AR-008-350

DSTO-TR-0048

DSTO

FFG-7 Class Frigate Airwake Viewer

C. A. Heinze and A. M. Arney

DEPARTMENT OF DEFENCE
DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

FFG-7 Class Frigate Airwake Viewer

C. A. Heinze and A. M. Arney

Aeronautical and Maritime Research Laboratory

ABSTRACT

Technical Report

This document presents the operation of the graphical display program *ffgView*, which has been written to run on the Silicon Graphics family of computers. The program allows the airwake around an 'Adelaide' class FFG-7 frigate to be displayed using data generated from wind tunnel testing or from actual ship trials. Details of the required file formats and of the structure of the source code are included here, as are detailed operating instructions.

APPROVED FOR PUBLIC RELEASE

DSTO-TR-0048

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Aeronautical and Maritime Research Laboratory
GPO Box 4331
Melbourne Victoria 3001 Australia

Telephone: (03) 626 7000
Fax: (03) 626 7999
© Commonwealth of Australia 1994
AR No. 008-350
AUGUST 1994

FFG-7 Class Frigate Airwake Viewer

EXECUTIVE SUMMARY

This report details the graphical display program *ffgView*, which allows the user to view in three dimensions a vector display of airflow around an 'Adelaide' class FFG-7 frigate. This package can readily be adapted to display airflow around any object by replacing the frigate geometry file with a file representing the required object. Wind vectors are displayed in colour to enable easy recognition of velocities, and the view can be manipulated using a mouse to provide the user with any desired viewing angle. The program is coded in C on a Silicon Graphics (SG) computer.

The graphical display package was designed to enable the display of airwake data collected during trials on a Royal Australian Navy 'Adelaide' class frigate. The 'Adelaide' class is based on the US FFG-7 class frigate and for the purposes of airflow over and around the flight deck the two classes are identical. With the availability of wind tunnel data for the frigate, it was considered desirable to be able to display this and perhaps other types of data that may become available in the future. For this reason, a general data file format for input into the graphical display program was chosen. Conversion programs were written to deal with the data format used for the full scale data as well as that used for the wind tunnel results.

The display package *ffgView* has the following capabilities:

- Display in colour an orthogonal or perspective view of the flight deck of an FFG-7.
- Display the airwake patterns around the frigate using coloured vectors.
- Allow the input of data obtained from any source conforming to the specified input file format.
- Provide utilities allowing the conversion of raw data to the required input file format.
- Allow three-dimensional rotation, translation, and scaling of the view.
- Provide a means of saving and printing screen images

The limitations of this package are as follows:

- It can only be run on machines supporting SG Graphics Library (GL) utilities, i.e. SG machines.
- Screen images can only be saved in Graphics Interchange Format (GIF) or SG Red-Green-Blue (RGB) format. An image saved in GIF format is easily transportable between machines (including Macintosh and IBM compatible personal computers).

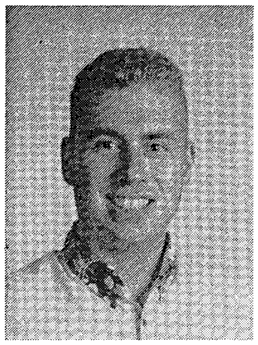
Other features of importance to the functioning of *ffgView* are explained throughout this report. To facilitate easier modification in the future, details of the structure of the code are given.

The frigate airwake viewer is successfully running on an SG workstation. It has already proved useful in the early analysis of available data and will continue to be an important tool for a variety of analysis tasks within Air Operations Division.

With modification the program could be used for a wider variety of tasks than the specialised frigate airwake analysis for which it is currently being used.

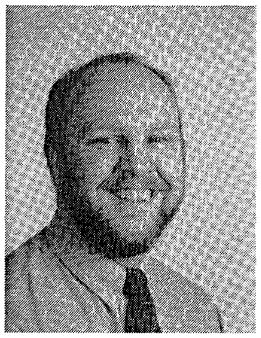
Authors

C.A. Heinze
Air Operations Division



Clinton A. Heinze graduated with first class honours from Aerospace Engineering at the Royal Melbourne Institute of Technology in 1990. He commenced employment at the then Aeronautical Research Laboratory in 1989, firstly as a vacation student, then in 1990 as a cadet engineer, and finally as a graduate engineer. During this time he has worked in several areas, including air combat analysis, aircraft turbojet engine performance and testing, aircraft structural fatigue and fracture repair, and helicopter flight dynamics. He is currently undertaking work involved with the analysis of air-to-air combat.

A. M. Arney
Air Operations Division



Ashley Arney graduated from the University of Sydney in 1981, having obtained an Aeronautical Engineering Degree, with honours. Since commencing employment at the then Aeronautical Research Laboratory in 1982, he has been involved with the mathematical modelling of the performance and flight dynamics of a wide range of helicopters. He has also obtained extensive experience in trials, data processing, and the use of such data for development of the appropriate mathematical model. More recently he has been involved in modelling the helicopter/ship dynamic interface, and gathering data for development purposes.

CONTENTS

1. INTRODUCTION

This report details the graphical display program *ffgView*, which allows the user to view in three dimensions a vector display of airflow around an 'Adelaide' class FFG-7 frigate. This package can readily be adapted to display airflow around any object by replacing the frigate geometry file with a file representing the required object. Wind vectors are displayed in colour to enable easy recognition of velocities, and the view can be manipulated using a mouse to provide the user with any desired viewing angle. The program is coded in *C* on a Silicon Graphics® (SG)[1] computer.

The graphical display package was designed to enable the display of airwake data collected during trials in 1989 on a Royal Australian Navy 'Adelaide' class frigate (see Refs 1 and 2). The 'Adelaide' class is based on the US FFG-7 class frigate and for the purposes of airflow over and around the flight deck the two classes are identical. With the availability of wind tunnel data for the frigate (Ref. 3), it was considered desirable to be able to display this and perhaps other types of data that may become available in the future. For this reason, a general data file format for input into the graphical display program was chosen. Conversion programs were written to deal with the data format used for the full scale data as well as that used for the wind tunnel results.

The display package *ffgView* has the following capabilities:

- Display in colour an orthogonal or perspective view of the flight deck of an FFG-7.
- Display the airwake patterns around the frigate using coloured vectors.
- Allow the input of data obtained from any source conforming to the specified input file format.
- Provide utilities allowing the conversion of raw data to the required input file format.
- Allow three-dimensional rotation, translation, and scaling of the view.
- Provide a means of saving and printing screen images.

The limitations of this package are as follows:

- It can only be run on machines supporting SG IRIS® Graphics Library™ (GL) utilities, i.e. SG machines.
- Screen images can only be saved in Graphics Industry Format (GIF) or SG Red-Green-Blue (RGB) format. An image saved in GIF format is easily transportable between machines (including Macintosh and IBM compatible personal computers).

Other features of importance to the functioning of *ffgView* are explained throughout this report. To facilitate easier modification in the future, details of the structure of the code are given.

Imperial units are adopted in this report because (a) they are used exclusively by research workers in the US with whom Air Operations Division (AOD) staff are collaborating and (b) both the helicopter and ship referred to are built in the US to imperial unit specifications.

2. PROGRAM INFORMATION

2.1 Input File Formats

2.1.1 Geometry File Format

The geometry file is used by the graphics program to define the shape, size, and colour of the frigate (or other object) about which the airflow vectors are to be displayed. The shape of the frigate is defined by a number of polygons. This geometry file format, developed by AOD, is similar to that used by SG for some of their demonstration programs. A viewer for these AOD geometry files had previously been written (see Ref. 4) and was used as the basis for the sections of the code that display the frigate.

[1] Silicon Graphics, Inc. Mountain View, California, USA.

All data within the geometry file have units of feet and the origin is at the bullseye on the flight deck (on the ship centreline 42.3 ft aft of the hangar face). See Section 4.3 for a definition of the axis system which is used to draw the frigate.

The structure of a typical geometry file is as follows:

- Line 1 contains two integer numbers. The first, `num_verts`, is the number of polygon vertices included in the file. The second, `num_polys`, is the number of polygons formed by these vertices.
- The following `num_verts` lines contain an integer (the vertex number, not used by the program but included to make the file more readable) and three real numbers. The three real numbers are the *x*, *y*, and *z* coordinates of each vertex measured in feet from the bullseye on the flight deck.
- The next two lines contain information about the material types (degrees of opaqueness, shininess, etc.) of the polygons to follow. The first number in the first line contains the number of times the material type will be changed when the object is drawn. The remaining numbers on that line are the numbers of polygons to be made from the material specified by the integer value on the line directly below it. This is more clearly illustrated by the example below.
- The rest of the file contains information about the polygons which compose the frigate. The first number indicates the number of vertices that will compose the polygon. This is followed by the vertex number of each of the vertices which comprise that polygon. Note that each polygon may have as many as 256 sides.

Below is an example of a simple geometry file representing a cube, as shown in Fig. 1. Comments not included in the actual data file are shown in italics.

```
 8 6                            Number of vertices and number of polygons

1   0.0    0.0    0.0           Vertex 1 at x=0.0, y=0.0, z=0.0
2   0.0    1.0    0.0           Vertex 2 at x=0.0, y=1.0, z=0.0
3   1.0    1.0    0.0
4   1.0    0.0    0.0
5   0.0    0.0    1.0
6   0.0    1.0    1.0
7   1.0    1.0    1.0
8   1.0    0.0    1.0           End of vertex data

3   4      1      1             3 Sets of material data to follow.  Make four
    5      3      7             polygons material 5, one material 3, and one 7.

                                      Start of polygon data :
4   1      2      3      4            4 sided polygons made by joining
4   5      8      7      6            vertices 1 2 3 4, 5 8 7 6, etc.
4   1      4      8      5
4   4      3      7      8
4   2      6      7      3
4   1      5      6      2
```

The order in which the vertices of each polygon are connected is important. The lighting model will decide how the polygons are to be shaded by using calculations involving the polygon normals. For the model to be shaded correctly, the polygon normals must face towards the viewer. To accomplish this the polygon vertices are numbered in a counter-clockwise direction when viewed from the front of the polygon, as shown in Fig. 1.

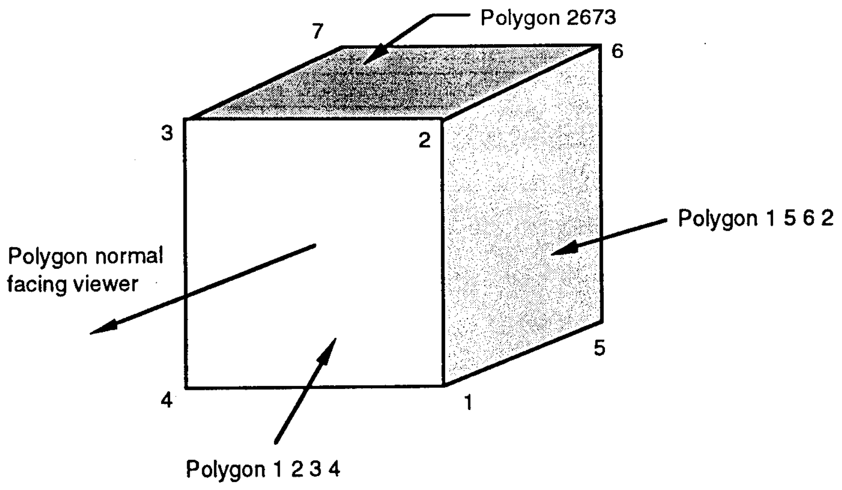


Figure 1. Numbering of Polygon Vertices

2.1.2 Velocity Vector File Format

Three types of velocity vector data file are used. These are ‘Shiptrial’, ‘Windtunnel’, and ‘Comparison’ which respectively give results of the full scale test, wind tunnel test, and comparisons of the two. The ‘Shiptrial’ and ‘Windtunnel’ types are created using the utility program *FFGWake*, described in Appendix A. The ‘Comparison’ data file type is created in one of two ways. It can be the product of program *windcomp* (Appendix A) or it can be made by editing together one or more of the other file types. This will allow the user to view several sets of data from a full scale trial conducted at the same yaw angle but at varying speeds simultaneously or to view a comparison between the full scale data and the wind tunnel data.

A typical velocity vector data file for the full scale trial is shown below. Comments not included in the actual data file are shown in italics. This example data file contains only nine points. In reality most trials involved the collection of data at 39 points. A wind tunnel data file will normally contain data at more than 2000 points.

```
Shiptrial     Type of Data File
          9   Number of Data Points in this File
X, Y, Z Velocities (ft/s)     X, Y, Z Positions (ft)     Row Column Layer
-3.022  -19.100  -22.776     0.000     0.000   31.500    1     1      1
-1.518   -8.747   -4.147     0.000     0.000   21.000    1     1      2
-3.219    2.672   -4.728     0.000     0.000   10.500    1     1      3
-7.640  -19.613  -29.240     0.000   -10.500   31.500    1     2      1
-3.732  -14.857  -19.400     0.000   -10.500   21.000    1     2      2
-5.925   -9.048  -11.495     0.000   -10.500   10.500    1     2      3
-7.390  -19.661  -33.316     0.000   -17.060   31.500    1     3      1
-4.867  -17.142  -31.245     0.000   -17.060   21.000    1     3      2
-7.881  -14.103  -27.667     0.000   -17.060   10.500    1     3      3
-3.500 -18.00          Free stream x and y components ('Shiptrial' & 'Windtunnel')
-3.243 -19.123         Aft anemometer x and y components ('Shiptrial' only)
```

An example of a ‘Comparison’ type for two sets of full scale data is shown below. Due to the large number of data points in each data set (41), some intermediate lines of data have been omitted for clarity.

```
Comparison        Type of Data File
2                 Number of Data Sets
41  030deg10kn[1] Number of Data Points in First Data Set and Legend Title
41  030deg20kn    Number of Data Points in Second Data Set and Legend Title
-15.946    7.733   2.454  23.850 -17.060  31.500   1  1  3    First Data Set
 -6.464    3.193   0.951  23.850 -17.060  21.000   1  1  2
 -1.661    2.908   0.546  23.850 -17.060  10.500   1  1  1
-14.401    8.641  -0.054   0.000   0.000  31.500   2  3  3
 -8.089    2.605  -0.787   0.000   0.000  21.000   2  3  2
                                  .

                                  .
-16.84029 7.086    0.000  50.000   0.000  50.000   1  1  1    First Data Set Free Stream
-13.47363 5.909    0.000 -40.000  -2.100  13.330   1  3  1    First Data Set Aft Anemometer
-22.767   19.096   3.035   0.000   0.000  31.500   2  3  3    Second Data Set
 -4.161    8.730   1.525   0.000   0.000  21.000   2  3  2
                                  .

                                  .
-30.397   21.043   0.000  50.000   0.000  50.000   1  1  1    Second Data Set Free Stream
-20.662   13.050   0.000 -40.000  -2.100  13.330   1  3  1    Second Data Set Aft Anemometer
```

Referring to the examples above, the first line of all types of velocity vector file contains a keyword (‘Shiptrial’, ‘Windtunnel’, or ‘Comparison’) identifying the type of file.

For data file types ‘Shiptrial’ and ‘Windtunnel’ the second line of the data file contains a single integer which indicates the number of data points contained within that file. The second line of a ‘Comparison’ data file will contain the number of data sets contained in that file. This can range from two (e.g. a wind tunnel run compared with a full scale test, or two sets of full scale data for the same direction at different speeds) but can be as large as five. The next lines in a ‘Comparison’ file will contain a single integer which indicates the number of data points in each set, along with a comment which will be displayed in the legend(Section 3.2).

For all file types the following lines will each have six real numbers and three integers. The six real numbers are, in order, the x, y, z components of velocity at the point, and the x, y, z positions of that point. The three integers correspond to the longitudinal, lateral, and vertical rows in which that particular data point is to be found. This information is used by *ffgView* when displaying sections of data as opposed to displaying every data point. As both the full scale and wind tunnel data were collected in a grid, it is a simple matter to number the rows, columns, and layers so that they can be displayed individually using the display program.

For data file types ‘Shiptrial’ and ‘Windtunnel’ the next line contains the x and y coordinates of the free stream velocity. This is measured at a reference anemometer for the full scale data and at an upstream reference point in the case of the wind tunnel data. The full scale data file concludes with the x and y velocity components from the ship reference anemometer.

2.2 Source Code

2.2.1 Modules

The source code for *ffgView* is split into several source modules. These files each contain several *C* functions which are related. This modularity allows for ease of modifying and recompiling the source code. The source files are as follows:

[1] Indicates a wind 30 deg to starboard at 10 kn relative to the ship.

ffgMain.c

This file contains all of the basic drawing routines as well as the principal *C* function '`main()`', which controls the basic functioning of the program.

ffgDisplay.c

The functions for drawing the basic screen and the menu windows, legend, help screens, etc. are contained in this file.

ffgUtils.c

This file contains all of the functions for freeing the dynamically allocated memory and for exiting the program. It also contains the functions which control any input by the user which needs to be typed at the keyboard.

ffgMenu.c

This file contains all of the functions required for drawing the main menu screen and its icons, and the functions which control the logic of the operation of the menu.

ffgInit.c

All of the functions which initialise variables and define material types are included within this file.

ffgRead.c

This file contains functions for reading the geometry and the velocity data files and setting some of the associated parameters. All of the dynamic memory is allocated in this file.

Also necessary for compiling the source code are the header files ffgMain.h, ffg.h, ffgMaterials.h, ffgTypedefs.h, ffgHeaders.h, and ffgColorvecs.h. The header files contain definitions and constants assigned for the entire program. If the header files are included in the count there is a total of slightly more than 2600 lines of code comprising *ffgView*.

2.2.2 Compiling

To expedite compilation, a *makefile* (shown below) which utilises the UNIX 'make' facility was written. It is necessary to link in the shared graphics libraries (-lgl_s), the mathematics libraries (-lm), and the dynamic memory allocation libraries (-lmalloc) when the program modules are linked. Also important is the need to use the switch '-cckr' in the compilation of the modules. This program was written originally using traditional *C*. During programming, the compiler was updated and became fully ANSI *C* compatible. This tightened up some of the variable checking and caused errors in sections of the code where pointers to structures were used (quite validly) when pointers to arrays were expected. The '-cckr' flag caused the ANSI compiler to compile in traditional *C* and avoid this problem.

```
#
#  Makefile for ffgView
#  C. Heinze  :  AOD DSTO
#

O   = ../obj#
S   = ../src#
R   = ~/clint/airwake/ffgc#
H   = ../src/ffgHeaders#
OBJ = $(O)/ffgMain.o $(O)/ffgDisplay.o $(O)/ffgInit.o \
      $(O)/ffgRead.o $(O)/ffgMenu.o $(O)/ffgUtils.o
H1  = $(H)/headers.h $(H)/ffgDefines.h $(H)/ffgTypedefs.h
H2  = $(H)/ffgMaterials.h $(H)/ffgColorvecs.h
H3  = $(H)/ffg.h
H4  = $(H)/ffgMain.h

# Executable
#
$(R)/ffg : $(OBJ)
       cc $(OBJ) -lm -lmalloc -lgl_s -o  ffgView
       mv $(S)/ffgView $(R)/ffgView
#
# Modules
#
$(O)/ffgMain.o :  $(S)/ffgMain.c $(H1) $(H2) $(H4)
       cc -g -cckr -c $(S)/ffgMain.c
       mv $(S)/ffgMain.o $(O)
$(O)/ffgDisplay.o :  $(S)/ffgDisplay.c $(H1) $(H3)
       cc -g -cckr -c $(S)/ffgDisplay.c
       mv $(S)/ffgDisplay.o $(O)
$(O)/ffgRead.o :  $(S)/ffgRead.c $(H1) $(H3)
       cc -g -cckr -c $(S)/ffgRead.c
       mv $(S)/ffgRead.o $(O)
$(O)/ffgInit.o :  $(S)/ffgInit.c $(H1) $(H2) $(H3)
       cc -g -cckr -c $(S)/ffgInit.c
       mv $(S)/ffgInit.o $(O)
$(O)/ffgMenu.o :  $(S)/ffgMenu.c $(H1) $(H3)
       cc -g -cckr -c $(S)/ffgMenu.c
       mv $(S)/ffgMenu.o $(O)
$(O)/ffgUtils.o :  $(S)/ffgUtils.c $(H1) $(H3)
       cc -g -cckr -c $(S)/ffgUtils.c
       mv $(S)/ffgUtils.o $(O)
```

The *makefile* shown above was for an existing area on the machine on which the program was developed. To implement the *makefile* on a new machine or in a new area, it is necessary to change the values to which O, S, R, and H are set to reflect the locations of the object code, source code, executable (Run) code and the header files.

3. RUNNING *ffgView*

The program *ffgView* is run from the UNIX prompt by typing `ffgView`, followed by the name of the file containing the velocity vector information. If the name of the file is not included, the program will prompt for it. Once the program is run a representation of the flight deck of an FFG-7 frigate will be seen, along with the basic menu screen. Examples of displays for full scale and wind tunnel data are shown in Appendix B. The program may then be controlled by a variety of mouse and keyboard commands described below.

3.1 Mouse Operation

The mouse is used to control the selection of menu items. Clicking the left mouse button when the cursor is over an icon will result in that icon option being initiated. If the mouse is not in the menu area, holding one of the mouse buttons and dragging the icon horizontally across the screen will result in the following. If the ROTATE option is selected, the display of the frigate can be controlled using the mouse buttons to rotate around the frigate. If the TRANSLATE option is selected, the display will translate along the x, y, or z axes if the left, center, or right button respectively is depressed. If the SCALE option is selected, the display will be scaled smaller or larger by depressing any of the mouse buttons and dragging the mouse. The operation of each of these functions is described in more detail below.

3.2 Menu Options

The following menu options are available:

ROTATE

This option changes the view point control to rotate mode, allowing rotation and scaling of the view of the frigate using the three mouse buttons. Originally the program was structured so that the three mouse buttons would control rotations about the three axes. This was changed so that the left mouse button controls the azimuthal angle, the center mouse button controls the elevation angle, and the right mouse button controls the scale factor or the distance from the viewer to the frigate. If the viewing mode is orthogonal, then the right mouse button controls the scale factor and thus zooms in or out on the frigate. If the viewing mode is set as perspective, then the right mouse button controls the distance of the viewer from the frigate.

TRANSLATE

This option changes the view point control to translate mode, allowing movement of the view in the directions of the three axes using the three mouse buttons.

SCALE

This option changes the view point control to scale mode which will zoom in or out from point (0,0,0), the center of the flight deck, using any of the mouse buttons. This function was originally created when the ROTATE function did not include a scale option with the right mouse button. It is now almost redundant. However, if the perspective viewing mode has been selected, this option can be used to scale the picture in place of using the right mouse button in the ROTATE option to move closer to the frigate. The scale function is activated by clicking and holding any mouse button. If the cursor is in the exact center of the screen there will be no scaling. If the mouse is moved to the left there will be a reduction in the size of the frigate. If the mouse is moved to the right the frigate will be scaled larger. Moving the mouse further to the left or right will increase the rate at which the scaling occurs.

SCREEN

This option selects the background colour of the screen display. The default colour is pale green, which was chosen as it is easy on the eye and provides clear definition of the wind vectors.

PRINT

This option selects the background colour of the screen display to be white and incorporates black text and menu borders. This selection gives the best results when using colour printers.

DRAW FRIGATE

This option toggles the drawing of the frigate on or off. Thus it is possible to examine the wind vectors without the presence of the frigate. This is useful if it is wished to view the flow over the deck from beneath, where the presence of the frigate may obscure the view. The default is drawing the frigate.

WIRE FRAME

This option displays the frigate in 'wire-frame' mode. This involves drawing only straight lines describing the edges of the polygons. Again, this gives the opportunity to observe the velocity vectors from previously obscured locations by rendering the frigate effectively transparent.

FLAT SHADED

This option draws the frigate using flat shaded polygons. This is the default drawing style.

GOURAUD

Gouraud shading is a function which blends colours in a special way in an attempt to remove the tessellated appearance which is often apparent in objects such as the frigate when they are drawn with relatively few polygons. This feature was included although its use is not recommended since the flat shaded polygons tend to provide a more even background against which the velocity vectors can be viewed.

WIND VECTORS

This option toggles drawing of the wind vectors on or off. The default is draw the vectors.

LEGEND

This option toggles the display of the legend on or off. When drawn, the legend appears in the upper left corner of the screen. When viewing a 'Comparison' file, the legend indicates which velocity vector colour refers to the respective velocity data file. When viewing a 'Shiptrial' or 'Windtunnel' file, the legend indicates a colour corresponding to a given velocity range. The default is do not display the legend.

FILE INFO

This option toggles the display of the file information on or off. When on, information about the currently displayed file appears in the lower left corner of the screen. Information includes the file name, number of vectors in that particular file, type of data file, and statistical information about the maximum and minimum velocities involved. The default is do not display file information.

OTHER FEATURES

This option toggles the display of a help screen on or off giving information on the commands which are accessed by the keyboard. A summary of these commands is included in Table 1. The default is do not display help screen.

ABOUT *ffgView*

This option toggles the display of information about the present version number of *ffgView*. The default is do not display information.

LOAD FILE

This option prompts the user for the name of a velocity vector data file which is to be displayed.

IMAGE FILENAME

This option prompts the user to enter the name of the image file to be saved. An '.rgb' or a '.gif' extension will be added automatically, depending upon how the image is saved.

SAVE AS RGB

This option causes the screen image to be saved as an RGB file. The file will have the name selected by the user plus an '.rgb' extension. If a filename has not been selected, the default filename is 'file_1.image.rgb'.

SAVE AS GIF

This option causes the screen image to be saved as a GIF file. The file will have the name selected by the user plus a '.gif' extension or, similarly, the default 'file_1.image.gif'. GIF images are likely to be compatible with commercially available software.

JUMP TO VIEWPOINT

This option prompts the user for the azimuth angle, the inclination angle, and the scale factor. Once these data have been entered, the viewpoint will immediately jump to the entered values.

QUIT

This option terminates *ffgView* after first freeing allocated memory and closing the graphics window.

Table 1. Keyboard Commands for *ffgView*

A	Toggle the display of the drawing axes. Default is no axes drawn.
B	Toggle *back-facing* on and off. Back-facing is a SG drawing function which improves drawing times by not drawing polygons which do not face the viewer. The default value is off.
C	Toggle scaling of the velocity vectors. If two or more full scale airwake files are compared, the velocity vectors will be scaled according to the ratio of the respective reference airwake velocities. Default is no scaling.
D	Display only those vectors which are in the proximity of the flight deck. This is useful for cutting down the displayed information for the wind tunnel data. Default is to display all data.
F	Toggle the type of vector which is drawn. The choices are (i) a straight line vector, (ii) a line with an arrow head, and (iii) a line with a fat base resembling a wind tunnel wool tuft. Default is option (i).
H	Display the panel normals on the model of the frigate. If the frigate is in flat shaded mode, the panel center normals are displayed. In Gouraud shaded mode, the panel corner normals are displayed. Default is do not display normals.
L	Label either the panels or the vertices on the model of the frigate with their appropriate number. The V key determines which of these two options is labelled by toggling between them.
J	Enter the new inclination and azimuth angles to which the viewpoint will jump.
P	Toggle the viewing mode between an orthogonal three dimensional view and a perspective three dimensional view. The orthogonal view is the default.
Q	Cycle through the airflow data displaying sections through the flow in the *yz* plane of the wind axes.
W	Cycle through the airflow data displaying sections through the flow in the *xz* plane of the wind axes.
E	Cycle through the airflow data displaying sections through the flow in the *xy* plane of the wind axes.
R	Display all data. Overrides P, Q, and R and reverts to showing all available data. Does not reset the D key.
S	Force all of the airwake vectors to be the same length. Default is display the vectors at a length proportional to their respective velocity magnitudes.
V	Toggle labelling of panels or vertices. See L key. Default is panels labelled.
Z	Toggle *z-buffering* on and off. Z-buffering is a SG function which controls hidden surface removal when drawing three dimensional images. The default is on.
<esc>	Quit *ffgView*.

In general if a menu icon is activated, it and its associated text will be darkened, indicating that the particular option is currently selected. In the case of the options 'Save as RGB' and 'Save as GIF', there will be a message 'Saving in Progress' displayed for as long as it takes for the program to save the screen to a file.

3.3 Other Features

The features described above are the basic controlling functions for *ffgView* and as such are accessed by the mouse-driven menu. There are many other controls and functions which may be accessed via the keyboard. These features are generally more specialised and are not likely to be used as often as those accessed by the mouse. Many, but not all, relate to the drawing of the frigate and are remnants from the original program which was designed to display only a single three-dimensional object.

Some of the features are not expected to be used often, but were simple to implement. This is true of the facility which allows the user to alter the view of the frigate from an orthogonal view to a perspective view. Experience with the program tends to suggest that the orthogonal view is more useful. Many of these features do not relate specifically to the viewing of airwake data but may be useful if it becomes necessary to modify the displayed frigate or to replace the frigate with some other object about which airflow data were obtained. Table 1 describes all of the commands which may be accessed via the keyboard.

4. STRUCTURE OF *ffgView*

A single main loop controls *ffgView* (Fig. 2), handling all keyboard and mouse events (otherwise known as inputs). If the left mouse button is clicked, a test is made to determine whether the mouse is in the vicinity of the menu. If the cursor is over the menu and the left mouse button is clicked then control is passed to a function called `icon_control` which controls and schedules the actions to be performed. If the cursor is not over the menu, the clicking of the left or any other mouse button is taken to be a SCALE, ROTATE, or TRANSLATE command and action is taken accordingly.

Following the processing of keyboard and mouse inputs, a call to the function `draw_all()` is made. This function is responsible for the drawing of all of the objects on the screen. The program will continue to loop, reading keyboard and mouse events and drawing the screen image until a call to `ffg_exit()` is made. This function is activated by hitting the escape key, or clicking the quit icon in the menu, which will result in the termination of the program.

Note that the program utilises an SG feature known as 'double buffering' in the drawing of the graphics screen. Whenever an object is drawn, it is drawn in a buffer, not directly on the screen. When the drawing is finished, the buffer which now contains the updated drawing is swapped for the buffer containing the information which is currently being displayed. This means that the new image can be flashed to the screen in a very short time, resulting in faster updating and smoother movements of the image being displayed. The SG function which handles the swapping of the backbuffer with the currently displayed front buffer is `swapbuffers()`, as referred to in Fig. 2.

The program incorporates two C data structures[1]. One of these data structures contains all of the information which is required by the program to draw the frigate. The other contains all of the velocity information which is needed to draw the velocity vectors.

These structures are defined and initialised in 'ffgTypedefs.h'. Many of the elements of both structures have their memory allocated dynamically and therefore there is a corresponding freeing of this memory undertaken as a part of the function `ffgexit()`.

The following subsections detail various aspects of the program structure in greater detail.

[1] For those unfamiliar with C programming, a data structure is a collection of many variables which are stored and may be passed to functions by referencing a single name. In Fortran, the only comparable feature is the *common block*, and in Pascal a similar feature is the *record*.

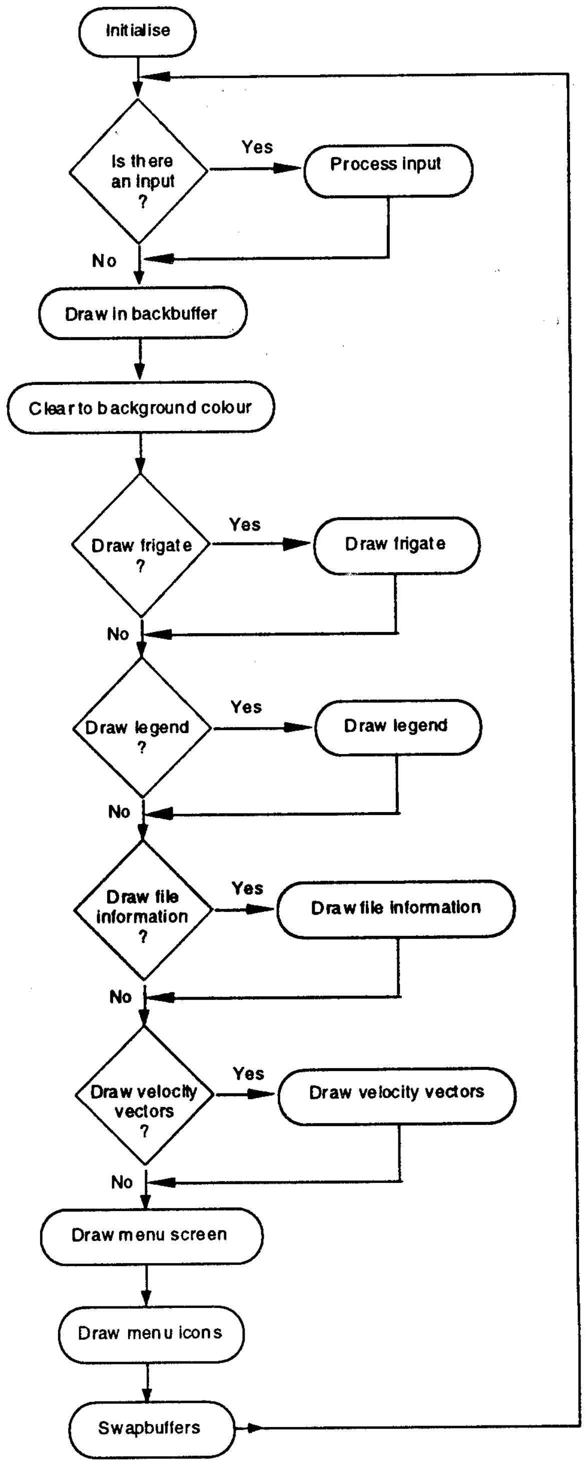


Figure 2. Basic Program Structure of *ffgView*

4.1 Initialising the Graphics

The following steps are undertaken to initialise the graphics:

- A call to the function `init_graphics()` is made, which in turn calls `init_window()`, `init_view()`, `queue_devices()`, and the two functions which control the initialising of the lighting model, `def_simple_light_calc()` and `use_simple_light_calc()`.
- `init_window()` opens the graphics window.
- `init_view()` sets the perspective or orthogonal viewing factors to their initial values and sets the initial viewing angle. The viewing distance may later be adjusted by the user invoking the 'SCALE' option (section 3.2).
- `queue_devices()` queues each of the devices required by *ffgView*. These include the mouse buttons and cursor locations, keyboard keys, and the window controlling events.
- `def_simple_light_calc()` and `use_simple_light_calc()` initialise the lighting model and define the positions and colours of any light sources which illuminate the frigate.

4.2 Dynamic Memory Allocation

Because *ffgView* requires between thirty and several thousand velocity vectors to be displayed, it is efficient to allocate the memory required for storing the data at run time. Thus the dynamic allocation of memory becomes important at several stages of program execution. Memory is dynamically allocated for both the data concerned with the drawing of the frigate and the velocity vector data. Primarily, this is achieved by defining data structures which contain pointers to memory that is allocated when the size requirements of that memory are known.

4.2.1 Frigate Data Structure

The frigate geometry data structure is defined in 'ffgTypedefs.h' as follows :

```
typedef struct {
                Coord x;
                Coord y;
                Coord z;
                } Point3d;
typedef struct {
                short nsides;
                short *pnts;
                } Polygon;

typedef struct {
                char object_name[40];
                short num_vertices;
                short num_polygons;
                Point3d *vert;
                Point3d *norm;
                Point3d *gnorm;
                Polygon *poly;
                } model;
```

During execution, the geometry data are read in and when the numbers of polygons and vertices are known, the storage for the geometry data is allocated. The single structure `model` (the third structure shown in 'ffgTypedefs.h') contains most of the information required to draw the frigate.

4.2.2 Velocity Vector Data Structure

The velocity vector information structure is defined in 'ffgTypedefs.h' as follows :

```
typedef struct {
                char vect_file_name[40];
                short num_vectors;
                float free_x;
                float free_y;
                float free_velocity;
                float free_yaw_angle;
                Boolean *draw_this_one;
                float *vel_mag;
                Point3d *vel_orig;
                Point3d *vel_x;
                Point3d *vel_y;
                Point3d *vel_z;
                Point3d *vel_xyz;
                Point3d *vel_1;
                Point3d *vel_2;
                Point3d *vel_3;
                int *x_layer;
                int *y_layer;
                int *z_layer;
                } Velocity_vector_data;
```

In a similar way to the frigate geometry data structure, data storage for the velocity vectors is allocated in memory once the number of velocity vectors is known. Several variables are set for each different velocity vector and so they must all be assigned memory dynamically. For each velocity vector, the following variables are required (some of these are read from the data file, some are calculated):

`draw_this_one` is a boolean variable which controls whether or not that particular vector will be drawn. This vector is originally set to true so that all vectors are drawn, but may be set to false during the execution of the program so that not all vectors are drawn.

`x_layer`, `y_layer`, and `z_layer` are three integers which are set to values that are read from the data file. These three variables store the layer numbers in which the data point occurs.

`vel_orig` is a structure of type `point3d` and stores the location in *x*, *y*, *z* space of the point at which that particular velocity is known.

`vel_xyz` is a structure of type `point3d` which stores the location in three-dimensional space of the end of the velocity vector. This is calculated by reading in the components of velocity for the point concerned and then scaling them to fit the grid and translating them to the correct location.

`vel_1, vel_2, vel_3, vel_x, vel_y`, and `vel_z` are used by the program for drawing arrow heads on the vectors and for scaling all vectors to be the same length.

4.3 Frigate Drawing Axes

The frigate drawing axes are as shown in Fig. 3. This axes set was chosen for ease of preparing the geometry file for drawing the frigate. It does not align with the axes set used to define the velocity directions in either the full scale or the wind tunnel data collections. These axes sets are shown for comparison in Appendix C. When converting the raw data to the format required by *ffgView*, it is necessary to apply coordinate transformations as dictated by these axes. This is discussed in Appendix C. The programs which perform these data manipulations and axes transformations are described in Appendix A.

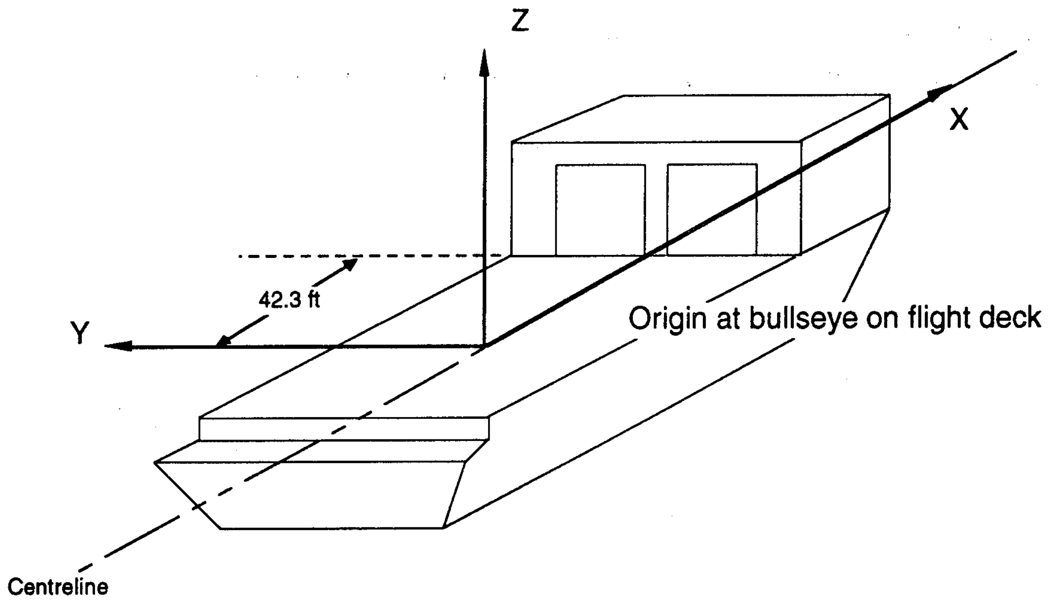


Figure 3. Drawing Axes

4.4 Frigate Geometry File

The geometry file which contains the information required by *ffgView* for drawing the frigate was composed in the axis system shown in Fig. 3. The values were obtained from scale drawings (Ref. 5) and photographs and are measured in feet. The deck and hangar markings are drawn as separate polygons that sit slightly above their respective surfaces. The Phalanx system was originally drawn as a separate entity and then included into the frigate geometry file. A geometry file for the Phalanx system alone exists should it be required for a future project.

Comment statements are included in the data file and should make the geometry file relatively easy to modify. The comments are lines that begin with a '%'. They are ignored by the program when it reads the geometry file.

4.5 Frigate Colour Information

The frigate is given colour during the drawing process by defining material types from which the polygons will be composed. These polygons then respond to the lighting model to produce the required colours and surface effects.

The material types used to compose the frigate are defined in the *C* language in 'ffgMaterials.h' as follows:

```
float material_1[]= {
                    SPECULAR, 1.0, 1.0, 1.0,
                    DIFFUSE,  1.0, 1.0, 1.0,
                    SHININESS,1.0,
                    AMBIENT,  1.0, 1.0, 1.0,
                    LMNULL
                   };
          .         .         .
          .         .         .
          .         .         .
          .         .         .

float material_5[]= {
                    SPECULAR, 0.1, 0.1, 0.1,
                    DIFFUSE,  0.1, 0.1, 0.1,
                    SHININESS,100.0,
                    AMBIENT,  0.1, 0.1, 0.1,
                    LMNULL
                   };
```

```
float material_6[]= {
                    SPECULAR, 0.35, 0.00, 0.00,
                    DIFFUSE,  0.35, 0.00, 0.00,
                    SHININESS,100.0,
                    AMBIENT,  0.35, 0.00, 0.00,
                    LMNULL
                   };
float material_7[]= {
                    ALPHA, 0.6,
                    SPECULAR, 0.20, 0.20, 0.20,
                    DIFFUSE,  0.20, 0.20, 0.20,
                    SHININESS,100.0,
                    AMBIENT,  0.20, 0.20, 0.20,
                    LMNULL
                   };
float material_8[]= {
                    SPECULAR, 0.80, 0.80, 0.80,
                    DIFFUSE,  0.80, 0.80, 0.80,
                    SHININESS,100.0,
                    AMBIENT,  0.80, 0.80, 0.80,
                    LMNULL
                   };
```

These materials are defined by their shininess and by three other properties (specular, diffuse, and ambient) which provide information as to the manner in which they reflect light (for more detail see Ref. 8). These properties are controlled by three floats (having a value between 0 and 1) which define the fraction of red, green, and blue light respectively, associated with that property. For instance `material_1` (above) is a purely white material since all three floats have a value of 1. This material is used for the deck markings.

It should be noted that `material_7` is slightly different in that it has an additional value defined by the word ALPHA and that this is set to 0.6. This material is the one used to colour the polygons which comprise the safety nets around the sides of the flight deck of the frigate. In order to retain a 'net-like' look, these polygons are made to appear translucent. The degree of translucency is controlled by the ALPHA value where 0.0 is totally transparent and 1.0 is totally opaque. Thus the nets appear to have a translucent grey colour. As can be seen in Appendix B, the bottom of the hull may be seen through the 'nets'.

4.6 Drawing the Data Entry Boxes

The data entry boxes which are required for entering file names and viewing angles are drawn in the overlay planes. Most SG machines have designated memory which is known as the overlay planes. Overlay planes are areas into which it is possible to perform limited drawing which, when erased or altered, leaves the image underneath intact. Commonly, the overlay planes are used for drawing pop up menus or other similar items which are temporary and do not require complex drawing techniques. This process is initiated by calling the function `overdraw()`. The limitation of overlay mode is that there is a restriction of only two different colours. For the purposes of data entry, this restriction is acceptable and the overlay planes have the advantage that the data entry box can be erased, revealing the original drawing underneath without requiring a redraw.

4.7 Drawing the Velocity Vectors

The data file which is used by *ffgView* to draw the velocity vectors contains information as to the *x*, *y*, *z* position of a particular vector and the *x*, *y*, *z* components of velocity at that point. This information is read in by the program, scaled, and then stored in the dynamically allocated memory as discussed above. The function which draws the vectors is `draw_vel_vectors` in the file 'ffgMain.c'.

The value of `vvd->draw_this_one[i]` is first tested for each vector. If it is true, the vector is drawn. If this variable is false the vector will not be drawn and the following steps are ignored.

The vector is tested for size by comparing the variable `vvd->vel_mag[i]` with `vel_mag_max`. Depending upon this value, a call to a SG function `c3f()` will be made to change the colour to that which is appropriate for its magnitude. The program then checks on the status of a number of variables which will determine the nature of the vectors to be drawn. This includes the option to draw the vectors with and without arrow heads and to have all of the vectors the same length rather than the default, which has them scaled according to their relative magnitude.

The vectors, including the arrow-heads if required, are then drawn with a simple series of move and draw commands.

4.8 Drawing the Information/Menu Screens

The view of the frigate and the velocity vectors is drawn in three dimensional space. When it is necessary to draw a flat menu screen or an information screen, it is much easier and preferable to draw these in a separate two-dimensional axes system. This is performed by saving a copy of the viewing matrix and the projection matrix that were used for the three-dimensional drawing and then making a call to the `ortho2()` function, which allows us to define a two-dimensional co-ordinate system in which to draw. After the two dimensional drawing is finished, the projection and viewing matrices are reinstated by calling the function `pushmatrix()`.

4.9 The Lighting Model

In much the same way as materials are defined, it is possible to define lights of a certain colour at various locations in order to provide lighting for viewing the object. Further details on lighting models may be found in Ref. 8.

For *ffgView*, the lighting model currently appears as :

```
float local_white_light[]=
                  {
                   LCOLOR, 1.0, 1.0, 1.0,
                   POSITION, -1.0, 1.0, 1.0, 0.0,
                   LMNULL
                  };
float local_white_light_2[]=
                  {
                   LCOLOR, 1.0, 1.0, 1.0,
                   POSITION,  1.0, -1.0, 1.0, 0.0,
                   LMNULL
                  };
```

These are defined in 'ffgMaterials.h' and are utilised in the file 'ffgInit.c'.

4.10 Mouse and Keyboard Inputs

The control of mouse and keyboard inputs is handled by the function, `process_event()`, which takes an event from the event queue and then decides how to process that event. If the left mouse button is clicked, the program will check to see if the cursor is over the menu area. If it is, then control will be passed to the function `icon_control()` which will decide upon the action to take according to which icon was clicked. If the left mouse button is not over the menu area, then the input will cause the image to be rotated or scaled as described in Section 3.1. Those keys listed in Table 1 will also enter the event queue and will be handled by the function `process_event()`.

4.11 Exiting *ffgView*

Exiting *ffgView* is initiated by calling a function `ffgexit()`. This in turn calls four other functions. The first two of these are SG functions, `gexit()` and `greset()`, which shut down the graphics in an orderly fashion. The second two, `free_malloced_velocity_stuff` and `free_malloced_geometry_stuff`, are functions contained in the source which are responsible for freeing up any of the dynamically allocated memory.

5. CONCLUDING REMARKS

The frigate airwake viewer is successfully running on an SG workstation. It has already proved useful in the early analysis of available data and will continue to be an important tool for a variety of analysis tasks within Air Operations Division. The colour images, created by *ffgView,* of wind vectors about an FFG-7 may be printed and used for presentations, reports, or closer inspection.

With modification the program could be used for a wider variety of tasks than the specialised frigate airwake analysis for which it is currently being used.

REFERENCES

1. Arney, A. M., Blackwell, J., Erm, L. P., and Gilbert, N. E., "A Review of Australian Activity on Modelling the Helicopter/Ship Dynamic Interface," *AGARD Conference Proceedings 509, Aircraft Ship Operations*, Paper No. 20, Seville, Spain, November 1991.
2. Arney, A. M., "FFG-7 Ship Motion and Airwake Trial: Part I - Data Processing Procedures," DSTO Technical Report 0039, Defence Science and Technology Organisation, Department of Defence, Australia, July 1994.
3. Matheson, N., "A review of the Airflow About a 1/64 Scale FFG-7 Frigate Model and Correlation with Full Scale Results," Proceedings of the Eleventh Australasian Fluid Mechanics Conference, Vol. I, pp. 655-658, Hobart, Australia, December 1992.
4. Muscat, R., "Geometry File Viewer," *Source code listing (Unpublished)* , Studies and Simulation Group, DSTO Aeronautical Research Laboratory, March 1992.
5. "ABR 5419 Ship Helicopter Operations Manual Volume 1," Defence Instruction (Navy), Department of Defence (Navy Office), Canberra, Australia, April 1992.
6. Blunt, D. M., "The Conversion of a Fortran Data Plotting Program Using DI-3000 Graphics to Operation on a Macintosh Personal Computer," DSTO Aeronautical Research Laboratory Flight Mechanics TM 446, Melbourne, Australia, September 1991.
7. Blunt, D. M., "Relative Wind Measurements on an FFG-7 Class Frigate," DSTO Aeronautical Research Laboratory Flight Mechanics TM 447, Melbourne, Australia, September 1991.
8. McLendon, P., *Graphics Library Programming Guide*, Document Number 007-1210-050, Silicon Graphics, Inc., Mountain View, California, USA, 1992.

APPENDIX A - Utility Programs Associated With *ffgView*

There are three programs which accompany *ffgView*. They are used for the analysis and reduction of data from its original collected form (either wind tunnel or full scale ship trial) to the format required by *ffgView*. These programs, *readcol*, *FFGWake*, and *windcomp*, are described below in some detail.

Data from the ship trial exists in the form of '.COL' files. These files contain time history data recorded on board the ship on a number of channels (Ref. 2). A program was developed to read this type of file, to extract the relevant information from it, and to write it in a more manageable form. This program, *readcol*, produces an '.out' file which is typically an order of magnitude smaller than the '.COL' file. Means and standard deviations are also calculated for the extracted data and written to a file specified by the user as discussed further below.

Data from the wind tunnel tests are recorded in ASCII files with a '.GR1' extension. These files are small in comparison with the ship trial data, as the wind tunnel recordings are static time averaged values and consist of only a single value at each location.

FFGWake is a program which will read both the '.GR1' files from the wind tunnel testing and the '.out' files output from *readcol* for the ship trial testing. *FFGWake* outputs a file which is read by *ffgView* and contains all of the information necessary for the display of vectors over the deck of the frigate.

Program *windcomp* creates a 'Comparison' type data file, containing wind tunnel data and a single full scale data file, that can then be displayed using *ffgView*. This form of 'Comparison' file contains a reduced set of wind tunnel data which only includes data that is directly comparable with full scale data. 'Comparison' files created by editing full scale data files, rather than by using *windcomp*, contain complete data sets.

The utility programs were written in Fortran in order to maintain compatibility with existing programs such as *MacShipRefine* and *MacTRANS* (Ref. 6) which are coded in Fortran. These utilities contain no SG intrinsics and should be portable to virtually any platform.

Program *readcol*

The '.COL files produced by *MacTRANS* [1] are often large and formatted for output to a printer so that they include page headers and page breaks. For many applications, it is required to read the data into another program, e.g. *ffgView*. Having a file which is irregularly filled with text makes this difficult and so *readcol* was written to extract any required data and output the data in unbroken column format. At present the program will extract only the data which is of relevance to the production of velocity vectors over the aft of the FFG. However , the program may easily be adapted to extract different information, as outlined further below. Thirteen '.COL' files, one for each mobile anemometer position used in the trials (Ref. 2), typically contain the data for one specific relative wind condition. Having extracted the information from one '.COL' file, *readcol* then calculates the averages and standard deviations of the required data (Ref. 7), and prompts the user for another input file. When all the '.COL' files have been processed, the user is prompted for a filename which will contain the statistical information.

A '.COL' file contains many channels of data arranged in five columns with a header on every page. A typical full scale airwake '.COL' file will contain more than fifty channels of data ordered from 1 to 57 (for example) in 11 sections with five columns and one section of two columns. A sample of a truncated '.COL' file is shown in Fig. A1. Note that non-varying block values (data that is constant over the entire recording) are shown at the beginning of the '.COL' file.

As a part of the header information contained within the '.COL' file, a number (shown in Fig. A1 as the BLK NUMBER) is given which corresponds to the channel upon which those data were collected. This number is above the column of data on each page. It is this number which the program will search for in order to extract the appropriate data.

---

[1] *MacTRANS* is a utility program converted to run on Macintosh computers (Ref. 6).

The data required for the display of wind velocities include the velocities from the nine anemometers on the mast, and the data collected from the ship reference anemometer and the aft main reference anemometer (see Ref. 2 for more detail). The mobile anemometer velocity data were collected on channels 41 through 49 inclusive. The ship reference anemometer data were collected on channels 53 and 54. The aft reference anemometer was output to channels 31 and 32. An array of characters (named `findthese`) is set up with a list of these channels.

```
1 Wind 90deg, 7kn - Position 4
  File 18091617 (filtered)

  RECORDED ON 01-MAR-93    AT 10:07:58    INTEGN INT = 0.0000E+00
                                                 RUN CPU TIME =    1 MIN.   0.00 SEC.

  NON-VARYING BLOCK VALUES

  BLK NUMBER =     33          34          35          36          37
              Wind Dim    Wind Dim    Wind Dim    Wind Dim    Wind Dim
              Top (1)     Top (2)     Top (3)     Mid (4)     Mid (5)
              1.0000E+00  0.0000E+00  0.0000E+00  1.0000E+00  0.0000E+00


  BLK NUMBER =     38          50          55          56          57
              Wind Dim    Exact       BLK#        BLK#        Drop out
              Mid (6      Start Time                          Factor
              0.0000E+00  0.0000E+00  0.0000E+00  0.0000E+00  5.0000E-01


  BLK NUMBER =     58
              BLK#

              0.0000E+00

1 Wind 90deg, 7kn - Position 4
  File 18091617 (filtered)

  RECORDED ON 01-MAR-93    AT 10:07:58    INTEGN INT = 0.0000E+00
                                                 RUN CPU TIME =    1 MIN.   0.00 SEC.

  BLK NUMBER =     1           2           3           4           5
     Time     Wind Dim    Roll Acc    Pitch Acc   Yaw Acc     Pitch Att
      (s)      Low (9)   (deg/s/s)   (deg/s/s)   (deg/s/s)     (deg)
  0.0000E+00  2.4898E-01  0.0000E+00  0.0000E+00  0.0000E+00 -7.1822E-01
  5.0000E-02  2.4410E-01 -3.4761E-02  5.4870E-02  5.7887E-02 -7.1542E-01
  1.0000E-01  2.4410E-01 -4.8213E-02  7.0716E-02  7.6894E-02 -7.1166E-01
  1.5000E-01  2.4654E-01 -6.1952E-02  8.0372E-02  9.2193E-02 -7.0707E-01
  2.0000E-01  2.4166E-01 -7.4663E-02  8.2162E-02  1.0177E-01 -7.0186E-01
  2.5000E-01  2.4654E-01 -8.5279E-02  7.6593E-02  1.0477E-01 -6.9627E-01
                   .                      .                      .
                   .                      .                      .
                   .                      .                      .
                   .                      .                      .

1 BLK NUMBER =     1           2           3           4           5
     Time     Wind Dim    Roll Acc    Pitch Acc   Yaw Acc     Pitch Att
      (s)      Low (9)   (deg/s/s)   (deg/s/s)   (deg/s/s)     (deg)
  2.5500E+00  2.4166E-01  1.9184E-02 -5.9375E-02  2.5927E-02 -4.5140E-01
  2.6000E+00  2.4166E-01  1.0518E-02 -5.6743E-02  2.5190E-02 -4.5821E-01
  2.6500E+00  2.4410E-01  4.4375E-03 -5.5130E-02  2.3641E-02 -4.6662E-01
  2.7000E+00  2.3922E-01  1.7225E-03 -5.5204E-02  2.1442E-02 -4.7621E-01
  2.7500E+00  2.4166E-01  2.8623E-03 -5.7704E-02  1.9097E-02 -4.8656E-01
                   .                      .                      .
                   .                      .                      .
```

Figure A1. Sample '.COL' file

An example of the execution of *readcol* is shown below. The user inputs are in bold type.

```
% readcol
Input the filename: ../data/test.COL
Which anemometer location was this recorded at ? 2
 Executing ....

Do you want another run (y/n) ? n
Enter file name to contain statistical results: test
%
```

A sample output 'test.out' file from the above example is shown in Fig. A2. Columns one through nine in Fig. A2 contain the data from the anemometer mast which were collected on channels 41 through 49 respectively. Columns eleven and twelve contain data from the ship reference anemometer and the final two columns contain data from the aft reference anemometer.

For the example above the statistical data are written to files 'test.1' and 'test.2' (Figs A3 and A4). File 'test.1' contains the means and standard deviations of data from the extracted channels. The mobile anemometer velocities are measured as individual components in the *xyz* axes, but the ship reference anemometer and the aft reference anemometer measured the total velocity magnitude in the *xy* plane and the direction. Program *readcol* resolves the mobile anemometer data into a total velocity vector as well as a velocity vector in the *xy* plane, with appropriate direction, and outputs the statistics to file 'test.2'. More detailed information on this process is given in Ref. 7.

There have been many trials involving data collection which have yielded '.COL' files. The data recording of airflow around RAN frigates is just one example. It was considered desirable that *readcol* be non-specific, enabling it to be used for a wide variety of purposes. Should it be necessary to extract data from a '.COL' file from some other trial and for some other purpose it is a simple matter to modify the values contained within the `findthese` array.

Program *FFGWake*

Program *FFGWake* reads either the '.out' file output by *readcol*, or the raw wind tunnel data '.GR1' file and converts the information to the format required by *ffgView*. For the case of the full scale data which were recorded as time histories, a part of the processing involves the data being averaged over time to give a single final value.

The functioning of *FFGWake* is controlled by a single input file which must have the name 'FFG_input.dat'. The first line of this file is the type of data, either 'Shiptrial' or 'Windtunnel'. For the 'Shiptrial' type, the second line is the number of data files to be analysed and the remaining lines are the names of the data files. For the Windtunnel' type, the second line is the tunnel reference velocity and yaw angle, the third line is the number of data files to be analysed, and the remaining lines are the names of the data files.

```
        2     Anemometer Location Number
-.295E+01 0.725E+01 -.143E+02 0.000E+00 0.955E+00 -.211E+01 -.104E+01 -.426E+01 0.530E+01 0.116E+02 -.316E+01 0.269E+01 0.193E+02
-.261E+01 0.699E+01 -.144E+02 -.473E+00 0.902E+00 -.203E+01 -.104E+01 -.427E+01 0.537E+01 0.116E+02 -.316E+01 0.269E+01 0.184E+02
-.251E+01 0.689E+01 -.142E+02 -.852E+00 0.110E+01 -.213E+01 -.111E+01 -.428E+01 0.544E+01 0.116E+02 -.318E+01 0.258E+01 0.169E+02
```

Figure A2. Sample 'test.out' File

```
Means
=====
Pos'n u(top)  v(top)  w(top)  u(mid)  v(mid)  w(mid)  u(low)  v(low)  w(low)  V(ref) Psi(ref) V(ship) Psi(ship)
      (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)   (deg)    (kn)     (deg)

  2  -16.252   5.583  -0.392  -5.851   1.416   0.509   2.285   0.459   0.633   2.589   7.474   10.592   -1.765

      Standard Deviations
      ===================
      Pos'n u(top)  v(top)  w(top)  u(mid)  v(mid)  w(mid)  u(low)  v(low)  w(low)  V(ref) Psi(ref) V(ship) Psi(ship)
            (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)  (ft/s)   (deg)    (kn)     (deg)

  2         3.213   2.652   1.647   3.274   2.963   1.058   2.414   3.594   3.465   0.125  23.712    0.578    4.393
```

Figure A3. Sample 'test.1' File

```
Means
=====
                            TOP                                        MID                                         LOW
Pos'n   V(tot)  V(horz)  PSI(horz)  PSI(vert)    V(tot)  V(horz)  PSI(horz)  PSI(vert)    V(tot)  V(horz)  PSI(horz)  PSI(vert)
        (ft/s)  (ft/s)    (deg)      (deg)       (ft/s)  (ft/s)    (deg)      (deg)       (ft/s)  (ft/s)    (deg)      (deg)

  2     17.189  17.184   -18.960    -1.307       6.042   6.020    -13.601     4.837       2.415   2.330   -168.632    15.192

              REF                 SHIP
Pos'n     V         PSI        V         PSI
        (ft/s)     (deg)     (kn)       (deg)

  2      2.589     7.474    10.592     -1.765
```

Figure A4. Sample 'test.2' File

Example input files are shown in Figs. A5 and A6 for full scale and wind tunnel data respectively (comments not included in the actual data file are shown in italics). The input file of Fig. A5 will cause *FFGWake* to analyse 13 full scale data files with the names as shown. The order in which the files are placed in the list is unimportant as each file contains the station at which those data were recorded.

```
SHIPTRIAL     Type of data file
13      Number of files to analyse
/usr/people2/fmlama/airwake/data/shiptrial/col/18091420.out    File names
/usr/people2/fmlama/airwake/data/shiptrial/col/18091424.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091428.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091432.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091435.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091439.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091443.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091446.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091452.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091455.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091458.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091502.out
/usr/people2/fmlama/airwake/data/shiptrial/col/18091505.out
```

Figure A5. Sample Input File for Ship Trial Data

```
WINDTUNNEL              Type of data file
50.0      15.0          Velocity & yaw angle
12                      Number of files to analyse
/usr/people2/fmlama/airwake/data/windtunnel/fn01y15.grl  File names
/usr/people2/fmlama/airwake/data/windtunnel/fp00y15.grl  The first number specifies the
/usr/people2/fmlama/airwake/data/windtunnel/fp01y15.grl  location of the wind tunnel
/usr/people2/fmlama/airwake/data/windtunnel/fp02y15.grl  probe with the n or p
/usr/people2/fmlama/airwake/data/windtunnel/fp03y15.grl  indicating negative or positive.
/usr/people2/fmlama/airwake/data/windtunnel/fp04y15.grl  The second number is the yaw angle.
/usr/people2/fmlama/airwake/data/windtunnel/fp05y15.grl
/usr/people2/fmlama/airwake/data/windtunnel/fp06y15.grl
/usr/people2/fmlama/airwake/data/windtunnel/fp08y15.grl
/usr/people2/fmlama/airwake/data/windtunnel/fp10y15.grl
/usr/people2/fmlama/airwake/data/windtunnel/fp14y15.grl
/usr/people2/fmlama/airwake/data/windtunnel/fp18y15.grl
```

Figure A6. Sample Input File for Wind Tunnel Data

For the case of wind tunnel data, *FFGWake* reads the data and transforms it from wind tunnel axes to the axes system used by *ffgView* (App. C). For full scale data, *FFGwake* reads the data, averages the velocities over time (as described in Ref. 7), then transforms it from ship axes to the required axes system.

To run the program type *FFGWake* at the prompt and the program will run requiring no input from the user. The resulting output file, named 'ffg_vel_vectors', contains the desired data in the format required by *ffgView*.

Program *WindComp*

Windcomp is a program which analyses a wind tunnel data file as output by *FFGWake* and performs a series of interpolations and data reductions to allow a direct comparison with data from the full scale ship trials. The data points at which data were collected in the wind tunnel seldom correspond with data collection points from the full scale ship trial tests. The wind tunnel probe collected data at points on a grid fixed with respect to the wind tunnel and so, except for the case where the wind tunnel model was at zero yaw, there is no direct comparison available between the wind tunnel data and the full scale data.

Program *Windcomp* processes the wind tunnel data in the following way. For the full scale data there are thirteen anemometer positions, with velocity measurements at three heights for each position. The wind tunnel grid has measurements at heights corresponding to each of the full scale anemometer positions and since the wind tunnel grid is denser than the full scale data grid (Fig. A7), the wind tunnel data is interpolated (in the horizontal plane) to a position which matches each of the full scale data points.

The program finds the four closest wind tunnel data points to the anemometer location under consideration. First, the closest wind tunnel data point is determined, then the three other data points are found which form a rectangle enclosing the anemometer location (Fig. A7). If wind tunnel data is unavailable (due to the inability of the probe to measure the flow)[1] no interpolation will be attempted. Otherwise a two dimensional linear interpolation will be applied.

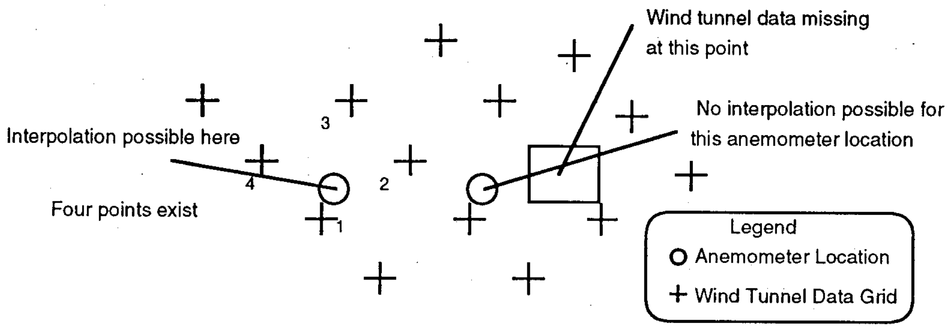

Wind tunnel data missing at this point
No interpolation possible for this anemometer location
Interpolation possible here
Four points exist
1
2
3
4
Legend
O Anemometer Location
+ Wind Tunnel Data Grid

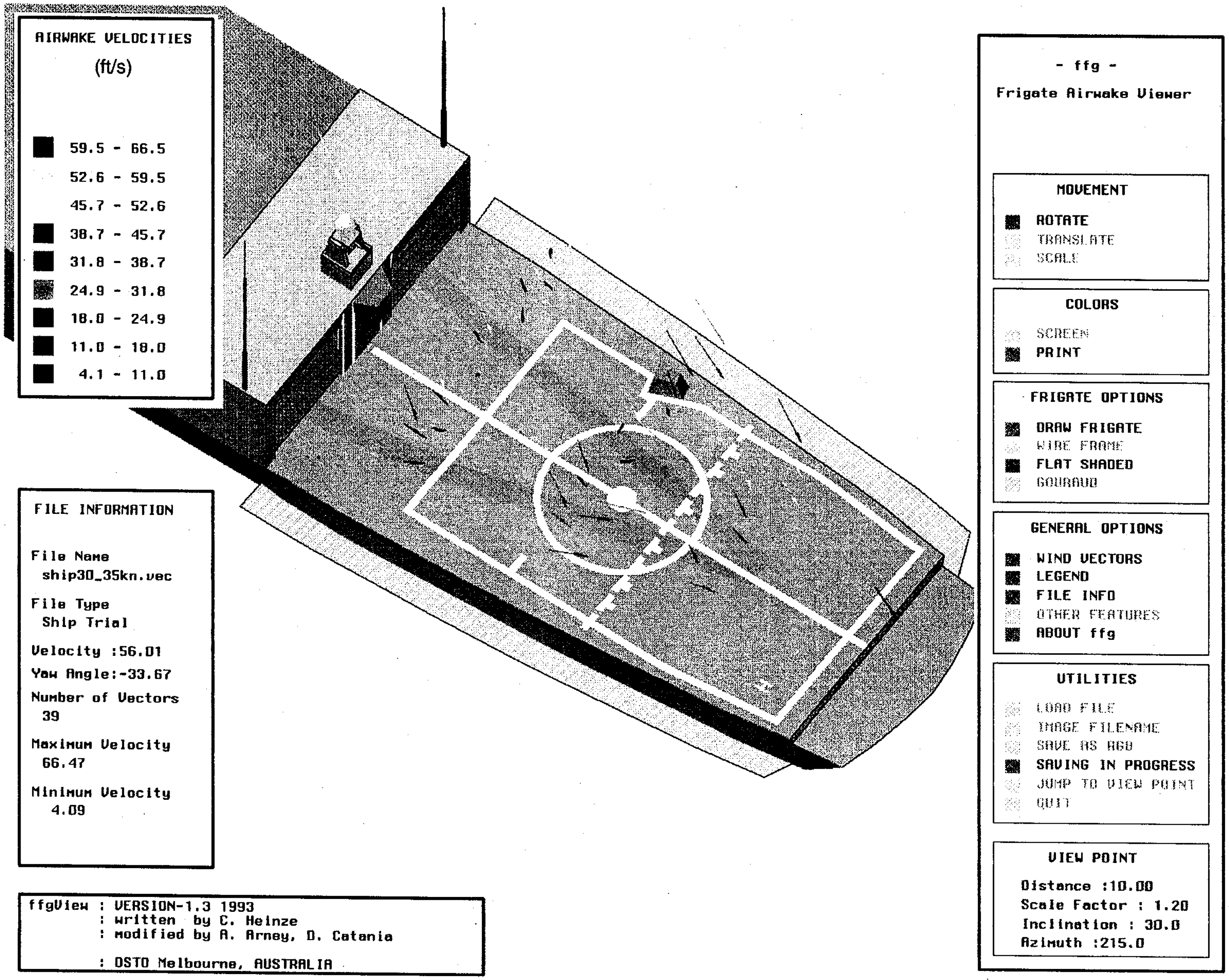
Figure A7. Interpolating Data from Wind Tunnel Data at Corresponding Anemometer Locations

To execute the program simply type `windcomp` at the prompt. The program will prompt for the yaw angle at which the wind tunnel data were recorded. The program will then look to read two files. These must be output files from *FFGWake* for full scale data and wind tunnel data renamed 'ffg_ship' and 'ffg_wind' respectively .

The program reads the data from these two files, processes the data, then writes a file called 'comp.vec' which contains the full scale data together with the data which has been interpolated from the wind tunnel results This data file is a 'Comparison' data file as described in Section 2.1.2.

[1] The pressure probes used for the wind tunnel data collection were calibrated only up to 40° and this limits the collection of reliable data, especially in the region of the flight deck where the presence of vortices creates regions of flow that are often at angles greater than this.

APPENDIX B - Sample Output From *ffgView*



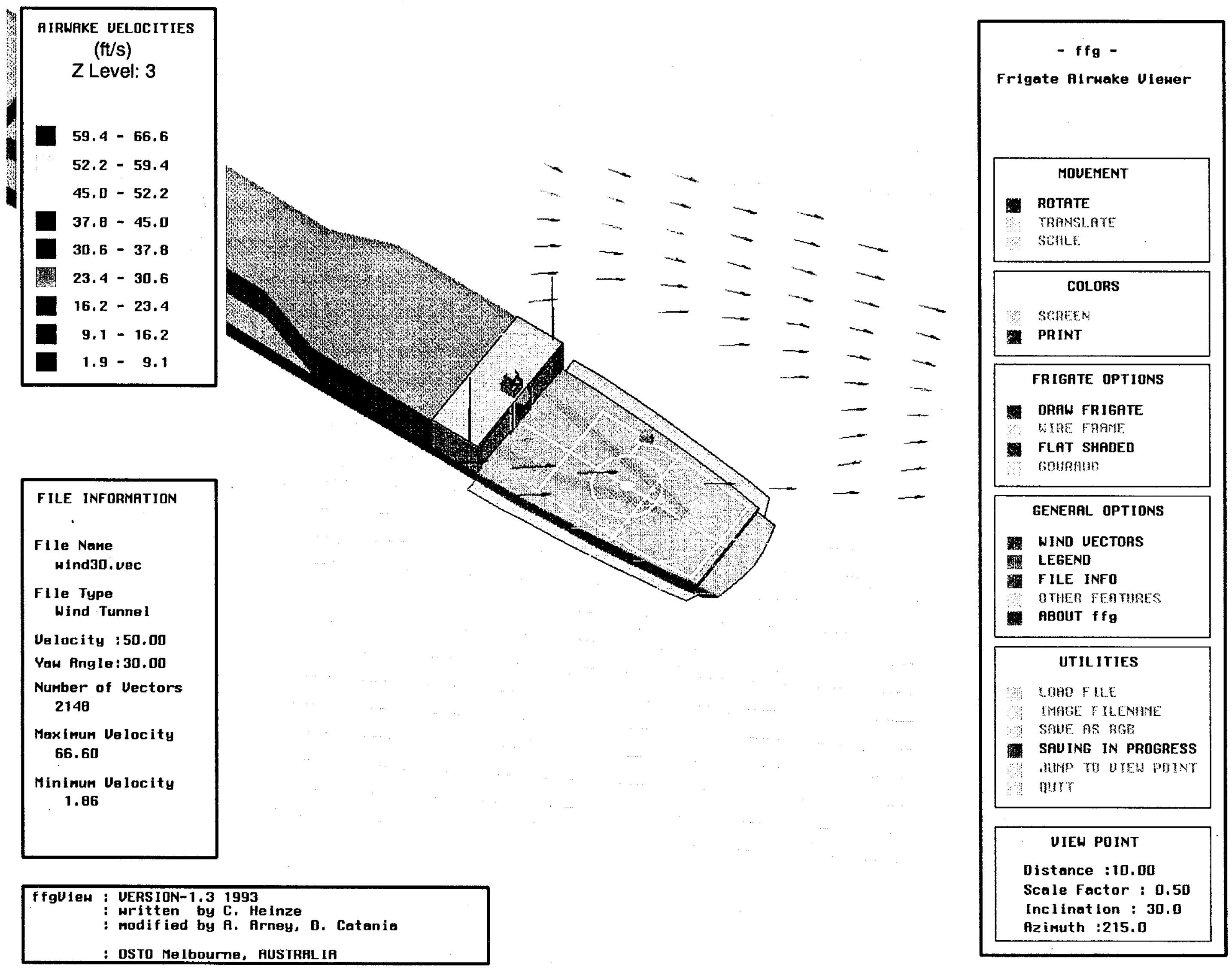
```
AIRWAKE VELOCITIES
(ft/s)

■ 59.5 - 66.5
  52.6 - 59.5
  45.7 - 52.6
■ 38.7 - 45.7
■ 31.8 - 38.7
▨ 24.9 - 31.8
■ 18.0 - 24.9
■ 11.0 - 18.0
■  4.1 - 11.0
```

```
FILE INFORMATION

File Name
 ship30_35kn.vec
File Type
 Ship Trial
Velocity :56.01
Yaw Angle:-33.67
Number of Vectors
 39
Maximum Velocity
 66.47
Minimum Velocity
 4.09
```

```
ffgView : VERSION-1.3 1993
        : written  by C. Heinze
        : modified by A. Arney, D. Catania

        : DSTO Melbourne, AUSTRALIA
```

```
- ffg -
Frigate Airwake Viewer

MOVEMENT
■ ROTATE
▨ TRANSLATE
▨ SCALE

COLORS
▨ SCREEN
■ PRINT

FRIGATE OPTIONS
■ DRAW FRIGATE
▨ WIRE FRAME
■ FLAT SHADED
▨ GOURAUD

GENERAL OPTIONS
■ WIND VECTORS
■ LEGEND
■ FILE INFO
▨ OTHER FEATURES
■ ABOUT ffg

UTILITIES
▨ LOAD FILE
▨ IMAGE FILENAME
▨ SAVE AS RGB
■ SAVING IN PROGRESS
▨ JUMP TO VIEW POINT
▨ QUIT

VIEW POINT
Distance :10.00
Scale Factor : 1.20
Inclination : 30.0
Azimuth :215.0
```

Figure B1. Relative Wind 35 kn From 30^o to Starboard - Full Scale Test



AIRWAKE VELOCITIES
(ft/s)
Z Level: 3

■	59.4 - 66.6
□	52.2 - 59.4
□	45.0 - 52.2
■	37.8 - 45.0
■	30.6 - 37.8
▨	23.4 - 30.6
■	16.2 - 23.4
■	9.1 - 16.2
■	1.9 - 9.1

FILE INFORMATION

File Name
wind30.vec
File Type
Wind Tunnel
Velocity :50.00
Yaw Angle:30.00
Number of Vectors
2148
Maximum Velocity
66.60
Minimum Velocity
1.86

```
ffgView : VERSION-1.3 1993
        : written  by C. Heinze
        : modified by A. Arney, D. Catenia

        : DSTO Melbourne, AUSTRALIA
```

- ffg -
Frigate Airwake Viewer

MOVEMENT
■ ROTATE
▨ TRANSLATE
▨ SCALE

COLORS
▨ SCREEN
■ PRINT

FRIGATE OPTIONS
■ DRAW FRIGATE
▨ WIRE FRAME
■ FLAT SHADED
▨ GOURAUD

GENERAL OPTIONS
■ WIND VECTORS
■ LEGEND
■ FILE INFO
▨ OTHER FEATURES
■ ABOUT ffg

UTILITIES
▨ LOAD FILE
▨ IMAGE FILENAME
▨ SAVE AS RGB
■ SAVING IN PROGRESS
▨ JUMP TO VIEW POINT
▨ QUIT

VIEW POINT
Distance :10.00
Scale Factor : 0.50
Inclination : 30.0
Azimuth :215.0

Figure B2. Relative Wind 30°to Port - Wind Tunnel Test (Single Horizontal Plane)

APPENDIX C - Axes Systems Used by *ffgView*

There are a total of three axes systems used in the generation and displaying of data for and by *ffgView*. The axes system used by the program for all of its drawing and rotations/translations is shown in Fig. 3. It is necessary that the velocities provided to *ffgView* correspond to this axes system. Figs C1 and C2 below show the axes systems for the wind tunnel tests and for the full scale trials respectively. Note that the axes system used for the wind tunnel data collection has the xy plane parallel to the flight deck with the origin at the centre of the hangar wall at a height above the flight deck equivalent to three feet full scale. A program *FFGWake*, described in Appendix A, has been written to perform the conversions from the axes sets below to that shown in Fig. 3.

A fundamental difference between the two types of collected data exists. The axes system for the full scale data is aligned with the ship whereas the axis system for the wind tunnel data is aligned with the wind and so corrections for the yaw angle must be included.

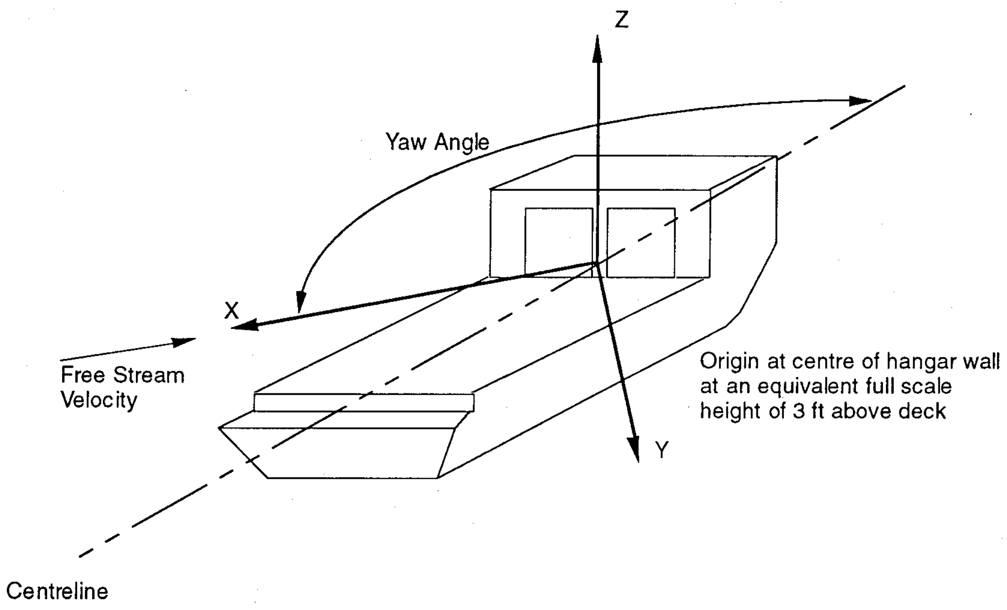

Z
Yaw Angle
X
Free Stream Velocity
Origin at centre of hangar wall at an equivalent full scale height of 3 ft above deck
Y
Centreline

Figure C1. Axes System for Wind Tunnel Data Collection

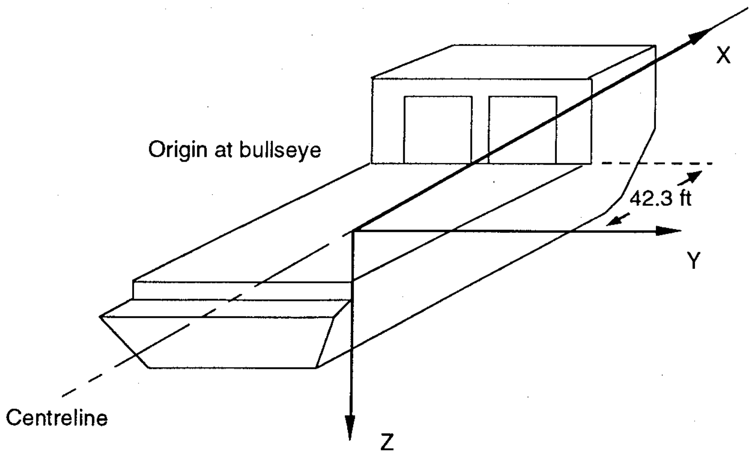


Figure C2. Axes System for Full Scale Data Collection

FFG-7 Class Frigate Airwake Viewer

C. A. Heinze and A. M. Arney

DSTO-TR-0048

DISTRIBUTION

AUSTRALIA

DEFENCE ORGANISATION

Defence Science and Technology Organisation

Chief Defence Scientist }
FAS Science Policy } shared copy
AS Science Corporate Management }
Counsellor Defence Science, London (Doc Data Sheet only)
Counsellor Defence Science, Washington (Doc Data Sheet only)
Senior Defence Scientific Adviser (Doc Data Sheet only)
Scientific Advisor Policy and Command (Doc Data Sheet only)
Navy Scientific Adviser
Scientific Adviser - Army (Doc Data Sheet only)
Air Force Scientific Adviser (Doc Data Sheet only)
Director Trials

Aeronautical and Maritime Research Laboratory
Director
Library Fishermens Bend
Library Maribyrnong
Chief Air Operations Division
Chief Airframes and Engines Division
Research Leader AEW&C
Research Leader Aircraft Performance
Research Leader Flight Systems
Research Leader Human Factors
Head Helicopter Operations (6 copies)
Authors: C.A. Heinze (2 copies)
A.M. Arney (2 copies)
J.F. Harvey
N. Matheson
B. Phelps

Electronics and Surveillance Research Laboratory
Director
Main Library - DSTO Salisbury

Defence Central

OIC TRS, Defence Central Library
Document Exchange Centre, DSTIC (8 copies)
Defence Intelligence Organisation
Library, Defence Signals Directorate (Doc Data Sheet Only)

Navy

Aircraft Maintenance and Flight Trials Unit
Director Aircraft Engineering - Navy
Director of Aviation Projects - Navy
Director of Naval Architecture
Director of Occupational Health and Saftey and Naval Medicine
CO HMAS Albatross
CO HMAS Cerberus
Office of Naval Attache, Washington
RAN Tactical School, Library
Superintendent, Naval Aircraft Logistics

Army

CO 5 RGT, Townsville
Engineering Development Establishment, Library
School of Army Aviation, Oakey

Air Force

Aircraft Research and Development Unit
    Tech Reports,CO Engineering Squadron,ARDU
Director of Flying Safety - Air Force
AHQ CSPT Amberley
RAAF Base East Sale
RAAF Base Glenbrook
RAAF Base Pearce
RAAF Base Richmond
RAAF Base Tindal
RAAF Base Williamtown
OIC ATF, ATS, RAAFSTT, WAGGA (2 copies)

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy
    Library
    Head of Aerospace and Mechanical Engineering
Melbourne
    Engineering Library
Monash
    Hargrave Library
Newcastle
    Library
    Institute of Aviation
NSW
    Physical Sciences Library
RMIT
    Library
    Aerospace Engineering
Sydney
    Engineering Library

OTHER GOVERNMENT DEPARTMENTS AND AGENCIES
AGPS
Bureau of Air Safety Investigations
Civil Aviation Authority

OTHER ORGANISATIONS
NASA (Canberra)

CANADA

Defence Research Establishment, Atlantic
J.L. Colwell
National Research Council, Ottawa
M. Sinclaire (Canadian NL TTCP HTP-6)
S. Zan
University of Toronto
L.D. Reid
Bombardier Inc., Canadair, Montreal
B.I.K. Ferrier

UNITED KINGDOM

Defence Research Agency, Bedford
G.D. Padfield
S. Tate
B. Lumsden
Defence Research Agency, Farnborough
A.F. Jones (UKNL TTCP HTP-6)

UNITED STATES

US Army Aeroflightdynamics Directorate, Ames Research Center
W.G. Bousman (USNL TTCP HTP-6)
Naval Air Warfare Center, Aircraft Division, Warminster
J.W. Clark, Jr (TTCP HTP-6)
J. Funk
Naval Air Warfare Center, Aircraft Division, Patuxent River
D. Carico
J. McCrillis
L. Trick
Naval Air Warfare Center, Aircraft Division, Lakehurst
H. Fluk
Georgia Institute of Technology, Atlanta
D. Mavris

SPARES (8 COPIES)

TOTAL (100 COPIES)

AL 149 DEPARTMENT OF DEFENCE

DOCUMENT CONTROL DATA

PAGE CLASSIFICATION
UNCLASSIFIED

PRIVACY MARKING

1a. AR NUMBER AR-008-350	1b. ESTABLISHMENT NUMBER DSTO-TR-0048	2. DOCUMENT DATE AUGUST 1994	3. TASK NUMBER NAV 92/019

4. TITLE

FFG-7 CLASS FRIGATE AIRWAKE VIEWER

5. SECURITY CLASSIFICATION (PLACE APPROPRIATE CLASSIFICATION IN BOX(S) IE. SECRET (S), CONF. (C) RESTRICTED (R), LIMITED (L), UNCLASSIFIED (U)).

DOCUMENT	TITLE	ABSTRACT
U	U	U

6. NO. PAGES
36

7. NO. REFS.
8

8. AUTHOR(S)
C.A. HEINZE
A.M. ARNEY

9. DOWNGRADING/DELIMITING INSTRUCTIONS

Not applicable.

10. CORPORATE AUTHOR AND ADDRESS

AERONAUTICAL AND MARITIME RESEARCH LABORATORY
AIR OPERATIONS DIVISION
GPO BOX 4331
MELBOURNE VIC 3001 AUSTRALIA

11. OFFICE/POSITION RESPONSIBLE FOR:

SPONSOR NAVY

SECURITY -

DOWNGRADING -

APPROVAL CAOD

12. SECONDARY DISTRIBUTION (OF THIS DOCUMENT)

Approved for public release.

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DSTIC, ADMINISTRATIVE SERVICES BRANCH, DEPARTMENT OF DEFENCE, ANZAC PARK WEST OFFICES, ACT 2601

13a. THIS DOCUMENT MAY BE ANNOUNCED IN CATALOGUES AND AWARENESS SERVICES AVAILABLE TO ....

No limitations.

14. DESCRIPTORS
Visual display units
Oliver Hazard Perry class frigates
Aerodynamic wakes
Data acquisition
Wind tunnel tests
Sea testing

15. DISCAT SUBJECT CATEGORIES
1310
2004

16. *ABSTRACT*
This document presents the operation of the graphical display program *ffgView*, which has been written to run on the Silicon Graphics family of computers. The program allows the airwake around an 'Adelaide' class FFG-7 frigate to be displayed using data generated from wind tunnel testing or from actual ship trials. Details of the required file formats and of the structure of the source code are included here, as are detailed operating instructions.

PAGE CLASSIFICATION
UNCLASSIFIED

PRIVACY MARKING

THIS PAGE IS TO BE USED TO RECORD INFORMATION WHICH IS REQUIRED BY THE ESTABLISHMENT FOR ITS OWN USE BUT WHICH WILL NOT BE ADDED TO THE DISTIS DATA UNLESS SPECIFICALLY REQUESTED.

16. ABSTRACT (CONT).

17. IMPRINT

AERONAUTICAL AND MARITIME RESEARCH LABORATORY, MELBOURNE

18. DOCUMENT SERIES AND NUMBER DSTO Technical Report 0048	19. WA NUMBER 51526J	20. TYPE OF REPORT AND PERIOD COVERED

21. COMPUTER PROGRAMS USED

ffgView
FFGWake
readcol
windcomp
Mac Trans

22. ESTABLISHMENT FILE REF.(S)

M1/9/14

23. ADDITIONAL INFORMATION (AS REQUIRED)