# Chapter 1

# Introduction

## 1.1 What is ICAO?

ICAO is the abbreviation of the *International Civil Aviation Organization*. If you are a pilot, that won't be new to you. And the ICAO hasn't got too much to do with this program. So why is it called ICAO?

Well, work on this application started in December 1992. I got interested in the mathematics involved with map projections. A few books from our library provided me with the knowledge to draw a part of the earth onto the flat screen of a computer, using the conical Lambert projection with two standard parallels. It was interesting, nice to watch and useless.

It was about the same time that I got Linux, which I liked better the longer I used it. Not only could I work under the same environment I used at the university; it also seemed to me much more stable and a lot faster than most other operating systems for IBM clones. Then I turned to X11 programming. Equipped with some literature about X, I began rewriting the map projection program, probably just because I couldn't think of anything more intelligent.

As an empty map is rather boring, the next step was to draw some objects on it. The only map handy at that time was - guess what - the "Aeronautical Chart ICAO 1:500000", simply refered to in Germany as "ICAO Karte". The first executable that could draw an airfield symbol was named *icao*. That's what it is still called today, and I guess unless a superb other name comes to my mind that won't change.

Naturally, Linux had become the development platform for ICAO. To verify portability, I compiled it on a HP Apollo Series 700 workstation from time to time, which worked fine.

Hardcoding the objects to be displayed into the program was OK for two airports, but impossible for the 600-odd objects that were in the world file in January 1993. That forced me to have a closer look at YACC and LEX (or rather BISON and FLEX, the GNU equivalents of the well-known UNIX tools), which I used to create the scanner and the parser that read the world file.

Most data currently contained in the world file comes from the official ICAO map and from the German AIP (Aeronautical Information Publication). The coordinates of civil airports and radio navigation facilities should be precise. Most other coordinates where read from the map with a digitizer, so that points of rivers, CTRs and towns are not always quite as precise.

At first, the program didn't have many functions: it could display the map, you could select new origins of the map to be displayed, zoom in and out as well as hide and show certain types of objects. You could also measure distances and true tracks from one point to another, and you could display info boxes for most objects. Many more things were planned for both soaring and motorized flying, but that would have to wait a bit - I was still looking for the Release-5-edition of the O'Reilly X Toolkit Intrinsics Programming Manual, which would (according to a lokal book shop) be available in April 1993. Till then, I improved the map display and added some more objects to the world file.

Well, the O'Reilly book finally arrived, and at about the same time Metrolink announced Motif for Linux. Not having seen much of Motif at that time, I just guessed that it could be useful for ICAO, so I ordered it. The result is what you are currently looking at.

If you have any suggestions about useful functions for the program, feel free to tell me. If anybody would like to contribute data to the world file - even better, I really can't manage it all by myself. If someone is willing to port ICAO to other (non-UNIX) platforms - great, contact me! Though I like UNIX, I'm aware of how few pilots have access to a UNIX machine.

## 1.2 About the Author

Martin Pauly was born in 1971 in Moenchengladbach, Germany. He is currently a student of computer science in Aachen. He used to live in Krefeld for a couple of years, next to two airline captains who would get him more than just interested in planes. At the age of 14, he started soaring in a club at the airfield in Grefrath (EDLF). If you ever see an ASW-15B with the registration number D-2310, chances are good that he might be sitting inside it.


As far as computers are concerned, Martin started hacking on a Commodore VIC-20 in the early 80s. An ATARI ST followed later, finally an IBM clone. Fascinated by Linux and the large amount of other public domain software available for UNIX systems, he'd like to offer ICAO to anyone who might find it useful.

You can contact Martin by E-mail as well as snailmail:
Martin Pauly, Ruetscherstr. 175/910, 52072 Aachen, Federal Republic of Germany
Internet: pauly@pool.informatik.rwth-aachen.de

# Chapter 2

# Implementation notes

The "original" ICAO application (and by the time this document is written the only one) is based on Linux and requires X11R5 and Motif 1.1 or 1.2. The distribution contains an Imakefile and should be working on any UNIX/X11R5/Motif environment.

Early versions of ICAO were succesfully compiled on Hewlett-Packard Series 700 workstations. However, I had some trouble with the Motif version. Most of the problems are probably caused by the mixture of Motif files from different versions of Motif that were available to me so far. One thing I did experience, too, is that compiling with *gcc* seems easier than using *cc*.

There is also a set of Linux binaries linked for both shared and static Motif libraries, meaning that Motif does not have to be installed on your Linux machine in order to use ICAO.

In October 1993, Oliver Meyer and Stefan Leuker announced that they started porting ICAO to NeXTstep 486. Any porting attempts to most other platforms are very much appreciated, as long as they respect the ICAO licence which you can find at the end of this document.

## 2.1 History

**December 1992:** Work on ICAO begins. The roots come from an MS-DOS program that displays part of the globe.

**January 1993:** Version 0.10 of ICAO is running under X11/Linux. Most object types have been implemented.

**February 1993:** Version 0.20 of ICAO includes PostScript output. The master world file contains nearly 700 objects by now.

**March 1993:** Version 0.50 is a first attempt to use the X-Toolkit for ICAO. Without any real programming manuals, writing a map widget was a bit difficult so that version did not contain any map display. For the first time, the number of objects contained in the world file exceeded 1000.

**Summer 1993:** Like most glider pilots, Martin preferred spending his spare time at the airfield, not in front of his computer. No work on ICAO is done.

**Oktober 1993:** In version 0.60a, a Motif interface serves as a front-end for the features that version 0.50 already had, combined with the map display of the older versions. Oliver Meyer and Stefan Leuker start working on a NeXTstep port.

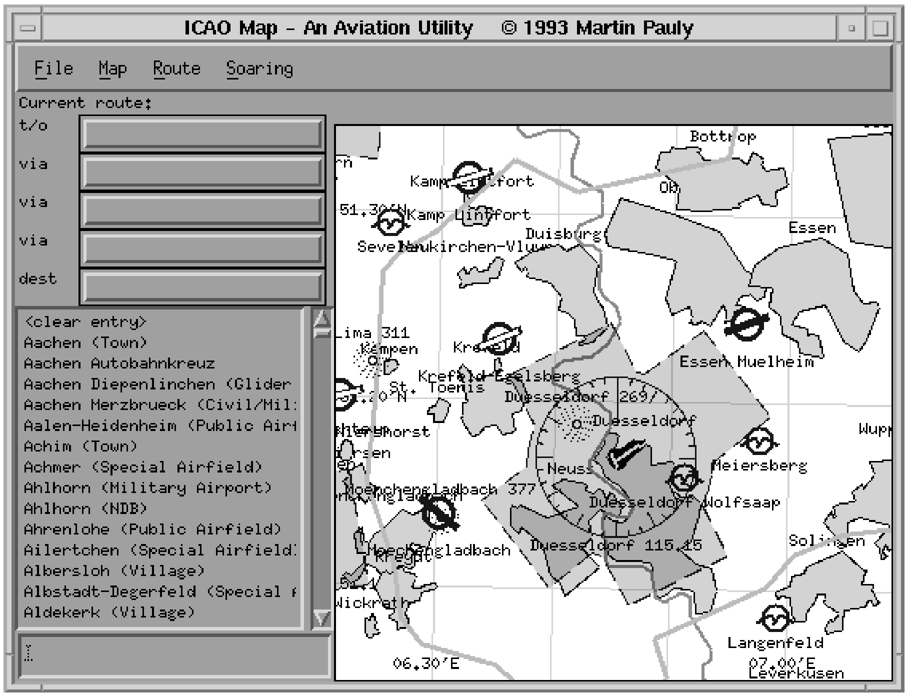
**November 1993:** Almost one year after work on ICAO began, the UNIX/X11 version is uplaoded to a number of FTP servers. Current version number is 0.60b.

# Chapter 3

# Running ICAO

To start ICAO, you need the executable *icao* and a *world file* that represents the database containing all the objects ICAO uses to perform its calculations and to draw maps. If the *world file* is called *world,* start ICAO from your X11 screen using by

icao world &




The main window will appear, consisting of

- the menu bar that works no different that any other Motif menu
- the map window on the right hand side
- the current route that is made of a take-off location and a destination, as well as up to three in-between positions
- the object browser that lists all objects that are contained in the world file by name
- the quick-search entry field in the bottom left corner of the ICAO window. As the name implies, it can be used to quickly find an object name in the browser.

If you don't specify a world file, ICAO will attempt to read it from standard input. Consequently, you can also start the application using something like

icao < world &

This allows you to have the world-file compressed on your file system and pipe the output of *uncompress* to ICAO.

## 3.1 The Map Window

Within the map window, mouse buttons perform the following functions:

**left:** Makes the current cursor location the new origin of the map. The map origin is the point in the middle of the map.

**middle:** This button is reserved for future use. It will be needed for the Drag&Drop feature supported by Motif 1.2.

**right:** Display additional information (if available) for an object close to the cursor. If no object is close enough, ICAO will tell you what airspace you are in. The respective information is displayed in a small popup window that automatically disappears when you release the mouse button.

The right mouse button is also used to measure distances and true tracks. Go to the starting point of your line, hold a shift key on your keyboard, press the right mouse button and hold it. Then move to whereever you want to go. The top line will display the exact distance and track. Releasing the mouse button will quit this mode. Please note that the "rubber band" you are moving snaps to objects when you get close enough. If you don't like the snap function, release the shift key but keep holding down the right mouse button.

## 3.2 The Object Browser

The object browser contains the names of all objects that you could fly to (not necessarily land at). That includes airports, waypoints, towns and lakes, but not rivers and roads. The latter objects are to long to have something that can be called a center or a reference point.

When it comes to waypoints, the type of the object is sometimes part of the name. You might have the "Kraftwerk Weisweiler" in your world file. When you look for it in the object browser, you might also try to find the name "Weisweiler Kraftwerk". Both names are listed in the browser to make finding objects easier.

Scrolling through the list is simple. When you double click on a list item, that item will be brought into the center of the map. It may be nice for some people to scroll through the browser and guess where in the world certain objects are - the answer is only a mouseclick away.

## 3.3 The Quick-Search Field

If you know the name of an object, you can quickly select it in the browser by typing its name (usually, the first couple of letters will be sufficient) into the quick search field that is located at the bottom left corner of ICAO's main window. Whenever you select it with the mouse, it is cleared. Typing in letters will make the first object name selected that matches the letters you have typed so far.

## 3.4 The Route Display

You can select a route by specifying a take-off location, a destination and up to three points to pass during your flight. The names of the respective waypoints are displayed in the top left corner of the ICAO main window. Please note that the way the route panel is used will be changed as soon as I figure out how the Drag&Drop system works.

Anyway, at the time being you choose a waypoint by clicking into the large button right of the t/o, via or dest label. The respective label will be bordered to indicate that the field is now "hot". Clicking on the button another time will deselect the field.

Once you have selected a waypoint field, the next object you select from the browser will be selected as a waypoint for your current route. To clear a waypoint field, select this field first and then click on the *<clear entry>* item at the very top of the object list.

Alternatively, you can select objects directly from the map. Whenever a route field is selected, pressing the left mouse button when the pointer is close to an object of the map will select this object as a waypoint. Airspace objects, rivers and roads can not be selected that way.

A third way to select objects is to use the quick search field. Once the desired object is highlighted in the object browser, press the Return key when the quick search field is focused to select that object as a waypoint.

# Chapter 4
# The Menu Bar

## 4.1 The File Menu

### 4.1.1 Load World File...

This function has not yet been implemented. To use a world file, you have to start ICAO and specify the name of the world file as a program argument.

### 4.1.2 Information...

This displays a small dialog box that shows how many objects ICAO currently finds in its world file.

### 4.1.3 Quit

Quits ICAO immediately.

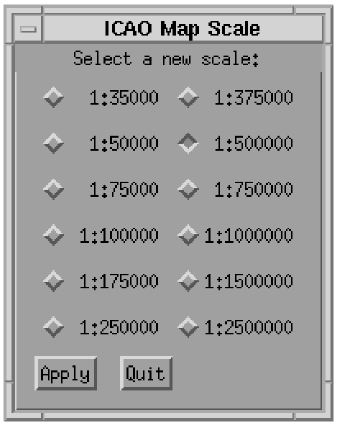
## 4.2 The Map Menu

### 4.2.1 Change Scale...

You should be familiar with the term *scale,* as far as maps are concerned. Using X11, the X server should know the dimensions of your physical display. Assuming that your X setup is correct, you should get a map onto your screen that shows the same area as a printed map with the same scale and size.

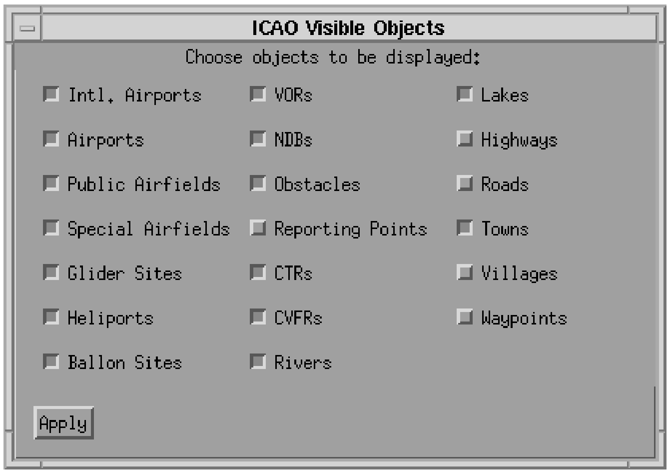
As you see in the set scale dialog, you can select scales from 1:35,000 up to 1:2,500,000. Smaller scales (= larger figure behind the colon) are probably not too useful on the screen. Still, using the *scale* directive in your world file, you may select any scale you like. When it comes to printing large routes, a scale is automatically chosen by ICAO without the limitation to the scales offered in the set scale dialog.

The *apply* button will use your selection and pop down the dialog box. Clicking on the *close* button only pops down the dialog without changing the current scale.




## 4.2.2 Select Visible Objects...

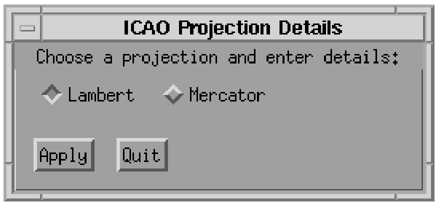
For different purposes, the user may want different type of objects to be visible on the screen. The IFR pilot won't be too interested in highways and small VFR-only airfields, whereas a glider pilot will probably not take much advantage of radio navigation facilities. Apart from that, the smaller your scale is, the more objects have to be drawn on the map - it's a matter of course that your screen will be crowded, and the time it takes to redraw the map increases.




Furtunately, you can choose what object types you would like to see on the map. A *visible objects* dialog box is used for that purpose. Small toggle buttons can be clicked on and off to enable or disable certain object types. The result is not only visible on the screen, it also selects object for printing.

## 4.2.3 Select Map Projection...

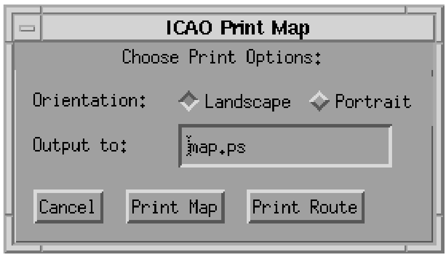
Without going into detail at this point: ICAO supports two different algorithms to project the three dimensional surface of the earth onto a two dimensional map. Depending on what characteristics you require, different projection algorithms can be more or less useful. In aviation, the conical Lambert projection and cylindrical Mercator projection serve most needs quite well as long as you do not approach the poles (which you should not try using ICAO - hey, it’s damn cold there anyway).




Using the *Select Map Projection* menu item, a small dialog box appears that gives you the choice of either Lambert or Mercator projection. Choose whatever you like and confirm by clicking on the *apply* button. Clicking on the *quit* button will close the dialog box without noticing your changes.

## 4.2.4 Print Current Map...

ICAO can create PostScript files that can be interpreted by GhostScript or similar programs. If you are the lucky owner of a PostScript printer, you can of course send the output file immediately to the printer.




When ICAO prints a map, it doesn’t bother what is currently displayed on the screen. It does, however, check what scale and what origin (center) has been selected. It also includes only those objects that are currently marked in the *Select Visible Objects* dialog. After these considerations, ICAO produces a 24.7cm times 17.6cm sized map that is drawn in either landscape or portrait orientation, depending on what you select in the print dialog. The PostScript code that is generated is sent to a file whose name can be entered in the dialog. It defaults to *map.ps.*

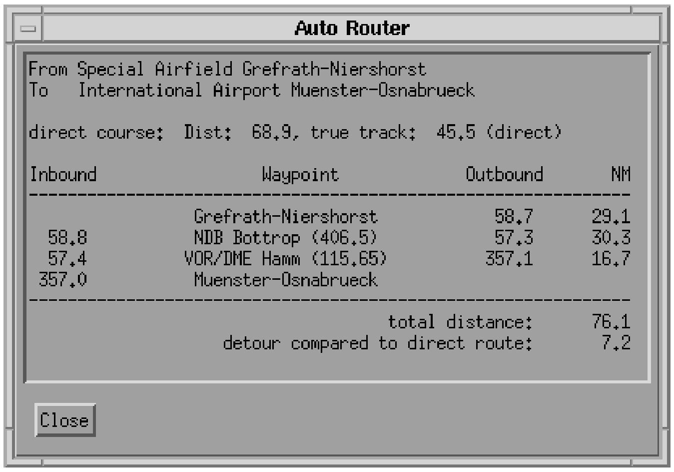
When you are planning a flight, you may want a printout of the complete route, covering the take-off location and destination as well as all the waypoints in between. In that case, click on the *Print Route* button in the print dialog box. ICAO then selects a scale that makes your route fit on a single sheet of paper. It also determines whether landscape or portrait orientation is preferable. Last but not least, it draws a thin line indicating your selected route. If the autorouter (s.b.) was used, its output is drawn to the map, replacing the direct course.

## 4.3 The Route Menu

### 4.3.1 Autorouter

Provided that you have selected a take-off location and a destination (and optionally via waypoints), the autorouter can find a short route using radio navigation facilities. It respects the different ranges of the different VORs and NDBs. It also knows that you can fly *to* and *from* radials of VORs, but only *towards* NDB stations.

The autorouter divides your flight into sections that usually go from one radio beacon to another one. It then shows a window listing all these sections with their respective distances and tracks. Finally, it compares the lotal length of the suggested route with the length of the direct connection between start and destination. If your trip is not very short, ICAO usually finds a route that has only a rather small detour.




The autorouter will fail to work if you did not specify a start and a destination point or if the distance between beacons is too large. It does not always find the *shortest* possible route.

### 4.3.2 Draw Route, Undraw Route

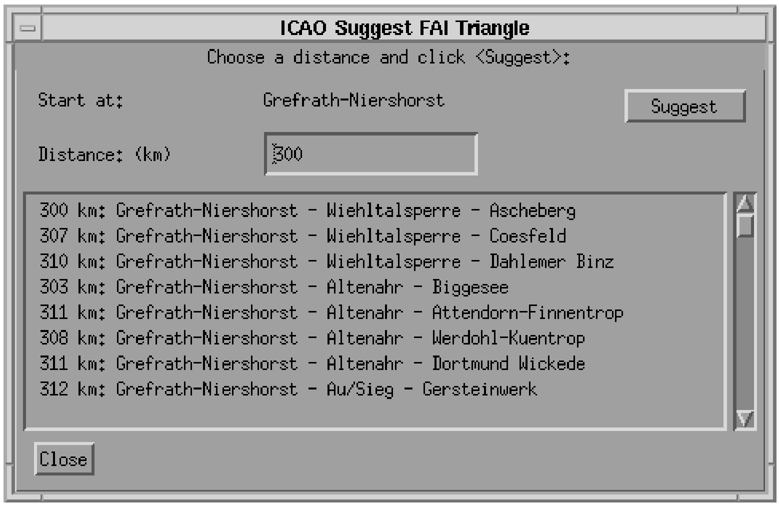
If you like, ICAO can draw the currently selected route as a thin black line on your display. Simply select the *Draw Route* menu item. To remove the lines from your screen, choose the *Undraw Route* command from the menu.

## 4.4 The Soaring Menu

### 4.4.1 Suggest Triangle

This function should only be interesting for glider pilots. When you do a cross country flight, usually triangles that respect the FAI rules result in the best score/distance ratio. The *Suggest Triangle* function was created to help you finding possible triangles that have a certain length and match the FAI requirements: the shortest side of the triangle must be at least 28% of the total distance (or 25% if the total distance is more than 500km), whereas the longest side may not exceed 45%.

To use this feature, select a take-off location in the route panel first. Then call the *Suggest Triangle* menu item. A dialog box will appear, allowing you to enter the desired total length in km. Click on the *Suggest*-button when you have entered a range. A browser will appear in the dialog, listing all the routes that heve been found.




Double clicking on the suggested triangles will select the current waypoints in ICAO's route panel and draw the route on the map.

# Chapter 5

# The World File

All the information that is used to display the map comes from an ASCII file called the *world file.* It contains object definitions of most common aviation-related objects such as airports and radio navigation facilities, but also of other ground objects that will usually be found on a pilot's map. A world file is read each time you start the program.

Being an ASCII file, the world file can easily be extended by anyone who'd like to contribute data to the system, so that maybe one day the term *world* file will be justified. You are strongly encouraged to provide information about areas that you know - the world is by far too big to be digitized by a single person!

This chapter tries to explain what the world-file looks like. It uses a meta-language called *Backus-Naur-Form.* Very simply, it is this:

- <nonterminal symbol> ::= A | B | C
  Assignment: the nonterminal symbol (i.e. an identifier) is either A, B or C.
- [symbol]
  Refers to either what symbol is or to nothing, i.e. you may either specify symbol or not.
- {symbol}
  In this case, symbol may be used any number of times in a row, including zero times.

Parentheses can be used to specify precedences.

## 5.1 Tokens and Types

A scanner usually has a list of tokens and data types that can be recongnized. They are the terminal symbols of the world file language:

| | | |
|---|---|---|
| intlairport | airport | airport_civ_mil |
| airport_mil | airfield | special_airfield |
| heliport | heliport_amb | glider_site |
| hang_glider_site | parachute_jumping_site | free_ballon_site |
| vor | vor_dme | vortac |
| tacan | ndb | marker_beacon |
| basic_radio_facility | obstruction | group_obstruction |
| fired_obstruction | fired_group_obstruction | waypoint |
| isogone | river | lake |
| highway | road | village |
| town | ctr | cvfr |
| reporting_point | at | @ |
| elev | range | frequency |
| alias | runway | gras |
| asphalt | asph | concrete |
| conc | out | scale |

```
origin          parallels          show_map
digitize        calibrate          zeromeridian
```

The following data types can be processed:

```
<float> ::= {<digit>} [.] {<digit>}
<string> ::= <alpha> {<alpha>|<digit>|.|_|-}
<latitude> ::= <digit><digit>.<digit><digit>'<digit><digit>"(N|S)
<longitude> ::= <digit><digit>[<digit>].<digit><digit>'<digit><digit>"(E|W)
```

Two nonterminal symbols still need to be defined:

```
 <alpha> ::= a | b | ... | z | A | B | ... | Z
 <digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Comments can be put anywhere in the world-file using the C-style:

/* this is a comment */

A comment can be serveral lines long. Comments can not, however, be nested; therfore, the following construct would cause an error:

/* anything /*comment*/ more of anything */

Whitespace characters, i.e. spaces, tabs and newlines are ignored by the scanner as long as they don't occur inside a token. The scanner is not case-sensitive, meaning that it doesn't matter whether you write *airport, Airport* or *AIRPORT.* The only exception are strings, for they are usually used for names and IDs.

## 5.2 Parsing the World File

With these tokens and types in mind, we can look at what the parser does when it reads through the world file:

```
<world-file> ::= {<line>}

<line> ::= [<loc> [<elev>]] [<object> {<quality>}]
         | <command>

<elev> ::= elev <float>

<loc> ::= at <location>
        | @ <location>

<location> ::= <latitude> <longitude>

<quality> ::= frequency <float>
            | runway <float> <float> <runwaytype>
            | alias <string>
            | range <float>

<runwaytype> ::= asphalt | asph
               | concrete | conc
               | gras
```

```
<object> ::= intlairport <string> {<location>}
           | airport <string> {<location>}
           | airport_mil <string>
           | airport_civ_mil <string>
           | airfield <string>
           | special_airfield <string>
           | heliport <string>
           | heliport_amb <string>
           | glider_site <string>
           | parachute_jumping_site <string>
           | free_ballon_site <string>
           | vor <string>
           | vor_dme <string>
           | vortac <string>
           | tacan <string>
           | ndb <string>
           | marker_beacon <string>
           | basic_radio_facility <string>
           | obstruction <string>
           | group_obstruction <string>
           | fired_obstruction <string>
           | fired_group_obstruction <string>
           | reporting_point <string>
           | river [<string>] {<location>}
           | lake [<string>] {<location>}
           | highway [<string>] {<location>}
           | road {<location>}
           | town [<string>] {<location>}
           | isogone <float> {<location>}
           | village <string>
           | waypoint <string>
           | ctr <string> {<location>}
           | cvfr <string> {<location>}

<command> ::= out <string>
            | scale 1:<float>
            | origin <location>
            | parallels <latitude> <latitude>
            | show_map
            | digitizer
            | calibrate <location> <location>
            | zeromeridian <longitude>
```

## 5.3 Parse Errors

If the parser thinks that something is wrong in the world file it will output a *parse error* message and the line number that was processed. It will not tell you what precisely is wrong, but usually an error will be easy to find. Typical mistakes are prohibited characters within a string (like blanks - use underscores (_) instead), mismatched quote marks in locations etc. Look at similar object definitions in the original world file and compare it with the line that's causing trouble.

# Chapter 6

# Geodetical Aspects of ICAO

This chapter has not been written yet. It will cover the different qualities of map projections and tell you how ICAO calculates distances and tracks.

# Chapter 7

# Licence Information

ICAO is *not* Public Domain or Shareware. It is copyrighted software that is simply free of charge, whether it is used for private use or for business purposes.

You are encouraged to use it, distribute it and even change the source code if you think you can make better use of it that way. Distributors may ask for a fee, but it must be obvious that this fee is for the media, shipping and handling only.

The world file contains nothing but names and coordinates. Typing it all in was a lot of work, but locations of objects can hardly be something that's worth a copyright. Therefore, the world file is not copyrighted but public domain.

Martin Pauly welcomes new regions added to the world file. Portings to other operating systems are also appreciated, except for any MicroSoft operating system or user interface other than MS-DOS 1.0, i.e. porting ICAO to the latter platforms is prohibited.

ICAO is distributed without any warrenty, without even the implied warrenty of merchantability or fitness for a particular purpose. The only thing Martin Pauly guarantees is that the program will occupy disk space.

The use of general descriptive names, trade names, etc., in this document, even if the former are not especially identified, is not to be taken as a sign that such names, as understood by the Trade Marks and Merchandise Act, may accordingly be used freely by anyone.

Unless especially mentioned, ICAO in this document refers to Martin Pauly's program and has nothing to do with the International Civil Aviation Organization.

# Chapter 8

# The Future

New features are planned for future versions of ICAO. As a glider pilot, my ideas mainly deal with the *soaring* menu of the program. Things that will definitely be included in the next release of ICAO include:

- an index function for the *suggest triangle* function that depends on the detour a pilot will have to make to avoid restricted airspace
- evaluation of cross country flights according to DMST rules, including flights that were not completed
- wind influence
- enhanced route printing, including text to help pilots with their flight preparation
- digitizer support (as soon as I figure out how that damn serial port can be accessed without getting a "Device or resource busy" error message - any ideas?)
- deviation support for magnetic North reference
- more precise distance and track measuring using an elliptical model of the earth rather than a spherical one.

Any other suggestions from your side? Then don't hesitate to tell me about it! If I like your ideas, I'll see what I can do...

Apart from program extensions, the world file will be extended by me. I plan to have all airports, airfields and glider sites in Germany in it, and more detailed information from the ICAO maps 1:500,000 "Duesseldorf" and "Frankfurt". Buying all the maps for Germany every year would be a bit expensive, but if anyone likes to contribute a map, I'll try to add it to the world file.

You might have the impression now that the world file should only include objects from Germany. This is not the case. However, as creating a detailed world file really takes a lot of time, I will not do anything but Germany myself. A future version of ICAO will include a more complex world file management, but no details have been fixed yet. If you want to create world files for other countries and you need help, just contact me and I'll try to help you.

# Chapter 9

# Known Bugs

There are some odd things about ICAO that I did not fix before releasing version 0.60b.

- When you resize the application's main window, all other windows that belong to ICAO change their size as well.
- Windows can be opened multiple times, which is not useful and can lead to mistakes.
- The *Load World File* menu item doesn't work yet.
- Dialogs don't have a consistent action area (the apply-done-close system found in many X applications)
- Errors in the world file sometimes crash the application.
- Trying to display the polar regions crash ICAO.
- Not all German airfields are contained in the world file at the moment. I am entering them in alphabetical order and only reached 'N' so far.

If you find more things that need to be corrected, please tell me what it is.

# Chapter 10

# Acknowlegdements

I'd like to thank the following people (in arbitrary order):

- Linus Torvalds for creating Linux and donating it to the public
- all the people who were involved in assembling the SLS package
- Michael Heck for his airport directory
- Stefan Leuker and Oliver Meyer for their suggestions and porting efforts
- Frans Goddijn for offering to edit the manual and being so patient with me as far as correspondence is concerned