memory, it is important to clarify the use of "U component" and "V component" which will be used to describe the variables below. The U component refers to the Uwind and u variables, and the V component refers to the Vwind and v variables.

Figure 16: Vector Components

Figure 16 provides a graphical representation of the U and V components of a vector. Suppose vector \( a \) describes surface wind. The magnitude describes the length of the vector which can be found using two components, U and V. U is the projection of vector \( a \) on the x axis and V the projection of vector \( a \) on the y axis. By using the simple equation \( ||a|| = \sqrt{U^2 + V^2} \), the magnitude of vector \( a \) may be determined. This method is used to determine the surface wind and water current vectors from the U and V wind and water current components respectively.

The U component is stored in the locally reserved magnitude variable which is double precision. The V component is then read and stored into the local angle variable. Immediately after the U and V components have been read, calculations to determine the magnitude and angle are processed. The calculated magnitude overwrites the previous U
component information, recycling it. The Array variable is then re-declared and used to retrieve another variable from the data file, recycling the V component. These four variables, Uwind, Vwind, u, and v are the only variables which must be read together before calculations can be made. They also happen to be the most expensive variables in terms of memory cost which is why they must be handled in such a way as to reduce overall memory use.

3.4 Data Processing

The data processing phase of the application focuses on three areas: applying the mask, angle calculations, and water current’s U and V component rectification. While it is important to handle each variable properly, those mentioned require unique handling due to certain qualifications each possesses. The next few subsections provide a detailed explanation of how and why these variables are handled uniquely.

3.4.1 Grid Mask

ROMS is based on a rectangular, curvilinear grid [6]. Some areas with this grid are land-based and most are water-based whether rivers or large bodies of water. To denote which areas are land and which are water, a mask was created. This mask may be directly applied to the CBOFS curvilinear grid.

The mask is a variable contained in the data file named ‘mask_rho’ which contains either a 0 or a 1 for a data point within the normal grid. The “normal” grid is considered to be the curvilinear grid as shown in Figure 2. If the mask is a 0, it means that the current data point is over land; therefore, a 1 is a data point over water. This also shows that the actual grid utilized is not exactly 291x332 data points; instead, the maximum number of
rows needed to define the curvilinear grid is 291, and the maximum number of columns needed is 332. After the mask is applied to the data, approximately 1/5 of the original grid is utilized to describe the utilized grid. For this reason, it is possible to display all of the data points for any data type at one time in Google Earth. Without the mask, there exist 96,612 data points, and with the mask only 19,772 data points are needed to display the curvilinear grid.

3.4.2 Angle of Direction

The second phase involves determining the angles of the wind and water current vectors. There are two angles which must be found in order to determine the final vector direction: the reference angle and the vector’s angle. Generally, the reference angle is zero as angles start on the positive x-axis and rotate counterclockwise based on the angle.
The CBOFS data, however, has a varying reference angle due to its curvilinear grid as shown in Figure 17. This angle may be between $-90^0$ and $90^0$ where $0^0$ is east.

Fortunately, the reference angle is provided in the CBOFS data file. The vector angle must then be computed by the U and V components using the following equation:

$$\theta = \left(\frac{180}{\pi}\right) \* \cos^{-1}\left(\frac{|U|}{\sqrt{U^2 + V^2}}\right).$$

$\theta$ is the angle, in degrees, of the vector in the first quadrant.

Figure 18 shows the relationship between angle $\theta$, U, and V. The graph on the left shows $\theta$ calculated as described earlier in the first quadrant. The blue line denotes the vector of U and V. Because we take the absolute value of the U component, the angle
will always result in the first quadrant. In order to determine the true quadrant and therefore the true angle of the vector, it must be known if $U$ and $V$ are positive or negative. Figure 18 shows the relationship between the four possibilities of $U$ and $V$ and their correspondence to the quadrants.

![Figure 18: Quadrants and Corresponding Angles](image)

**Figure 18: Quadrants and Corresponding Angles**

Figure 19 shows how an object is oriented in Google Earth. Using this graph, the quadrant of the vector can be determined. Once the appropriate quadrant is determined, the corresponding angle may be obtained. It is then added to the corresponding reference angle, and the result is the direction of the vector in Google Earth.

**3.4.3 Water Current**

Water current is derived from the variables ‘$u$', ‘$v$', and ‘$w$' in the CBOFS data file. Similar to ‘$Uwind$' and ‘$Vwind$', these components describe a vector for the water
current. It is important to note that the ‘w’ variable, which is the component that
describes the magnitude and direction of current in the z-direction, is unnecessary
because the ‘w’ variable is very small. Therefore, it can be concluded that the variation in
depth is minimal, and the vector may be approximated in two dimensions.

Although the reduction in vector components simplified the calculations to determine
the magnitude and direction, a problem still persisted. The two remaining components,
‘u’ and ‘v’ have different array sizes as shown in Table 2. The difference in array sizes is
the result of an offset present in the ‘u’ and ‘v’ components. The ‘u’ component has been
offset by half of a column grid cell, and the ‘v’ component has been offset by half of a
row grid cell as shown in Figure 20.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>‘u’</th>
<th>‘v’</th>
</tr>
</thead>
<tbody>
<tr>
<td>ocean_time</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>s (Depth)</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>eta (Row)</td>
<td>291</td>
<td>290</td>
</tr>
<tr>
<td>xi (Column)</td>
<td>331</td>
<td>332</td>
</tr>
</tbody>
</table>
In order to apply the mask to the ‘u’ and ‘v’ components, the offset must be corrected and the missing grid cell must be approximated. The following approximation method was applied to ‘u’ and ‘v’ respectively:

\[
u[r][c] = 0.5(u[r][c] + u[r][c + 1])
\]

\[
v[r][c] = 0.5(v[r][c] + v[r + 1][c]),
\]

where \(r\) denotes the current row index and \(c\) denotes the current column index [30]. Once the normal grid has been approximated, the ‘u’ and ‘v’ components may be handled in the same manner as the ‘Uwind’ and ‘Vwind’ components.

After all the above computations are performed, the CBOFS data is ready to be visualized. Chapter 4 will discuss the visualization, i.e., generation of graphical representations, of CBOFS data in detail.
CHAPTER 4
VISUALIZATION

Visualization, in the sense of computer graphics, may be described as the technique for creating images, diagrams, or animations to communicate a message [19]. In general, a message may be anything from directions to a place, emotion, processes, ideas, and more. In this research, the message being conveyed is scientific data. This chapter first describes the current visualization methods used by the NOAA CBOFS website. It then discusses the visualization methods in Google Earth, and, finally, presents this research’s approach to visualize CBOFS data in Google Earth.

4.1 Current CBOFS Data Visualization

Currently, CBOFS displays the data obtained from measurements and mathematical approximations as 2D images. These images are generated for both Nowcast and Forecast Guidance in each of the 5 attributes. Each attribute displays a specific number of time steps depending on whether Nowcast or Forecast Guidance was the selected category. Each time step represents a 1 hour period. If Nowcast is selected, there are 31 time steps available; if Forecast Guidance is selected, 49 time steps are available. Each time step, independent of the category or attribute selected, is added to an animation queue where each image is displayed for 1 second before switching to the next image. The user is then allowed the option to select any time step from a list of available times as well as stop the animation cycle and progress through the individual time steps at their leisure. Figures 21 through 25 show examples of Nowcast images visualized by CBOFS for each attribute. There is no significant difference between Nowcast and Forecast Guidance as far as data
visualization is concerned; therefore, examples of Forecast Guidance will be omitted.

Figure 21: Water Levels Nowcast [31]
Figure 22: Surface Winds Nowcast [32]
Figure 23: Field Surface Water Temperature Nowcast [33]
Figure 24: Field Surface Water Salinity Nowcast [34]
Figure 25: Field Surface Water Currents Nowcast [35]
4.2 Visualization in Google Earth

Google Earth provides a wide variety of objects which may be added to the virtual planet. Users may add points, lines, polygons, ground and screen overlays, and even 3D models to Google Earth. Determining which object to add is crucial to properly displaying and conveying the purpose of the data. This section will discuss the different objects which may be added to Google Earth and the different styles that may be used to customize them.

4.2.1 Placemarks

In Google Earth, any object which has geometry is contained within a placemark. A placemark may have a name, an address, a phone number, a description, a style, a region, or geometry, and be visible to the viewer [29]. It does not have to include all of these attributes, but it must have at least one of these attributes to exist. In the generated KML files, each placemark only contains a description, a style, and geometry. It is important to note that all placemarks are nameless; when a placemark is visible and has a name, that name is displayed about the placemark. When there is a large number of placemarks in a small area, each with a name, it can cause the screen to become cluttered. A menu option does exist which hides the names of placemarks on the screen; however, this menu option is only available in the standalone application, and these KML files will be viewed primarily through the web plugin; therefore, names have been omitted. Figure 26 provides an example of how cluttered the screen becomes when placemarks have names.
As mentioned previously, each placemark may contain a geometry element. A geometry element is defined as something which is visible to the user while viewing the virtual globe; this includes points, lines, polygons, and models. A point in Google Earth, by default, is a yellow pin on the globe [29]. Figure 26 provides an example of the yellow pin icon, though it may be difficult to see. A point may also have an orientation; however, depending on the icon, it may not be obvious what the orientation is. This is the simplest object Google Earth has to offer as it is a 2D image which exists at a designated location. All locations in Google Earth rely on the geographic coordinate system: latitude, longitude, and altitude. Latitude and longitude are defined by decimal coordinates in the North and East directions respectively. Altitude is measured in meters.
and may be referenced to sea level, the bathymetry of the ocean or any other body of water, or to ground level.

Lines in Google Earth may be drawn in two different ways depending on the user’s needs. The first method for rendering a line is by using Linestring geometry. A Linestring is created by declaring two locations, and Google Earth renders a line between them. The second method for rendering a line is by using the LinearRing geometry which creates a closed loop. A LinearRing geometry may have as few as three locations specified, creating a triangle, and as many additional locations as deemed necessary [29].

Polygons are defined using LinearRing geometry but are rendered solid. The polygon may be defined by several LinearRing geometries by defining the outerBoundaryIs and innerBoundaryIs attribute. The inner boundary defines one or several cutouts within the polygon [29].

A model is a geometry which cannot be explicitly created inside Google Earth. It must be referenced by a URL where Google Earth will request a copy of the model and then load the model at the designated location. There are some limitations to models in Google Earth. First, the model must be a COLLADA file which has the “.dae” extension. Google Earth only recognizes triangles and lines as primitives, and the maximum number of triangles allowed for any particular model is 21,845 [29]. Animation and skinning is not supported, and external geometry references are also not supported [29]. So long as the model abides by these standards, Google Earth will create the model in its own coordinate space where it will then be located, positioned, and scaled [29].
Figure 27: Orientation of a Model in Google Earth [29]

Figure 27 shows how a model may be oriented in Google Earth. The orientation parameter includes heading, tilt, and roll. These parameters affect the model by rotating the model about its local axes. The heading is the facing direction which may be any angle, 0 to 360 degrees, off the North direction (where North is 0 degrees) and is the rotation about the z direction (where z points up). The tilt describes a rotation of the model about the x axis, and roll describes a rotation about the y axis. These axes reference the appropriate local axis of the model.

4.2.2 Geometry Styles

In addition to geometry, placemarks may contain style information. A style affects how geometry is presented in Google Earth. All geometry, with the exception of a model, may be redefined using a style. A point can be styled by the following attributes:
• Color
  o Defined by a hexadecimal string (AARRGGBB)

• Scale
  o Defined by a floating point number

• Orientation
  o Defined in degrees (refer to Figure 16 for further details)

• Custom Icon
• Defined by a URL pointing to an imageHotspot
  o Specifies the position within the icon which is “anchored” to the
    placemarks location, e.g. (0.5, 0.5) designates the center of the icon as the
    point referencing the placemarks location

A Linestring may be styled using the following attributes:

• Color
  o Defined by a hexadecimal string (AARRGGBB)

• Width
  o Defined by a floating point number

A Polygon may be styled using the following attributes:

• Color
  o Defined by a hexadecimal string (AARRGGBB)

• Fill
  o Boolean value

• Outline
  o Boolean value
The outline of a polygon may be styled differently than the polygon itself. The outline is styled like a Linestring [29].

4.2.3 KML Files for Google Earth

Figure 28 above provides a small excerpt of a KML file similar to the ones generated in this thesis. The KML file provides instructions for Google Earth which assists in the creation of objects. These objects, linestrings, points, polygons, etc. are described by the <placemark> tag which is crucial in the generation of these KML files. Without this particular tag, objects would not be created, and the data would remain invisible. This particular example provides Google Earth with a single object, a water current vector. The magnitude is provided as 2.41 knots and has been colored black, and the angle of this vector is 93.76 degrees: therefore, the vector will be pointing approximately due East.

The files exported by the CBOFS KML Exporter are much larger, containing approximately 20,000 <placemark> declarations which includes all of the description and styling data as well as the individual placemarks location.
Figure 28: Example KML file
4.3 CBOFS Data Representation in Google Earth

It is important to ensure that the objects in an image complement the message being conveyed. In this research, the "message" is the CBOFS data, and each type of data must be handled differently in order to convey the message correctly. This section will discuss the approaches to determine what objects in Google Earth best suited the five data types provided by CBOFS, starting with Surface Winds and Water Current, followed by Salinity, Temperature, and Water Level.

4.3.1 Surface Wind and Water Current

Velocity is always represented by a vector because it has magnitude and direction. This vector may be represented in two or three dimensions. Surface Wind and Water Current velocity data is no different; however, it exists to only two dimensions. The third dimension for Surface Wind velocity, height, is irrelevant in this context because this data is assumed to be parallel to the water's surface. Water Current is also assumed to be parallel to the water's surface because the third component, 'w', is several magnitudes smaller than the 'u' and 'v' components.

The most common technique for representing a vector is to use an arrow; this technique is used in this research. Currently, a small arrow replaces the default icon provided by Google Earth. Its length is evenly distributed in the image, and a hotspot is declared in the exact center of the arrow. This allows a very even spin when the arrow is oriented in different directions. Also, the arrow may be colored which would indicate the magnitude of the arrow. Additionally, each individual arrow may be selected from the main window in Google Earth. The selected arrow will display a balloon window which
contains information concerning the arrows representing Surface Wind velocity (Knots) or the Water Current velocity (Knots) and the corresponding angle (degrees).

The arrow has 21x9 pixels. In Google Earth, the size of the icon makes a big difference in the viewing range of the data. An icon, when present on the screen, occupies as much space as the original size of the icon. Larger icons occupy more space than smaller icons. As the user zooms in and out of the screen, the icons appear to change size in relation to the view of Google Earth; however, the icons are just shifting slightly on the screen to compensate for the camera's current location (where the camera refers to the user) as icons are always located directly above the coordinate they have been assigned. As the user zooms in, however, the icons appear to shrink dramatically, almost to the point where the icons cannot be seen which poses a problem when users wish to zoom in very close to the water's surface. Additionally, the icons can become quite crowded when the users zooms extremely far from the Chesapeake Bay. Small arrows were used, however, because it has been assumed that the user will want to stay zoomed out to see a majority of data points. So long as this assumption holds true, the negative effects of using small arrows is irrelevant. However, there are methods which would allow the size of the arrow to increase in size which is one of the future additions which will be discussed in the final chapter of this document.

4.3.2 Salinity, Temperature, and Water Level

Salinity, Temperature, and Water Level are all scalars. This property allows them to be visualized in a multitude of different methods. In this research, there are two methods for visualizing scalars; the first method uses a point and a color to represent the scalar at a designated location, and the second method displays the entire data set as a mesh where
color and height differences display the relationship between different data points.

The point method for representing scalars is used to represent Salinity and Temperature. Water Level could also be represented using this method; however, it is only represented by the mesh method. In this method, the Google Earth object “point” is used, and a small circular icon replaces the default yellow pin. Its hotspot is set to the center of the circle such that it is perceived as hovering over its designated position. Finally, the icon is colored based on its magnitude. Similar to the arrow, the circle is a small icon such that the user may zoomed out and still distinguish individual data points.

The mesh method is used to represent Salinity, Temperature, and Water Level. This method takes advantage of Google Earth’s polygon geometry such that a basic mesh can be generated to display the data. A mesh is constructed by creating faces between neighboring vertices and is generally created using the most basic primitive, triangles. Figure 29 displays an example wireframe mesh which was generated using XNA Game Studio and Visual Studio 2010.
This same technique can be used in Google Earth; however, the basic primitive must be a square instead of a triangle. In the provided CBOFS data, there are approximately 19,772 data points to represent. Each of these data points represents a vertex for the mesh. If a triangle mesh were to be used, approximately 35,000 triangles would be needed to represent the mesh. However, this number can be reduced by half if square primitives are used instead, requiring only 17,500 squares.

To complement the smooth transitions of the mesh geometry, color smoothing was implemented for all mesh models. Unfortunately, Google Earth does not provide support for color interpolation. In the point method for displaying data, each data point is compared against a color scale to determine which color the icon will be. This is the same method used by CBOFS to represent their data in the 2D image format, and it works well
for data which are not connected to one another. However, because connecting polygons were used, it was appropriate to provide a transition in color as well as height differences.

Color smoothing is the result of attempting to create a transition between triangles in a mesh. However, before color smoothing is described, it is important to know how the color of a polygon is selected. Suppose we are looking at 3 cells of a mesh describing temperature and they have values as shown in Figure 30.

![Figure 30: Color smoothing example part 1](image)

Currently, the cells are blank because a color has not yet been selected. In order to determine the appropriate color, a weighted average of the temperatures at each corner must be obtained and compared against a color scale. Figure 31 provides the colors selected for each grid and the appropriate average temperature for each cell. The relevant part of the scale has been included for comparison.
If the point method were chosen to represent the data, these would be the final colors for the grid; however, as this is a mesh, color smoothing was implemented. From the scale in Figure 31, it is apparent that a temperature of 79.2°F and 79.625°F receive the same color; however, 79.625°F is much closer to the color representing the next interval than it is to the color representing the color of the previous interval. To compensate for this difference, weighted color blending between the selected color and the next closest interval is calculated, and the resulting color is returned. To illustrate the end result of this process, Figure 32 shows the same grid cells with color smoothing.
4.3.3 Models

Google Earth also provides support for displaying COLLADA models, though it was not used in this research. They require a large amount of memory and very few may be rendered before Google Earth becomes unresponsive. The responsiveness of Google Earth was a very important aspect in determining what type of object to use when displaying data as the use of models was the first choice. To counteract the responsiveness problem, more simple geometric objects were sought after. Thus, points and polygons were selected to display the CBOFS data. Using these more simple geometric objects, all of the CBOFS data may be viewed simultaneously whereas models would have to be hidden when they were too far from the camera.

The use of models sparked interest in generating COLLADA models based on Water Level data; however, the data files proved to be too large. The model also had to be split into 18 different models in order for Google Earth to properly display the entire set of data. Finally, COLLADA model generation required an intermediate step which could not be automated. This step involved use of a custom importer in Google Sketchup which would create the mesh geometry based on vertex and color information. The models, however, were aesthetically pleasing once the end result was loaded into Google Earth. The application was also rather responsive, but it required several minutes to load depending on the capabilities of the computer.
CHAPTER 5
IMPLEMENTATION, RESULTS, AND DISCUSSIONS

This chapter describes the implementation of the system, demonstrates the final results, and discusses existing problems.

\[\text{Figure 33: Implementation Overview}\]

5.1 Implementation

The flow diagram shown in Figure 33 provides an overview of the project implementation. CBOFS data may be visualized on a web site using the Google Earth Web plugin as a result of using the Unidata NetCDF Java Library to extract the CBOFS data and export them as KML files, native to Google Earth.

The data, contained in a single data file, is read into the CBOFS KML Exporter application and processed such that individual components of the data file may be extracted, when needed, in order to facilitate the composition of each KML file. As each data type is extracted from the data file, it is primed and stored so that exporting the
information may be a smooth process. Priming the data involves determining vector magnitude and direction or simply converting units so that the end result resembles the data hosted by CBOFS. Finally, the data is exported to individual KML files which are then ready to be loaded into the Google Earth standalone or Web plugin to be viewed.

When the files are viewed using the Web plugin, they must be appropriately archived on a server which will host the files. They may then be downloaded and imported to Google Earth via a website also created in this research. Viewers are then encouraged to explore the different aspects of each file, zoom in to view individual data points, or zoom out to look over the large picture which is created by all of the data for the Chesapeake Bay area. The purpose of this website is to explore the CBOFS data in a 3D environment and to expand that which has been already accomplished by CBOFS.

5.2 Results

This section is dedicated to providing screen captures of the results of both the CBOFS KML Exporter and the web page created to allow users to explore the CBOFS data. First, the results of the web page will be displayed as well as a description of the web page. What follows are the results of the CBOFS KML Exporter as shown in Google Earth.
The web page as shown in Figure 34 will allow users to explore the data provided by CBOFS. To the right of the Google Earth window resides several drop menus which allow the user to select a specific day, time, data file, and depth to display in the Google Earth window. Below these two windows are two sections; the left provides a brief
description of CBOFS as well as providing helpful links to the main CBOFS webpage. The right section provides information about the authors.

The following figures have been created by the CBOFS KML Exporter application and visualized using the Google Earth standalone application. Figure 35 visualizes Water Level and may be compared to Figure 21. Figure 36 visualizes Surface Wind velocity and may be compared to Figure 22. Figures 37 and 38 visualize Temperature and may be compared to Figure 23. Figures 39 and 40 visualize Salinity and may be compared to Figure 24. Finally, Figure 41 visualizes Water Current and may be compared to Figure 25. Further screen captures for Salinity, Temperature, and Water Current may be found in Appendices A – C respectively. Each appendix provides all 20 depth variations for the data visualized below and in Figures 21-25. This information was included to show the completeness of the application and to show how depth affects each data type respectively.

The screen captures provided in Figures 35-41 display the CBOFS data in the same nature as that provided on the CBOFS website. These screen captures are expected to be used for comparison against those on the CBOFS website and to provide an overall view of the Chesapeake Bay area; however, these screen captures do not show the true purpose of visualizing CBOFS data in Google Earth. Figures 42-44 provide more specific screen captures which will provide more understanding of why Google Earth is the best option for visualizing CBOFS data.
Figure 35: Water Level
Figure 36: Surface Wind Velocity
Figure 38: Temperature Mesh
Figure 39: Salinity
Figure 40: Salinity Mesh
Figure 41: Water Current
Figure 42: Exploring Water Current

Water Current Velocity: 0.78 Knots
Angle: 270.01 Degrees
5.3 Discussions

This research reproduces measurements taken by CBOFS and visualizes them in Google Earth. The data provided in the KML files generated by this research is no more accurate than what is provided in the data files on the OPeNDAP servers. As such, the results of this research are meant to provide users the ability to explore and learn more about the Chesapeake Bay and its climates.
Although this research is complete, there are still some aspects of the visualization techniques used which could be improved. For instance, a combination of placemarks and mesh modeling would provide users with sufficient resources at the same time. This proposed technique would provide the user with a mesh whenever the camera is zoomed out further than 100 km from the Earth’s surface. As the camera zooms in further, the mesh model would be replaced with the point model where icons would display the results. This would allow the user a smooth overview of the entire Chesapeake Bay and the ability to select specific data points while zoomed in closer to the data.

Additionally, the color smoothing algorithm could be improved to provide a better estimated color for a more smooth transition between grid cells. For all models, a sense of level of detail could also be implemented. While viewing the entire Chesapeake Bay, there is not a need to see every single data point as many overlap, but as the user zooms in further more data points could come into view as room is available on the screen. Finally, when the user zooms in close enough to only see very few data points on the screen, interpolation between data points could be implemented such that wide spaces would be filled in with approximated data to provide a finer grid.

Throughout this research, errors have been found with reproducing the CBOFS data visualization. The first error is that it seems the scale by which the CBOFS data is compared changes periodically. The method by which the scale is determined has not yet been determined and has made reproducing CBOFS data visualization on a large scale difficult. The scales for surface wind and water currents do not change; however, temperature, salinity, and water level have been found to change, sometimes drastically, which provides errors in the coloration of the reproduced data in Google Earth.
CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This thesis proposed a framework for visualization of the Chesapeake Bay Operational Forecast System in Google Earth. The developed application framework consists of importing the CBOFS data file, data processing, and exporting the KML files. Finally, these files may be further imported into a Google Earth standalone application or the Google Earth Web plugin, such as the one hosted by ODU.

The results produced by the research in this thesis demonstrate the usefulness and potential for the proposed method; however, with more research and more involved KML files, better visualization techniques can be acquired to further separate the 3D CBOFS data from its 2D counterpart on the CBOFS website. Furthermore, if further interpolation of the CBOFS data was implemented, a much finer grid would be generated and provide more specific information about the Chesapeake Bay. The additional information would require use of visualization optimization, such as level of detail where the finer grid would be shown only when the viewer zoomed in to see the data. Finally, contour mapping could be implemented and added to the KML files. These contour maps would provide additional information, such as showing patterns in temperature and salinity changes, as well as distinguishing the differences in water level for any given time step. These additions would provide the viewer with many tools to analyze the CBOFS data, and considering the data would be available to anyone with Internet access, it could provide users with useful information on-the-fly in an easy to see and beneficial format.
REFERENCES


APPENDIX A: Salinity Mesh

Figure 45: Salinity Mesh Depth -0.025

Figure 46: Salinity Mesh Depth -0.075

Figure 46: Salinity Mesh Depth -0.125

Figure 47: Salinity Mesh Depth -0.175
Figure 48: Salinity Mesh Depth -0.225

Figure 49: Salinity Mesh Depth -0.275

Figure 50: Salinity Mesh Depth -0.325

Figure 51: Salinity Mesh Depth -0.375
Figure 52: Salinity Mesh Depth -0.425

Figure 53: Salinity Mesh Depth -0.475

Figure 54: Salinity Mesh Depth -0.525

Figure 55: Salinity Mesh Depth -0.575
APPENDIX B: Temperature Mesh

Figure 63: Temperature Mesh Depth -0.025
Figure 64: Temperature Mesh Depth -0.075

Figure 65: Temperature Mesh Depth -0.125
Figure 66: Temperature Mesh Depth -0.175
Figure 67: Temperature Mesh Depth -0.225
Figure 68: Temperature Mesh Depth -0.275
Figure 69: Temperature Mesh Depth -0.325
Figure 70: Temperature Mesh Depth -0.375
Figure 71: Temperature Mesh Depth -0.425
Figure 72: Temperature Mesh Depth -0.475
Figure 73: Temperature Mesh Depth -0.525
Figure 74: Temperature Mesh Depth -0.575
Figure 75: Temperature Mesh Depth -0.625  
Figure 76: Temperature Mesh Depth -0.675  
Figure 77: Temperature Mesh Depth -0.725  
Figure 78: Temperature Mesh Depth -0.775
Figure 79: Temperature Mesh Depth -0.825
Figure 80: Temperature Mesh Depth -0.875
Figure 81: Temperature Mesh Depth -0.925
Figure 82: Temperature Mesh Depth -0.975
APPENDIX C: Water Current

Figure 83: Water Current Depth -0.025

Figure 84: Water Current Depth -0.075

Figure 85: Water Current Depth -0.125

Figure 86: Water Current Depth -0.175
Figure 95: Water Current Depth -0.625

Figure 96: Water Current Depth -0.675

Figure 97: Water Current Depth -0.725

Figure 98: Water Current Depth -0.775
Figure 99: Water Current Depth -0.825
Figure 100: Water Current Depth -0.875
Figure 101: Water Current Depth -0.925
Figure 102: Water Current Depth -0.975
VITA

Gary Lawson

Educational Background

- M.S. in Modeling and Simulation (Dec. 2011), Department of Modeling, Simulation, and Visualization Engineering, Old Dominion University, Norfolk, VA

- B.S. in Electrical Engineering (May. 2010), Electrical and Computer Engineering Department, Old Dominion University, Norfolk, VA

Experience

- Research / Teaching Assistant
  Old Dominion University
  Norfolk, VA
  August 2010 – Present

- Co-op Electrical Engineer
  Harris Corporation
  Chesapeake, VA
  July 2008 – January 2010

  - Testing: Bit Error, Jitter, Power, New Hardware/Firmware
  - Documentation of Test Results/Procedural Documents
  - Initialize New Hardware with Updated Firmware / Rail Testing
  - Small Design / Analysis of Schematics

Projects

- Google Earth-based Framework for Wind Farm Modeling and Simulation
  A wind farm model which takes advantage of Google Earth’s unique visualization and interaction capabilities. The wind farms are located off the East Coast of the U.S., providing users tools for adding additional wind turbines as well as measuring theoretical measurements for energy, efficiency, and weather conditions affecting individual wind turbines.

- Virtual Tour of Vmasc
  Allows users the opportunity to explore the Virginia Modeling Simulation Analysis and Simulation Center located in Suffolk, VA. It may be downloaded from the web or played directly from the web browser on any personal computer, providing a first-person experience to freely walk through the building.