THE JOURNAL OF NAVIGATION (2002), **55**, 263–275. © The Royal Institute of Navigation
DOI: 10.1017/S0373463302001765 Printed in the United Kingdom

The Rules of Sailing Races for Hand-Held Devices

Kenneth J. Turner and Mark A. Jennings

(*University of Stirling*)

The motivation for this paper was to provide computer support on the rules for sailing races. A brief overview is given of sailing terminology and racing rules. It is argued that a useful program for the racing rules should run on hand-held devices. The program should support an archive of rule scenarios, race training, self-learning of the rules, and analysis of rule disputes. *SailRule* is a freely available program intended to analyse and improve performance in applying the racing rules. The *SailRule* program has been implemented using the SuperWaba programming environment for hand-held devices. The user interface is described for the rules program. An explanation is given of the principles behind formalising and codifying the racing rules so that they can be efficiently implemented. Examples are given of how the program represents and analyses rule scenarios.

KEY WORDS

1. Sailing. 2. Rules. 3. Automation.

1. INTRODUCTION.

1.1. *The Rules of Sailing Races*. The worldwide rules for sailing races are defined by ISAF (the International Sailing Federation, www.sailing.org). The governing body in each country, such as the UK's RYA (Royal Yachting Association, www.rya.org.uk), may also set national rules or define national interpretations of the rules. The rules are intended to ensure fair and safe competition.

The racing rules are usually updated every four years. For 1997–2000, ISAF issued a much-simplified set of rules that are certainly easier to learn. The 2001–2004 rules have continued this simplicity. However, a surprising number of complications can arise in their application. Professional sailors, and those sailing at national levels, are of course expert in the rules. But many sailors, such as those who race casually at their local clubs, know only the major rules. Situations in a race can also change quickly; for example, a change of wind direction can completely alter which boats have priority. Unless the sailor can immediately relate the situation to the rules, it is easy to infringe them.

The rules are written as precisely as natural language (English) will permit. Of necessity they are somewhat legalistic in style. It is not always easy to interpret how the rules should apply. The rules are textual and linear, whereas the real world demands a dynamic and visual understanding of the relationships between boats. To give some idea of the difficulty faced in learning the rules, here is part of the text

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

describing who has right of way on passing a mark (e.g. a course buoy) or an obstruction (e.g. a rock):

> If boats were overlapped before either of them reached the two-length zone and the overlap is broken after one of them has reached it, the boat that was on the outside shall continue to give the other boat room. If the outside boat becomes clear astern or overlapped inside the other boat, she is not entitled to room and shall keep clear.

During a race, a boat may be fouled by another over which it has priority. For example, a boat with right of way may be forced to alter course; the boat in the wrong can recover by taking a penalty. However, a dispute can arise as to which boat had right of way. In such a case a boat may protest, meaning that the issue has to be settled ashore. A formal protest hearing is held to allow the parties to argue why they were in the right. A protest has to be justified with reference to the rules, calling on other sailors as witnesses if necessary. If a protest is upheld, the boat in error is disqualified.

Figure 1 gives a feel for what is involved in the rules. This is a plan view of three boats trying to pass a buoy in the course. They are in danger of collision, so who has right of way? A study of the rules reveals that C has right of way over A and B. However, in turn, A has right of way over B. This situation is analysed and explained in more detail in Section 3.4.

1.2. *Software Support for The Racing Rules.* The nature of the racing rules suggests that some kind of software support may be feasible. The authors set out to explore this hypothesis. The aim was to support the analysis and improvement of performance in understanding and applying the rules. The specific objectives were to support a variety of activities:

(a) It is useful to have a diagram editor for drawing scenarios in which the racing rules have to be applied. Of course, simple pen-and-paper could be used for this. However, computerised support means that the diagrams can be drawn neatly and unambiguously. In a protest hearing, such a diagram could be a convenient way of recording how the involved parties saw the situation.
(b) Computer support also makes it possible to save diagrams for later retrieval and analysis. For example, an archive of common situations can be prepared and used in coaching and learning.
(c) It is useful to set up 'what if' scenarios, either to check an understanding of the rules or to investigate how changes in a scenario might affect which rules apply.
(d) If a dispute arises, computer support can suggest which rules apply and therefore which boats have right of way. Much of this needs routine book-keeping that can be conveniently automated. Of course it still requires human expertise to decide on a protest, but at least the factual aspects can be supported automatically.

It would have been possible to write a conventional program to achieve these objectives. However, a program running on a PC would really only permit offline analysis. What is required is something much more portable. An attractive possibility was an implementation of the racing rules on a hand-held device such as a PDA (Portable Digital Assistant, also called a palm-top). Current PDAs are much more than simple devices for recording appointments and telephone numbers. In fact they are small computers and programmable in their own right. The authors have

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

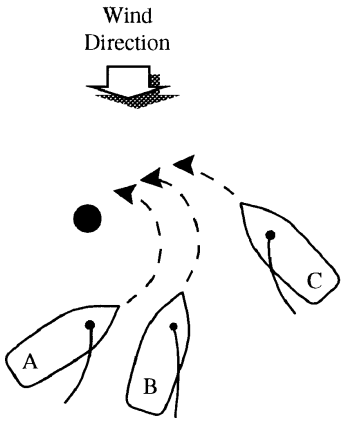


Figure 1. A sample racing situation.

implemented the racing rules in *SailRule* – a program that runs on a range of hand-held devices.

For example, the program runs on PalmOS™ and WinCE™ devices. PalmOS™ is a small operating system that supports hand-held devices like the Palm Pilot™ and the Handspring Visor™. WinCE™ (Compact Edition) is a cut-down version of Microsoft Windows for small PC-like devices. (All the foregoing names are trademarks.) PalmOS and WinCE account for a large proportion of the millions of PDAs in current use. As a result, the sailing rules can be made available on a wide variety of pocket-sized devices. These are highly portable, and can be used on the water (say, by a coach) or in the club-house (say, during a protest hearing). Use of the software by someone actually sailing a race might be regarded as unfair, not to mention impractical!

In fact, the program is not restricted to running on a PDA. It can be run on a standard computer (such as a PC) under a variety of operating systems. It is even possible to run the program in a suitably configured web browser. As a service to the sailing community, the *SailRule* program has been made available for public download and free use at: www.cs.stir.ac.uk/~kjt/software/pda/sailrule.html.

The program is distributed under the GNU General Public Licence. Basically this means it is free and without warranty.

1.3. *Related Work*. As far as the authors know, the *SailRule* program is unique. Creating it was not a trivial task. The constraints of a PDA (limited processing power, memory and screen size) place critical demands on how the program is implemented. It has also been necessary to formalise and to codify the racing rules for computer use.

From a sports point of view, the aim of this work was to help improve competitive performance. Understanding the rules better allows a sailor to take legitimate tactical advantage of the rules, as well as to avoid right-of-way boats. From the perspective of sports science, the work embodies novel research on formalising the rules of sailing races. For the computer scientist, the challenge was to implement a complex set of rules within the constraints of a small hand-held device.

2. BACKGROUND.

For the non-sailor, this section gives an overview of

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

sailing terminology and the racing rules. The SuperWaba programming environment for hand-held devices is also briefly introduced.

2.1. *Sailing Terminology*. Sailing employs a very large vocabulary of specialised terms. For the benefit of the non-sailor, the terminology needed for this article is explained below. A number of the terms are illustrated in Figure 2.

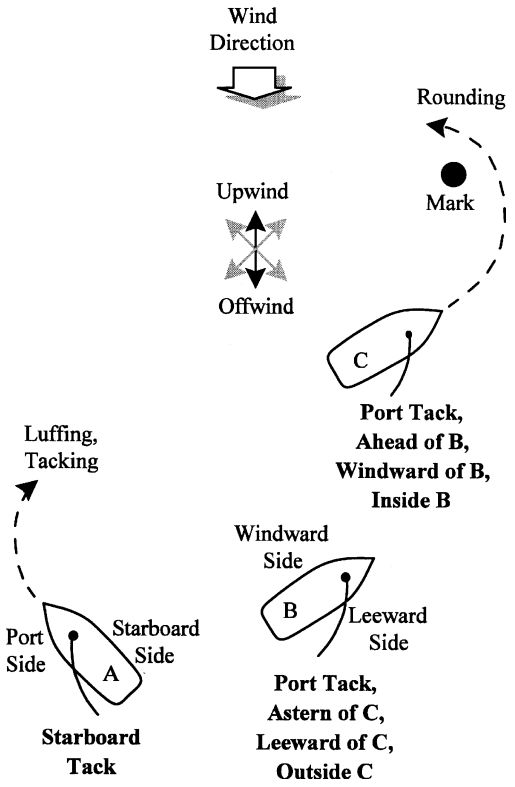


Figure 2. Some sailing terms.

Boat: This means any sailing vessel such as a yacht, a dinghy or a wind-surfer (sail-board). By common usage, boats are referred to as 'she'.

Course, *Leg*: A course is the route defined for the race. This is split up into a number of sections called legs that lie between course marks.

Mark, *Rounding*, *Obstruction*: Boats follow a route that is determined by marks such as buoys. A boat is said to round a mark when she passes it onto the next leg of the course. An obstruction is anything like a rock or a shoal that has to be avoided.

Windward, *Leeward*: Windward means towards the point from which the wind is blowing. Leeward is the opposite, i.e. towards the lee (shelter).

Head to Wind, *Close-hauled*, *Luffing*: A boat may not point too directly into the wind or her speed falls off dramatically. A boat is head-to-wind if she is heading towards the point from which the wind is blowing. In practice, a boat can sail at an angle of about 45 degrees away from the wind, called being close-hauled. A boat is said to luff if she steers closer to the wind.

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

Port, *Starboard*, *On a Tack*: Port means left, while starboard means right. A boat is said to be on a tack determined by the side on which her sail lies. A port tack boat has her sail out to the right; a starboard tack boat is the converse.

Tack, *Gybe*: A boat tacks when her bow crosses the wind from one tack to the other. A boat gybes when her stern crosses the wind from one tack to the other.

Clear Ahead, *Clear Astern*, *Overlapped*: One boat is clear ahead of another if she is in front; clear astern is the opposite. Two boats are directly overlapped if neither is fully ahead nor fully astern of the other.

Upwind, *Offwind*, *Downwind*: Upwind means towards where the wind is blowing from, while offwind (downwind) means away from the wind.

Inside, *Outside*: An inside boat is one whose path takes her closer to the mark; an outside boat is further away.

Above Proper Course, *Below Proper Course*: A boat's proper course will take her efficiently to the next mark of the course. Sailing above (or below) the proper course means sailing closer to (or further away from) the wind than this course would dictate.

2.2. *Racing Rules*. The racing rules are organised into a number of parts. Of direct concern to the *SailRule* program are the definitions of terms and the rules for when boats meet. A boat on her own in open water (i.e. not close to any other boat) may sail freely. When boats approach each other, the rules define which boats have right of way over which others. Roughly speaking, boats are considered to be close if they are within two boat-lengths of each other. The simplest rule cases concern a pair of boats as they approach. Three basic rules apply:

(a) a starboard tack boat has right of way over a port tack boat;
(b) if both boats are on the same tack, the one ahead has right of way over the one astern;
(c) if both boats are on the same tack but overlapped, the one to leeward has right of way over the one to windward.

In practice, the rules are rather more complicated. For example, if a boat is being overtaken to windward, she may luff the other boat (i.e. steer towards her) in order to avoid having her wind stolen. However, there are restrictions on when this may be done and to what extent.

More complex situations involve more than two boats. These may be on different tacks, ahead, overlapped or astern. To some extent, the rules can simply be applied to each pair of boats. For example, a starboard tack boat usually has right of way over a port tack boat. But other rules deal with a complete group of boats. For example, a boat may luff an overlapped boat only if she is allowed to luff all the boats overlapped to windward.

As boats approach a mark or obstruction, the rules are changed. At an offwind mark (sailing away from the wind), the normal port/starboard rule may be suspended; instead, an inside boat may have right of way over one not sailing so close to the mark.

To complicate matters further, the rules take the dynamics of a situation into account. It is not sufficient to take a snapshot and then apply the rules; the past behaviour of the boats can be important. Suppose that two boats are overlapped when approaching a mark, but the overlap is then broken. Although there is no current overlap, the fact that they were previously overlapped affects the rules. In

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

other cases, what the boats are presently doing is relevant. For example, a starboard tack boat normally has right over way over port tack. But if the boat is only just tacking onto starboard then she does not yet have right of way. Some rules are also varied at the start of a race.

As can be seen, the rules are not straightforward. It is not so surprising that amateur sailors find them difficult to grasp. A computer-supported tuition and analysis aid is therefore very useful.

2.3. *SuperWaba*. The style of the sailing rules immediately suggests some kind of expert system. This could analyse a situation and apply the relevant rules to draw inferences about which boats have right of way. Unfortunately expert systems are complex and computationally hungry, so they are not suitable for hand-held devices.

Java would be an appropriate choice of implementation language in order to maximise portability. However, Java is a sophisticated language with a very large class library. It is difficult to support on small hand-held devices. Several creditable attempts have been made, such as the K Virtual Machine (java.sun.com/products/kvm) and Spotless (www.sun.com/research/spotless). In the authors' experience, the most attractive of the Java look-alikes is SuperWaba (www.superwaba.org).

SuperWaba is an open-source project, making it accessible to a wide number of developers. It respects a substantial portion of the Java language. The restrictions in SuperWaba, such as lack of exception handling, are not particularly significant for implementing the sailing rules. SuperWaba is supported by an enthusiastic group of developers who have helped to create a respectable class library. This is *not* the same as the Java class library, though they share some common features. A Java programmer can, however, learn to use SuperWaba fairly quickly.

In fact, SuperWaba builds on a standard Java development environment. For example, standard Java tools like a compiler are used. SuperWaba programs may be previewed by the usual applet viewer or in a web browser. This is extremely useful because development can then take place on a desktop computer. The developer can take advantage of a full operating system, filing system, IDE (Integrated Development Environment), etc. Once the program is fully developed, a utility converts it into a form that can be loaded and executed on a PalmOS or WinCE device. Developing software *on* such a small device is difficult because of the extremely limited facilities.

Like Java, SuperWaba compiles into compact code. This is executed by a Java-like virtual machine on the hand-held device. The main disadvantage of SuperWaba is that it does not compile into native code, so its execution is relatively slow (especially on the limited processors in PDAs). In addition, a user cannot execute the code immediately. The freely available SuperWaba virtual machine must first be installed, but this can then be used for a variety of SuperWaba programs. Because SuperWaba is designed to run on a range of devices, it is not integrated with the PDA's operating system. For example, graphical widgets like buttons and menus are those of SuperWaba and not the native ones. However, the foregoing weaknesses are more than offset by the portability and popularity of SuperWaba.

It is possible to write SuperWaba programs as one might write Java, i.e. without regard to memory or processor limitations. The most obvious restriction is the limited screen size of a PDA (e.g. 160×160 pixels on a Palm device). However, it is necessary to bear the target environment in mind. For example, complex data structures, wasteful use of memory, and inefficient algorithms should be avoided. The SuperWaba program will run, certainly, but its performance may be unacceptable. In

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

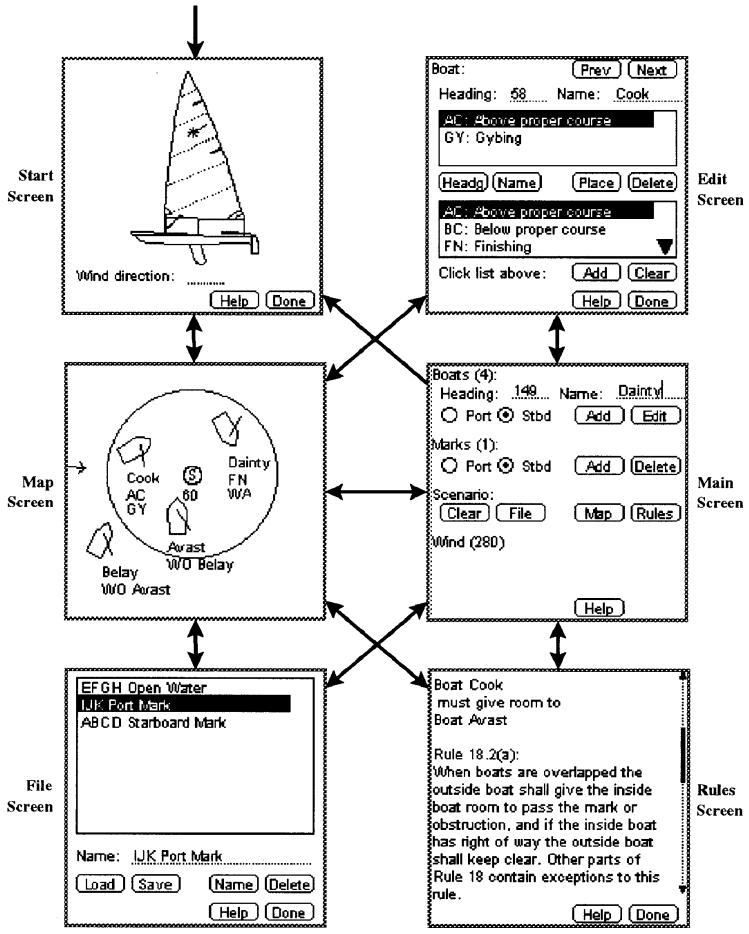


Figure 3. Main screens for racing rules program.

practice, care in program design will allow SuperWaba to execute with acceptable efficiency on a PDA.

3. PROGRAM DESIGN. This section describes the two key activities of the *SailRule* program: drawing a scenario, and determining the applicable rules. The first concerns the design of the user interface. In principle, this is straightforward, though the limited screen size is a notable constraint. The second activity requires the racing rules to be formulated in a fashion that allows efficient retrieval according to the scenario. Formalising and codifying the rules was a significant and novel piece of work.

3.1. *User Interface*. Figure 3 shows the main screens for *SailRule*. The diagrams are actual screenshots. The arrows in the figure show how the user can progress from screen to screen. Nearly all the screens have help pages (not shown). This is important because the program is intended for non-technical users in any location. It would be unsatisfactory to have the program on a pocket-sized device but to need to carry around a large manual!

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

The program begins with the start screen, where the wind direction is entered. By convention, directions are angles in degrees measuring clockwise from 0 as North (the top of a diagram). The wind direction is a critical parameter because it determines how boats are displayed. The sail position and tack of a boat are automatically determined by its heading relative to the wind. For this reason it is not allowed to change the wind direction while drawing a scenario. If necessary the current scenario can be cleared, returning to the start screen.

Scenarios are most easily drawn on the map screen. To keep this clean, there are deliberately no controls on this screen. Instead, menu choices or simple keystrokes are used to control the actions. For example, 'F' calls up the file screen and 'R' states the rules that apply. The wind direction is shown by an arrow at the edge of the screen (280 degrees in the map screen of Figure 3). Boats are added to the scenario by clicking on the screen. Subsequently, boats may be dragged to the desired position and may be rotated to the required heading. The sail position and tack are calculated automatically knowing the wind direction. As boats are entered, they are simply numbered.

For a scenario in open water, this is sufficient. However, rule disputes usually arise as boats round a mark. The map screen therefore allows a mark to be added. Certain rules come into play as boats approach a mark. The significant region is that defined by a circle round the mark with a radius of two boat lengths. (Technically this is twice the length of the nearest boat, but this complication is ignored by the program.) The map screen in Figure 3 shows this two-lengths circle; Avast, Cook and Dainty are within it.

An 'S' against a mark indicates that the mark is rounded to starboard (i.e. the boats keep the mark to starboard as they pass it); a 'P' (port) mark is also possible. It is necessary to know the direction to the next mark in the course. The map screen of Figure 3 shows this as 60 (degrees). Avast, Belay and Cook are therefore moving clockwise round the mark, aiming to sail towards the two o'clock position. Boat Dainty has already rounded the mark.

The map screen does not provide fine control of the scenario, though it is much easier to use because it is graphical. The main screen offers complete control. If desired the tack, heading and name can be entered for each boat. The automatic determination of tack can be partly overridden in this way, e.g. to show a boat 'sailing by the lee' (in danger of gybing accidentally). Boat parameters can be edited. Marks can be added and deleted (though because of screen limitations only one mark is currently allowed). The main screen also allows other screens to be visited such as for filing and boat editing.

The edit screen allows boat parameters such as heading, name and position to be changed. More importantly, boat status can be declared. A number of rules depend on what boats are currently doing and what their previous relationship was. That is, the rules depend on the dynamics of the situation. Short of animating a scenario, this is difficult to depict. The program solves this by representing current and previous activity.

The map screen of Figure 3 shows some examples of what boats are currently doing. Cook is currently sailing above her proper course (AC) and is gybing (GY). Dainty is currently finishing the race (FN). The map screen also shows some examples of previous relationships. Previously, Avast was overlapped (WO) on Belay. This is important when rounding a mark. Although Belay is not currently overlapped by

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

Avast, she was previously overlapped as Avast entered the two-lengths circle. Another previous relationship is that Dainty was ahead (WA) of the other boats when approaching the mark.

By means of the map screen, main screen or edit screen the user draws the scenario to be analysed. This scenario can be stored in a file by entering the file screen and saving it. Previously stored scenarios can be retrieved and edited. They can be saved under another name, making it easy to create variations on a base scenario. In practice, SuperWaba does not save scenarios as individual files; they are stored as records in the program's database. However, the *SailRule* program refers to 'files' as they behave like normal computer files.

The program database is created separately on the hand-held device and is separately backed up. This means that it is possible to provide the user with pre-prepared scenarios. The *SailRule* program is supplied with a substantial set of 45 examples illustrating the main rule cases. Users can also swap scenarios. If the *SailRule* program were to be used widely, an extensive library of scenarios could be made available from a web site. This would be very valuable for training, and could be used to record 'case law'. This arises when national sailing bodies provide their interpretations of difficult rule cases.

3.2. *Invoking The Rules*. Once a scenario has been set up, selecting the rules screen will cause the applicable rules to be displayed. For each pair of nearby boats, their relative right of way is stated and justified by reference to the relevant rules. Sometimes more than one rule may justify the right of way. As an example, the scenario in Figure 1 yields the following literal output:

(a) Boat B must give room to Boat A – Rule 18.2(a): When boats are overlapped the outside boat shall give the inside boat room to pass the mark or obstruction, and if the inside boat has right of way the outside boat shall keep clear. Other parts of Rule 18 contain exceptions to this rule.
(b) Boat C has right of way over Boat B – Rule 10: When boats are on opposite tacks, a port-tack boat shall keep clear of a starboard-tack boat.

The program aims to provide a succinct statement of the rules. For example, if A has right of way over B, it does not restate that B must give way to A. Many rules can be repeated if there are multiple boats. Only the first instance of a rule is cited in full; the rule is referenced by number if it is used for another pair of boats.

As a more complex example, the scenario in the map screen of Figure 3 yields the following output:

(a) Boat Avast has right of way over Boat Belay – Rule 11: When boats are on the same tack and overlapped, a windward boat shall keep clear of a leeward boat.
(b) Boat Cook must give room to Boat Avast – Rule 18.2(a): When boats are overlapped the outside boat shall give the inside boat room to pass the mark or obstruction, and if the inside boat has right of way the outside boat shall keep clear. Other parts of Rule 18 contain exceptions to this rule.
(c) Boat Dainty has right of way over Boat Avast – Rule 10: When boats are on opposite tacks, a port-tack boat shall keep clear of a starboard-tack boat.
(d) Boat Cook has right of way over Boat Belay – Rule 12: When boats are on the same tack and not overlapped, a boat clear astern shall keep clear of a boat clear ahead.

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

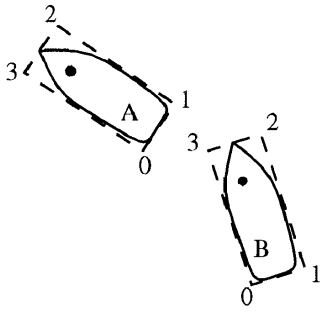


Figure 4. Relationship between boats.

3.3. *Checking Boat Relationships*. As will be seen, there are many factors that determine the application of the rules. The program has to perform many geometrical calculations and logical inferences in order to decide how the rules apply. Many of the rules require the relationship between two boats to be decided: leeward, astern, overlapped, etc. This is achieved by considering how the boats lie in relation to each other. As a small simplification, each boat is considered to be bounded by a rectangle as shown in Figure 4. The corners of this rectangle are numbered from 0 to 3. Each relationship is then checked between the corners of these rectangles.

Although the relationships are a straightforward matter of coordinate geometry, there do not seem to be standard formulae that are efficient to program. The authors therefore devised their own approach. As an example, suppose it is necessary to check whether B in Figure 4 is clear astern of A. Each of B's corners is checked against the (extended) line from corner 0 to corner 1 of A. If all of B's corners are behind this line, then B is clear astern. If one or more corners are ahead of this line, then B is overlapped or clear ahead.

The *SailRule* program follows the usual convention of screen-based programs. The point (*0*, *0*) is at the top left of the screen, and the y coordinate grows downwards. Since lines may be vertical, they cannot be represented in $mx+c$ form (gradient m, y intercept c). Instead the endpoints of the line are used: (*x0*, *y0*) and (*x1*, *y1*). The difference between these x and y coordinates is calculated: *xdiff* is $x1-x0$, and *ydiff* is $y1-y0$. Now a candidate point (x, y) can be checked for being astern of a line. The algorithm is:

```
if xdiff ≠ 0
  then astern := (y − y0) × xdiff > (x − x0) × ydiff
  else
      if ydiff > 0
        then astern := x < x0
        else astern := x > x0
```

If the x difference is non-zero, the line is not vertical. Whether the point is astern depends on its displacement from the start of the line (*x0*, *y0*) and the extent of the line (*xdiff*, *ydiff*). If the x difference is zero, the line is vertical. The check now splits on whether the y extent is positive, and the x coordinate relative to the start of the line.

Similar rules are used for checking clear ahead, leeward and windward. Checking for overlap is more complex than might be expected. Two boats overlap directly if

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

neither is clear astern of the other and there is no intervening boat. However, two boats are considered to overlap if there is an intervening boat that is overlapped on both. In Figure 5, A is clear astern C and so does not overlap it directly. But because

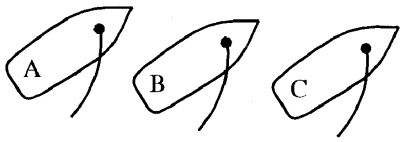


Figure 5. Boats overlapping.

A is overlapped on B, and B on C, A is considered to be overlapped by C. The logic behind this is that actions by A will affect C.

3.4. *Formalising The Racing Rules*. The racing rules are defined in natural language. Although they are written very carefully, they are not in a form that a program can use. Furthermore, like any natural language statement they often rely on the reader's common sense or sailing experience for their interpretation.

For example, the rules rely on the notion of one boat being to port of another. Although obvious (in most cases), surprisingly this term is not explicitly defined in the ISAF rules. In fact 'to port' is ambiguous, so the program aims to reach the same judgment that a person would. The authors also discovered other incomplete or unclear wordings in the rules, and had to interpret them in a sensible way.

Computer programs need explicit algorithmic instructions. A major challenge was to represent the rules in a form that allows their efficient matching to the scenario being analysed. This is far from easy since a scenario contains graphical and dynamic information, whereas the rules are textual.

The problem was solved by identifying the key factors that cause a rule to become applicable. This information had to be teased out of the manner in which the rules are written. The rules generally deal with a pair of boats, although by extension they apply to relationships involving multiple boats. The relevant ISAF rules are those of Part 2 ('when boats meet'). The following factors are significant:

(a) whether a boat is on a port or starboard tack, or is tacking to port or starboard;
(b) whether a boat is head-to-wind, sailing above or below her proper course;
(c) whether the boats are in danger of colliding;
(d) whether the boats are sailing in open water or near a mark;
(e) whether one or both boats is within two lengths of a mark, obstruction or other boat;
(f) whether a pair of boats is on the opposite or the same tack;
(g) whether one boat is clear ahead of another, clear astern or overlapped;
(h) whether one boat is to leeward or to port of another.

Each rule was then indexed by which of these factors apply. A number of rules are indexed by multiple factors. If all these factors apply in a scenario, the rule applies between the pair of boats.

The graphical scenario is used to identify the factors that apply to each distinct pair of boats. This produces a list of candidate rules. The program's rules resolver then checks a number of additional issues that may modify which rules apply or how right of way is determined. These further factors deal with:

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

(a) whether a boat is approaching a windward mark;
(b) whether a boat needs to tack in order to round a mark;
(c) whether a boat is tacking within the two-lengths circle round a mark;
(d) whether a boat is above close-hauled, has already passed head-to-wind during a tack, or has completed her tack;
(e) whether one boat was ahead of, or overlapped on, the other before entering the two-lengths circle;
(f) whether one boat is inside the other when rounding a mark.

As an example, consider A and B in Figure 1. Both boats are on port tack, and within two lengths of the mark. A is overlapped on B and on the inside line to the mark. This selects rule 18.2(a) as a candidate. The rules resolver determines that the boats are approaching a windward mark, that they need to tack in order to round the mark, and that they were overlapped prior to enter the two-lengths circle. This confirms that rule 18.2(a) does indeed apply, and that B must give room to A under this rule.

3.5. *Program Design.* The program was implemented using SuperWaba. The design is object-oriented. For example, boats and rules are implemented as classes. Each new boat or rule is an object as an instance of its defining class. The program contains about 3,400 non-comment lines of code, developed with about seven man-months of effort. The corresponding object code takes about 96 kilobytes of storage, and the program runs with about 50 kilobytes of dynamic memory. This is reasonable considering that the program has such a complex task to perform. PDAs usually have memory measured in the megabytes.

The most intensive activity for the program is determining and displaying the rules that apply to a scenario. As an example, it takes a Palm m505 (33 MHz processor) about three seconds to show the rules applying to the scenario in Figure 3. In fact this is quite a complex scenario with four boats rounding a mark, plus previous and current behaviour to take into account. Execution speed is quite reasonable, bearing in mind that the program must perform a lot of geometrical calculations and logical inferences. PDA processors are also relatively slow.

In fact, the program can also be executed on a desktop computer. Instead of the SuperWaba virtual machine it requires elements of the SuperWaba Development Kit, which is also freely downloadable. The program can execute on a standard applet viewer or web browser. Using a PC obviously gives much greater processing and storage capacity. However, the program screen size is maintained at that of a typical PDA for consistency.

4. CONCLUSION.

The objectives and design of the *SailRule* program have been presented. The program allows sailing rule scenarios to be created graphically, loaded, saved and manipulated. The rules are stored in a database that may be distributed and used to evaluate knowledge of racing scenarios. It has been explained how the program codifies the racing rules so that they may be efficiently applied to a scenario.

Examples have been given of how the program represents scenarios and reports the applicable rules. The program runs on a wide variety of hand-held devices such as those running PalmOS or WinCE. Memory and execution requirements are reasonable.

The program meets the objectives set out in Section 1.2.:

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press

(a) It accurately represents the geometrical and dynamic information describing a rules scenario. This information is stored in computer form and can be readily analysed. The graphical user interface reflects sailing terminology and concepts.
(b) The program makes it easy to prepare libraries of scenarios for analysis and training.
(c) The program lends itself to 'what if' studies. Thus small variations in a scenario can be created, e.g. whether a boat was overlapped or not, to see how the rules would change.
(d) Although the program could not replace a protest jury, it is nonetheless a way of recording the basic facts and providing an initial analysis of the rights of way.

Although the program does not embody every rule, it supports the major rules that often cause problems in understanding. Future work will include handling additional rules, such as those governing obstructions and acquiring right of way. However, the program has already demonstrated its potential to improve racing performance. It is hoped that the sailing community finds it of value.

ACKNOWLEDGEMENT

The authors warmly thank Mike Harrison, Port Edgar Yacht Club, for reviewing a draft of the article.

https://doi.org/10.1017/S0373463302001765 Published online by Cambridge University Press