Appendix A

Empirical Modelling of a Sailboat

This chapter describes my experiences of constructing a sailboat simulator (SBS). The physical properties of the sail, rig and hull are described within the model. Following the principles introduced in Chapter 2, the state of the SBS is described using a definitive script, and an agent-oriented design method is used to determine and construct each of the sailboat components: sail, rig, hull and sailor. The resulting simulation combines a model of the sailboat dynamics, a simple graphical animation and an interface through which the user can play the role of the sailor.

A.1 Common-sense knowledge

Sailing and sailboats were already familiar subjects to me before I began constructing the SBS. This knowledge was based on my experience of sailing various craft, including dinghies, sailboards and a yacht. I knew

- how to react to situations as they arose in the boat,
- how to predict situations arising in and around the boat,
- how to devise courses of action to deal with predicted situations, and
- the meaning of sailing terms for giving and understanding instructions.

In other words, I *knew* how to sail. I shared this common-sense [Wol92] sailing knowledge with everybody else who was able to sail. My task was to represent this essentially subjective, practical, vague, inconsistent, situated, analogical [HT95] and phenomenological [GS83] knowledge of how a sailboat behaves in the form of a simulation.

A.2 Agent-oriented modelling

I found that the concepts of LSD corresponded to my common-sense notions of sailing. I was able to identify four agents:

- the sailor who steers the sailboat and adjusts the position of the sail;
- the sail that is blown by the wind;
- the rig that holds the sail between a boom and mast;
- the hull that holds the rig and is stablized by a keel.

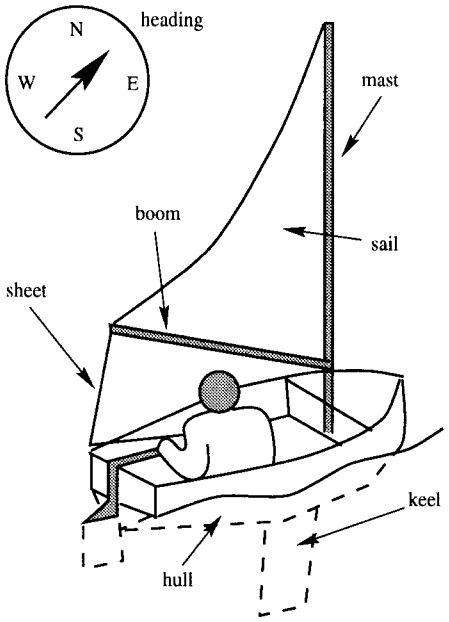
The sailor agent was the easiest to identify because it was simply me. The other three agents I identified by considering causality between agents. I began with the sail agent and considered what stops the sail from blowing away in the wind. This thought process resulted in the identification of the rig agent. Similarly, by thinking what stops the rig from blowing over I identified the hull agent. This was essentially common-sense thinking involving personification and the notions of causality [Wol92, HT95].

A.3 Observation-oriented modelling

I continued my LSD specification of the SBS by identifying the observables associated with each agent. The oracles and handles of the sailor were the easiest to identify because I had only to remember what I observed during sailing. Having identified the oracles and handles of the sailor agent, as shown in Example A.1, I then went about forming oracle-handle pairs by attributing oracles and handles to

the sail, rig and hull agents. Some oracles and handles did not make a pair, such as the oracle for wind direction, suggesting an openness about the model.

Example A.1. Representing the modeller in the SBS. By defining the sailor agent in LSD I was effectively modelling myself. The diagram shows what I was aware of while sailing.




```
agent sailor {
oracle
  list
  driving_force
  hull_speed
  wind_dir
  heading
  sheetlenmin
  sheetlenmax
  sheet_len
handle
  heading
  sheet_len
derivate
  turn = user_input(turn_type)
  sheetdir = user_input(sheet_type)
protocol
  turn == starboard -> inc(heading)
  turn == port -> dec(heading)
  (sheetdir == out) &&
  (sheet_len < sheetlenmax) -> inc(sheet_len)
  (sheetdir == in) &&
  (sheet_len > sheetlenmin) -> dec(sheet_len)
}
```

Having defined the oracles and handles for the sailor agent I formed oracle-handle pairs by attributing oracles and handles to the sail, rig and hull agents.

When I came to define the derivates and protocols for the sail, rig and hull agents I had a decision to make. I could either

- represent my common-sense knowledge of causality and agency in sailing as mainly protocol definitions, or
- apply my school-book knowledge of Newtonian mechanics to explain the relations between observables, represent this relation as derivates and test the resulting definition against my common-sense knowledge.

I had the choice between two approaches because of my combined sailing and scientific background. However, if I was a sailor who had not learned about physics

then I would have produced a model based on my common-sense knowledge alone (cf. naive physical models of children [GS83]) suggesting that theory is not necessary for EM. But, the fact remained, that I did know about physics and decided that a more generalized and complete model would result from taking the second choice.

During the definition of the derivates for the first agent, the sail agent, I found my theoretical knowledge to be insufficient and had to resort to directly representing my common-sense knowledge. Using the idea of vectors helped to explain some of the observations of the sail, such as the angle between the sail and wind resulting from subtracting one from the other. However, I could not apply my theoretical knowledge to explain the driving force of the sail with respect to its angle to the wind or to derive the values of constants. I have since discovered that modelling and simulating aerodynamic effects from first principles has always been a largely unsolved problem in mathematics and physics [Asp90, MK97, sai63]. I avoided this problem by representing my knowledge as a function in terms of sail angle relative to the wind. Example A.2 shows the sail agent definition and the visualization/animation (the visualization and animation are not distinguished by a screen-shot) that I used to test my theory for the sail forces.

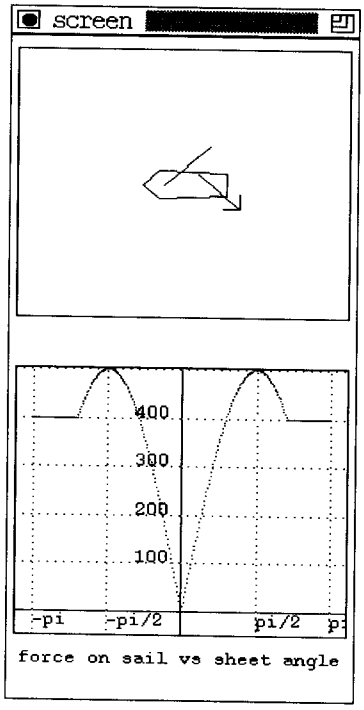
By simplifying my model of the dynamics of the hull in the water I was able to define the rig and hull agents almost entirely in terms of physical theories of motion, the only exceptions being the values of constants. The rig and hull agent definitions use integrals to represent the invariant relation between observables over time. For example, the speed of the sailboat at any time is an integral of the acceleration of the boat. This had the effect of introducing time into the model as an observable. Example A.3 shows the rig and hull agent definitions and the visualization/animation that I used to test my theory of sailboat forces.

A.4 Definitive representation of the sailboat

While defining the sail agent I would constantly refer to and modify the visualization/animation, shown in Example A.2, in order to test the emerging theory of the sail forces and discover appropriate values for constants. DoNaLD and SCOUT scripts, defining the image of the sailboat and forces, were written at the start

of modelling and remained essentially the same throughout. I would repeatedly convert the derivates into definitions in EDEN and experiment with the resulting visualization by changing the sheet variable with the graph, shown in Example A.2, emerging over time. In order to save time I defined an EDEN action that changed the sheet observable automatically thus animating the visualization.

Example A.2. Representing the sail in the SBS. I defined the sail as an LSD agent and then tested the specification using the combined visualization and animation shown in the diagram.



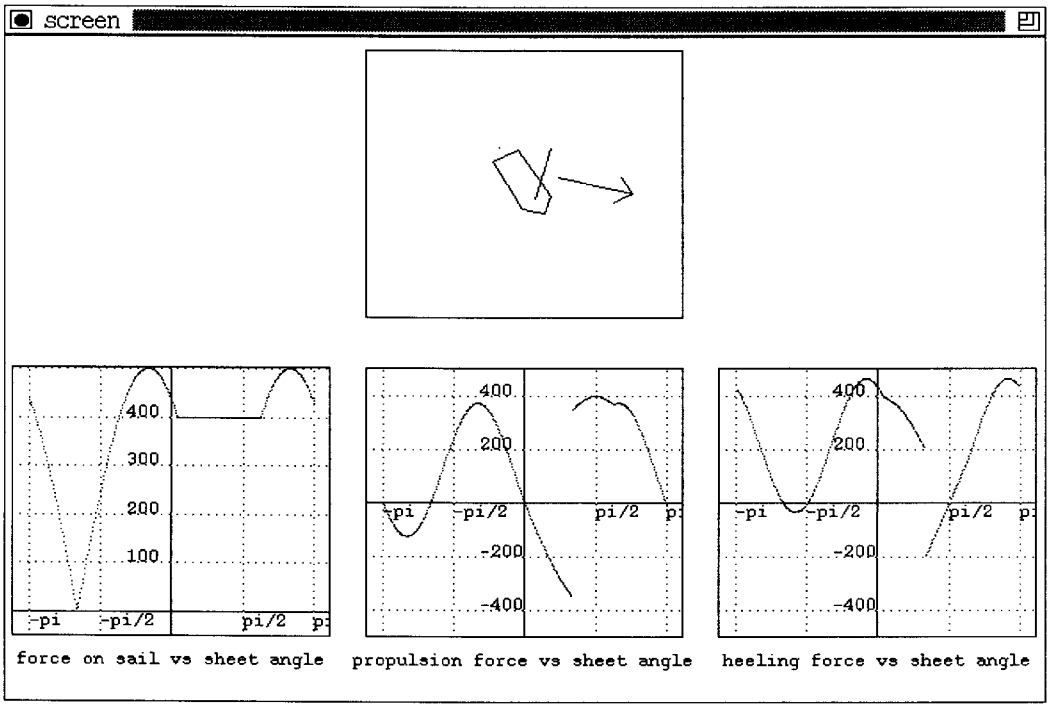

```
agent sail {
const
  drivingFmin = 400.0 // minimum driving force [N]
  drivingFmax = 500.0 // maximum driving force [N]
  k = asin(drivingFmin/drivingFmax))
state
  sail_dir      // sail direction as bearing [rad]
  sail_wind     // angle between sail and wind [rad]
  driving_dir   // sail driven anti/clockwise [1/-1]
  driving_force // driving force of sail [N]
oracle
  heading
  wind_dir
  sheet
derivate
  sail_dir = heading + sheet
  sail_wind = wind_dir - sail_dir
  driving_dir = (sin(sail_wind) < 0.0) ? 1 : -1
  driving_force is
    (cos(sail_wind) < -cos(k)) ?
    drivingFmin : abs(sin(sail_wind))*drivingFmax;
}
```

I had to experiment with the simulation in order to find the appropriate representation for the driving force and values for constants.

The visualization/animation of the sail was extended to include the graphs of propulsion and heeling force experienced by the sailboat, as shown in Example A.3. The new screen image was simply created by appending a script defining the new graphs to the existing DoNaLD and SCOUT scripts. The derivates were converted into definitions except for the integral derivates that were implemented using EDEN actions, as shown in Example A.3. The new visualization/animation was used in the same way as the original to test the emerging theory of sailboat forces.

Example A.3. Representing the rig and hull in the SBS. I defined the rig and hull agents in LSD and tested the specifications using the visualization/animation shown in the diagram.

```
agent rig {
const
  boom_len = 2.5  // length of boom [m]
  mast_len = 4.0  // height of mast [m]
  rig_moi = 200.0 // moi of rig [kg m^2]
  resistK = 50.0  // friction constant
state
  sheet             // angle between keel and sail [rad]
  sheet_set         // setting of sheet [rad]
  sheet_len         // length of sheet [m]
  sailT             // torque of sail about mast [Nm]
  resistT           // dampening torque [Nm]
  sailAacc          // angular acc of sail [rad/s^2]
  sailAvel          // angular vel of sail [rad/s]
oracle
  driving_force
  driving_dir
  t                 // time [s]
handle
  sailAvel
derivate
  sailT = driving_force * -driving_dir * boom_len / 2.0
  resistT = resistK * -sailAvel
  sailAacc = (sailT + resistT) / rig_moi
  sailAvel = integ_wrt(sailAacc,t)
  sheet = integ_wrt(sailAvel, t)
  sheet_set = 2 * acos(sqrt(1 - (sheet_len*sheet_len)
                                / (4*boom_len*boom_len)))
}
```



```
agent hull {
const
  dragK = 100.0          // drag coefficient of water
  boat_mass = 400.0      // mass of boat [kg]
  ballast_mass = 100.0 // mass of ballast [kg]
  ballast_weight = ballast_mass * g // weight of ballast [N]
  keel_depth = 1.5       // depth of keel holding ballast [m]
  hull_moi = 600.0       // moment of inertia of hull [kg m^2]
  dampK = 500.0          // resistance coefficient to hull listing
state
  hull_speed             // boat speed in water [m/s]
  list                   // angle of boat from vertical [rad]
  drag                   // drag of boat in water
  forward_force          // forward force of boat [N]
  acceleration           // acceleration of boat [m/s^2]
  hull_speed             // speed of boat [m/s]
  side_force             // sideways force of boat [N]
  sailTq                 // torque of sail about hull [Nm]
  ballastT               // torque of keel about hull [Nm]
  dampT                  // dampening torque [Nm]
  hullAacc               // angular acceleration of hull [rad/s^2]
oracle
  driving_force
  driving_dir
  sheet
  t                      // time [s]
derivate
  drag = dragK * hull_speed
  forward_force = driving_force * -sin(sheet) * driving_dir
  acceleration = (forward_force - drag) / boat_mass
  hull_speed = integ_wrt(acceleration, t)
  side_force = driving_force * cos(sheet) * driving_dir
  sailTq = side_force * mast_len / 3.0
  ballastT = ballast_weight * sin(list) * keel_depth * -driving_dir
  dampT = dampK * -hullAvel
  hullAacc = (sailTq + ballastT + dampT) / hull_moi
  hullAvel = integ_wrt(hullAcc, t)
  list = integ_wrt(hullAvel, t)
}
```

I generated the EDEN script for the rig and hull agents by transforming the derivates into definitions except for the integral derivates. The hull speed integral derivate, for example, was implemented by the EDEN action

```
/* hull_speed = integ_wrt(acceleration, t) */
proc integ_hull_speed : iClock {
  hull_speed = hull_speed_iVal + (acceleration * iPeriod / 2.0);
  hull_speed_iVal = hull_speed + (acceleration * iPeriod / 2.0);
}
```

and the EDEN action for the clock.

A.5 Exploring the sailboat simulation

The SBS was constructed with the aim of recreating the experience of sailing so that I could use my knowledge of sailing directly to test theories and discover values for constants. The animation that allowed this use of knowledge is shown in Examples A.4 and A.5. The sailboat was represented by a view from above and from the stern (rear) and an interface was defined through which I was able to control the boat by turning it anticlockwise (port) or clockwise (starboard) and reducing the length of the sheet (sheeting-in) or increasing the length of the sheet (sheeting-out). A speed indicator gave the current speed of the boat. Later a clock and driving force indicator were added. Although primitive, the animation served its purpose of recreating the experience of sailing for testing the model and finding appropriate values for constants.

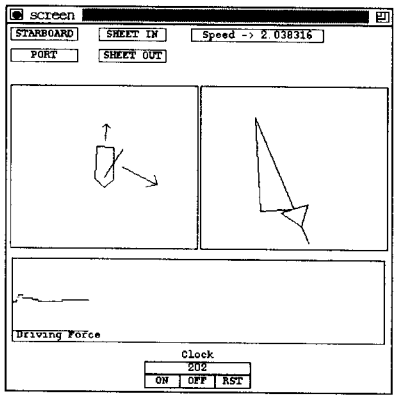
I was able to explore the SBS during its construction more as a sailor than a system developer. By interacting with the model via the interface or directly by changing the values of EDEN variables I explored the SBS:

- searching for emergent behaviours;
- considering the behaviour in terms of physical principles;
- performing both familiar and novel manoeuvres;
- shifting the context of the sailing experience by changing variable values;
- looking for confirmation that the model is faithful to my experience;
- searching for behaviours which are not faithful to my experience.

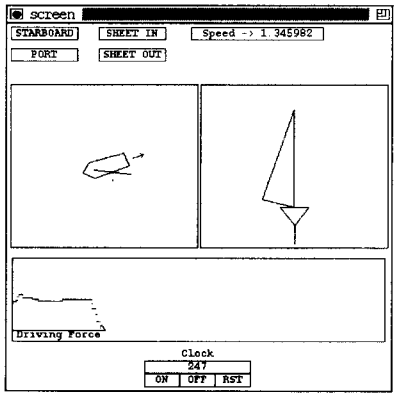
Example A.4 shows four screen-shots while I was performing a common sailing manoeuvre called a "tack" in which the boat turns half-circle through the wind. Because of my familiarity with this common sailing manoeuvre I knew what to expect. If the model did not meet my expectations I either changed my beliefs, thus learning from my interaction with the simulation, or changed the model. For example, I discovered during interaction with the model that it was possible to

capsize the boat, as shown in Example A.5. This represented a change in my beliefs about the scope of the model.

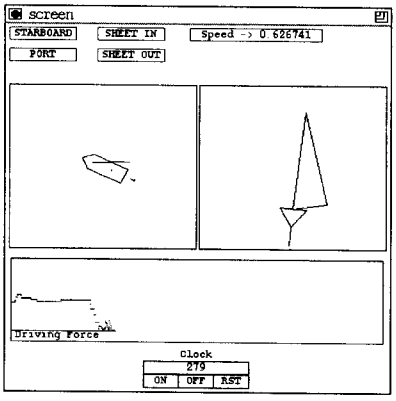
Example A.4. Exploration in the SBS. One way I explored the SBS was to perform familiar sailing manoeuvres. The following diagrams are screen-shots during a manoeuvre called a tack in which the sailboat turns half-circle through the wind.



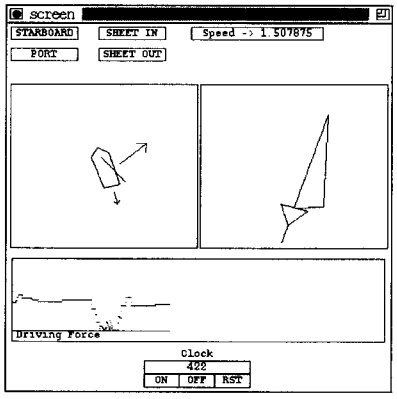
(a) Preparing to tack.



(b) Turning into wind (luffing).



(c) Sail changes sides.



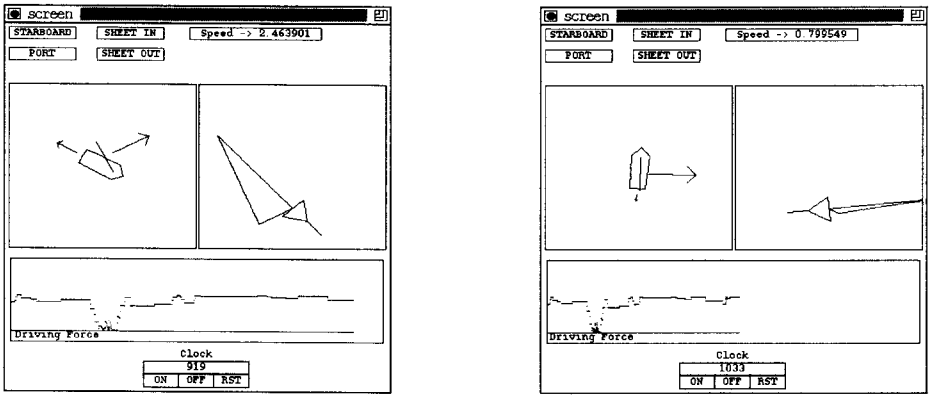
(d) Tack completed.

I found that the SBS provided a good approximation to the handling of a real sailboat.

A.6 Extending the sailboat model

Once I was satisfied with the SBS I placed it in the Empirical Modelling Group archives. Later, another modeller retrieved it and continued to extend the model. This modeller had no experience of sailing but had discovered a book that detailed

Example A.5. Emergence in the SBS. It emerged during the exploration of the SBS that it represented the sailboat capsizing. This was a surprise to me because I did not think of the capsize situation during modelling.



(a) Turning to port. (b) Sailboat capsizes.

This unexpected behaviour was discovered when I performed a manoeuvre incorrectly. The manoeuvre is called a "jibe" in which the stern (rear) passes through the wind thus making the sailboat unstable.

---

the phenomenon of turbulence on sails and the apparent shift in wind direction caused by the motion of the boat. The modeller was able to add this information to the existing LSD specification of the sail, as shown in Example A.6, without having to consult me. This provides evidence of the openness of models in EM and the ease with which they can be extended.

Example A.6. Openness in the SBS. Yung extended the LSD definition for the sail agent, shown in Example A.2, to include the effects of turbulence and the wind generated by the motion of the sailboat.

```
agent sail {
const
  FpushK = 10 // pushing force constant [N m^-3 s^-2]
  FsucK = 20 // suction force constant [N m^-3 s^-2]
state
  sail_dir      // sail direction as bearing [rad]
  driving_dir   // sail driven anti/clockwise [1/-1]
  driving_force // driving force of sail [N]
  sheet_angle   // angle between keel and sail [rad]
  sail_area     // effective sail area [m^2]
oracle
  rel_wind_dir // wind direction experienced by the sail [rad]
  rel_wind_vel // wind speed experienced by the sail [m s^-2]
  heading
  wind_dir
  sailAvel
derivate
  sail_dir = heading + sheet
  sail_wind = rel_wind_dir - sail_dir
  driving_dir = (sin(sail_wind) < 0.0) ? 1 : -1
  driving_force = rel_wind_vel * abs(cos(sail_wind)) * FsucK * sail_area
                + rel_wind_vel * abs(sin(sail_wind)) * FpushK * sail_area))
  sail_area = boom_len * mast_len * cos(list)
  rel_wind_vel = sqrt(wind_vel^2 + hull_speed^2 -
                              2*wind_vel*hull_speed*cos(wind_dir - heading))
  rel_wind_dir =
       heading - asin(sin(wind_dir-heading) * wind_vel / rel_wind_vel) + pi
}
```

The new definition is essentially the the original with the the addition of more constants, observables and derivates.