Appendix C

Definitive scripts for Restaurant Management Model

Developed by C. Roe and modified by T. Heron

/* NEWrest.e*/

```
/* Restaurant visualisation in Tkeden */
/* 27/1/00 by Chris Roe */
/*
 Revisions 19/11/01 by Tim Heron:
 Added blue line to show current time on timeline
Made colour highlighting hollow so we can see the text in the time line
Allow manager to choose table as well as suggesting a table (prevents manager double booking)
Reduced console commentary
Updated mouse events to work with latest tkeden
Now allows pre-defined events to take place instead of customers arriving randomly
Beeps when the clock stops
User not allowed to start clock until all events are dealt with

Future enhancements :
Combine tables together to create larger tables
Prevent user from restart clock without dealing with current events
Deal with multiple events occuring at the same time instance
*/

%eden

/* Restaurant definitions */

R_height is 70;
R_width is 80;
R_doorsize is 10;
R_numberoftables is 8;
R_tablecapacity is [4,4,2,2,4,4,2,2];
R_maxpartysize is 4;
R_tablewidth2 is 8;
R_tablewidth4 is 15;
R_tableheight2 is 10;
R_tableheight4 is 10;
R_busy is 50;
R_customersettlingtime is 2; /* number of minutes to hang up coats and sit down!! */
occupycolor = "green";
basecolor = "black";
starthour = 7;
stopClock = TRUE;
clock = 0;
potential_arrive_time = 0;
endclock = 240;
delay = 5000;
average_meal_time is 75;
booking_length is 90;
recorded_customer_data = []; /* to record information about customers */
customers_in_restaurant = [];
groups_in_restaurant = [];
allow_clock = 0; /* Counts events that need to be dealt with */
```

```
/* the fields in booking list are (name,table_number (determined by manager), number of people, arrival time,cancelled) */
b1name = “Russ”;
b1table = 1;
b1nopeople = 4;
b1arrivetime = 15;
b1cancelled = FALSE;
b2name = “Rasmequan”;
b2table = 7;
b2nopeople = 2;
b2arrivetime = 30;
b2cancelled = FALSE;
b3name = “Beynon”;
b3table = 4;
b3nopeople = 2;
b3arrivetime = 30;
b3cancelled = FALSE;
b4name = “Roe”;
b4table = 6;
b4nopeople = 4;
b4arrivetime = 45;
b4cancelled = FALSE;
b5name = “Ward”;
b5table = 8;
b5nopeople = 2;
b5arrivetime = 45;
b5cancelled = FALSE;
b6name = “Wong”;
b6table = 5;
b6nopeople = 4;
b6arrivetime = 60;
b6cancelled = FALSE;
b7name = “Maad”;
b7table = 3;
b7nopeople = 2;
b7arrivetime = 90;
b7cancelled = FALSE;
b8name = “Rungrattanaubol”;
b8table = 2;
b8nopeople = 2;
b8arrivetime = 90;
b8cancelled = FALSE;

booking_listis [[b1name,b1table,b1nopeople,b1arrivetime,b1cancelled],[b2name,b2table,b2nopeople,b2arrivetime,b2cancelled],[b3name,b3table,b3nopeople,b3arrivetime,b3cancelled],[b4name,b4table,b4nopeople,b4arrivetime,b4cancelled],[b5name,b5table,b5nopeople,b5arrivetime,b5cancelled],[b6name,b6table,b6nopeople,b6arrivetime,b6cancelled],[b7name,b7table,b7nopeople,b7arrivetime,b7cancelled],[b8name,b8table,b8nopeople,b8arrivetime,b8cancelled]];

seating_arrangement = [ [1,2,3,4] , [5,6,7,8], [9,10], [11,12], [13,14,15,16], [17,18,19,20], [21,22], [23,24] ];
```

```
%donald

# This info is the DONALD script for the visualisation of the restaurant

viewport RESTVIEW

point R_bottomleft,R_topright,R_frontdoor,R_kitchendoor
line R_front_entrance,R_kitchen_entrance
rectangle R
R_bottomleft = {(100-R_width!) div 2,(100-R_height!) div 2}
R_topright = R_bottomleft+{R_width!,R_height!}
R = rectangle(R_bottomleft,R_topright)
R_frontdoor = R_topright-{0,R_height! div 2}
R_kitchendoor = R_bottomleft+{0,R_height! div 2}
R_front_entrance = [R_frontdoor-{0,R_doorsize! div 2},R_frontdoor+{0,R_doorsize! div 2}]
R_kitchen_entrance = [R_kitchendoor-{0,R_doorsize! div 2},R_kitchendoor+{0,R_doorsize!
div 2}]
?A_R_front_entrance is “color=black,linewidth=5”;
?A_R_kitchen_entrance is “color=black,linewidth=5”;

point
table1_topleft,table2_topleft,table3_topleft,table4_topleft,table5_topleft,table6_topleft,table7
_topleft,table8_topleft
rectangle
resttable1,resttable2,resttable3,resttable4,resttable5,resttable6,resttable7,resttable8
label
table1_label,table2_label,table3_label,table4_label,table5_label,table6_label,table7_label,table8_l
abel

table1_topleft = R_bottomleft+{10,10}
resttable1 = rectangle(table1_topleft,table1_topleft+{R_tablewidth4!,R_tableheight4!})
table1_label = label(“Table1”,table1_topleft+{R_tablewidth4! div 2,-15})
table2_topleft = R_bottomleft+{30,10}
resttable2 = rectangle(table2_topleft,table2_topleft+{R_tablewidth4!,R_tableheight4!})
table2_label = label(“Table2”,table2_topleft+{R_tablewidth4! div 2,-15})
table3_topleft = R_bottomleft+{50,10}
resttable3 = rectangle(table3_topleft,table3_topleft+{R_tablewidth2!,R_tableheight2!})
table3_label = label(“Table3”,table3_topleft+{R_tablewidth2! div 2,-15})
table4_topleft = R_bottomleft+{65,10}
resttable4 = rectangle(table4_topleft,table4_topleft+{R_tablewidth2!,R_tableheight2!})
table4_label = label(“Table4”,table4_topleft+{R_tablewidth2! div 2,-15})
table5_topleft = R_bottomleft+{10,R_height!-10-R_tableheight4!}
resttable5 = rectangle(table5_topleft,table5_topleft+{R_tablewidth4!,R_tableheight4!})
table5_label = label(“Table5”,table5_topleft+{R_tablewidth4! div 2,R_tableheight4!+15})
table6_topleft = R_bottomleft+{30,R_height!-10-R_tableheight4!}
resttable6 = rectangle(table6_topleft,table6_topleft+{R_tablewidth4!,R_tableheight4!})
table6_label = label(“Table6”,table6_topleft+{R_tablewidth4! div 2,R_tableheight4!+15})
table7_topleft = R_bottomleft+{50,R_height!-10-R_tableheight2!}
resttable7 = rectangle(table7_topleft,table7_topleft+{R_tablewidth2!,R_tableheight2!})
table7_label = label(“Table7”,table7_topleft+{R_tablewidth2! div 2,R_tableheight2!+15})
table8_topleft = R_bottomleft+{65,R_height!-10-R_tableheight2!}
resttable8 = rectangle(table8_topleft,table8_topleft+{R_tablewidth2!,R_tableheight2!})
table8_label = label(“Table8”,table8_topleft+{R_tablewidth2! div 2,R_tableheight2!+15})

# restaurant seats represented as circles which will be filled when occupied

point pseat1,pseat2,pseat3,pseat4 ,pseat5,pseat6,pseat7,pseat8, pseat9,pseat10,
pseat11,pseat12
point pseat13,pseat14,pseat15,pseat16 ,pseat17,pseat18,pseat19,pseat20 ,pseat21,pseat22
,pseat23,pseat24
```

```
circle seat1,seat2,seat3,seat4 ,seat5,seat6,seat7,seat8, seat9,seat10, seat11,seat12
circle seat13,seat14,seat15,seat16 ,seat17,seat18,seat19,seat20 ,seat21,seat22 ,seat23,seat24

pseat1 = table1_topleft+{3,-5}
pseat2 = table1_topleft+{R_tablewidth4!-3,-5}
pseat3 = table1_topleft+{3,R_tableheight4!+5}
pseat4 = table1_topleft+{R_tablewidth4!-3,R_tableheight4!+5}

pseat5 = table2_topleft+{3,-5}
pseat6 = table2_topleft+{R_tablewidth4!-3,-5}
pseat7 = table2_topleft+{3,R_tableheight4!+5}
pseat8 = table2_topleft+{R_tablewidth4!-3,R_tableheight4!+5}

pseat9 = table3_topleft+{R_tablewidth2! div 2,-5}
pseat10 = table3_topleft+{R_tablewidth2! div 2,R_tableheight2!+5}

pseat11 = table4_topleft+{R_tablewidth2! div 2,-5}
pseat12 = table4_topleft+{R_tablewidth2! div 2,R_tableheight2!+5}

pseat13 = table5_topleft+{3,-5}
pseat14 = table5_topleft+{R_tablewidth4!-3,-5}
pseat15 = table5_topleft+{3,R_tableheight4!+5}
pseat16 = table5_topleft+{R_tablewidth4!-3,R_tableheight4!+5}

pseat17 = table6_topleft+{3,-5}
pseat18 = table6_topleft+{R_tablewidth4!-3,-5}
pseat19 = table6_topleft+{3,R_tableheight4!+5}
pseat20 = table6_topleft+{R_tablewidth4!-3,R_tableheight4!+5}

pseat21 = table7_topleft+{R_tablewidth2! div 2,-5}
pseat22 = table7_topleft+{R_tablewidth2! div 2,R_tableheight2!+5}

pseat23 = table8_topleft+{R_tablewidth2! div 2,-5}
pseat24 = table8_topleft+{R_tablewidth2! div 2,R_tableheight2!+5}

seat1 = circle(pseat1,3)
seat2 = circle(pseat2,3)
seat3 = circle(pseat3,3)
seat4 = circle(pseat4,3)
seat5 = circle(pseat5,3)
seat6 = circle(pseat6,3)
seat7 = circle(pseat7,3)
seat8 = circle(pseat8,3)
seat9 = circle(pseat9,3)
seat10 = circle(pseat10,3)
seat11 = circle(pseat11,3)
seat12 = circle(pseat12,3)
seat13 = circle(pseat13,3)
seat14 = circle(pseat14,3)
seat15 = circle(pseat15,3)
seat16 = circle(pseat16,3)
seat17 = circle(pseat17,3)
seat18 = circle(pseat18,3)
seat19 = circle(pseat19,3)
seat20 = circle(pseat20,3)
seat21 = circle(pseat21,3)
seat22 = circle(pseat22,3)
seat23 = circle(pseat23,3)
seat24 = circle(pseat24,3)
```

```
%eden

include("NEWrestwindows.s");

%donald
viewport DONALD
%eden

proc occupyseats {
auto i;
 for (i=1; i<=$#;i++) {
  execute("A_seat"//str($[i])//" is \"color="//str(occupycolor)//",fill=solid\";");
 }
}

proc leaveseats {
auto i;

 for (i=1; i<=$#;i++) {
  execute("A_seat"//str($[i])//" is \"color="//str(basecolor)//",fill=hollow\";");
 }
}

func totime {
para c;
auto i,j;

i = c / 60;
j = c % 60;

ctime = strcat(i,":",j);

return ctime;
}

/* displays the current time in a window */

proc clocking : clock,stopClock
{
auto i;

 if (clock % 60 > 9)
 {clock_string = strcat("\n\n ",str(7+(clock / 60)),":",str(clock % 60));}
 else
 {clock_string = strcat("\n\n ",str(7+(clock / 60)),":0",str(clock % 60));}

 if (!stopClock)
 {potential_arrive_time = 0; todo("clock++;");}

 for (i=1; i<=delay; i++)
 {;}

 if (clock >= endclock)
 {todo("stopClock = TRUE;");}
}

/* for customers entering the restaurant, if the time is correct by their booking (i.e in
booking_list)
   then they enter the restaurant and occupy their seats at a particular table */
```

```
proc customers_whohavebooked_entering : clock
{
auto i,j;

 for (i=1; i<=booking_list#; i++)
 {
  if ((booking_list[i][4]==clock)&&(booking_list[i][5]==FALSE)) /* i.e time to enter and not
cancelled*/
  {
   append customers_in_restaurant,booking_list[i];
   for (j=1; j<=booking_list[i][3]; j++)
   {
occupyseats(seating_arrangement[booking_list[i][2]][j]);
   }
  }
 }
}

/*
This function generates the amount of money that a group spends in the restaurant. It is this
function which is specified by the modeller at the moment which will generate the patterns in
data explorer. change this function to change the pattern of data for the restaurant. At the
moment this function depends on the number of people in the group perturbed by a small ran-
dom factor, and then takes account of the fluctuations spent by groups of differing sizes during
the evening. Aiming to get the following relationships, bigger groups spend more earlier,
smaller groups spend more later...
*/
/* currently takes no look at the amount of time in restaurant */

func moneyspent
{
 para timeinrestaurant,arrivetime,numberinparty;
 auto x,y,result;

 x = ((numberinparty*15)-10)+(random(numberinparty*8)-(numberinparty*4));

 if (numberinparty==2)
 {
  x = x + random(arrivetime/15)-random(arrivetime/30);
 }
 else
 {
  x = x + random((endclock-arrivetime)/15)-random((endclock-arrivetime)/30);
 }

 result = x;
 return result;
}

/* need to non-deterministically decide when customers are going to leave the restaurant */
proc customers_leaving : clock
{
 auto i,j,r;
 /* customers_in_restaurant will be of form [name, table_no, number of people, arrival time
*/
 if (customers_in_restaurant#>0)
 {
 i = 1;
 while (i<=customers_in_restaurant#)
 { /* since deleting elements - check inserted below for this */
```

```
      if (customers_in_restaurant==@) {break;}
                                          if        (((i<=customers_in_restaurant#)        &&
(customers_in_restaurant[i][4]+average_meal_time<clock))||(customers_in_restaurant[i][5]==
TRUE))
      {
       for (j=1; j<=customers_in_restaurant[i][3]; j++)
       {
   leaveseats(seating_arrangement[customers_in_restaurant[i][2]][j]);
       }
       writeln("Customer ",customers_in_restaurant[1]," is leaving the restaurant ");

       /* show the customer leaving the restaurant, manager can accept or ignore their data */
       _clwt = "Departure information";
       cleavewinnamefield_setText(strcat(str(customers_in_restaurant[i][1])," "));
       cleavewinnopeoplefield_setText(strcat(str(customers_in_restaurant[i][3])," "));
       cleavewinarrivetimefield_setText(strcat(str(customers_in_restaurant[i][4])," "));
       cleavewinallocatedtablefield_setText(strcat(str(customers_in_restaurant[i][2])," "));
       cleavewindeparttimefield_setText(strcat(str(clock)," "));
                                    cleavewinmoneyspentfield_setText(strcat(str(moneyspent(clock-
customers_in_restaurant[i][4],customers_in_restaurant[i][4],customers_in_restaurant[i][3]))," "));

       allow_clock++;

       writeln(customers_in_restaurant,i);
       delete customers_in_restaurant,i;
       writeln(customers_in_restaurant);
       stopClock = TRUE;

      }
      i++;
     }
    }
   }


   /* draw timeline in the booking info window at the bottom */

   %donald

   viewport TTABLEDISPLAY

   #Utilised across viewports - hence shouldn't define in a viewport
   int blockheight,blockspacing
   blockspacing = 10
   blockheight      =      (trunc(ttableheight!)-40-(R_numberoftables!*blockspacing))      div
(R_numberoftables!)

   point tp,timeline_topleft
   label timelabel
   line currenttimevertline
   line bookingtimevertline
   rectangle timeline
   timeline_topleft = {50,10}
   tp = {25,20}
   timelabel = label("Time",tp)
   timeline = rectangle(timeline_topleft-{5,0},timeline_topleft+{ttablewidth!-100+5,20})
   ?A_timeline is "color=cyan,fill=solid";
```

```
    currenttimevertline=[timeline_topleft+{(clock!*(ttablewidth!-100))div(end-
clock!),0},timeline_topleft+{(clock!*(ttablewidth!-100))div(endclock!),20+(blockspac-
ing*R_numberoftables!)+(blockheight*R_numberoftables!)}]
    ?A_currenttimevertline is “color=red,linewidth=2”;

    bookingtimevertline=[timeline_topleft+{(potential_arrive_time!*(ttablewidth!-100))div(end-
clock!),0},timeline_topleft+{(potential_arrive_time!*(ttablewidth!-100))div(end-
clock!),20+(blockspacing*R_numberoftables!)+(blockheight*R_numberoftables!)}]
    ?A_bookingtimevertline is “color=blue,linewidth=2”;

    %eden

    /* because of the problems with donald declarations add the hours to the timeline using
execute and trigger each time the end
    time of the restaurant changes */

    proc puthours : endclock
    {
    auto i;

     for (i=0; i<=endclock / 60; i++)
     {
      execute(“%donald
       viewport TTABLEDISPLAY
       point hour”//str(i+starthour)//”
       label hour”//str(i+starthour)//”label
         hour”//str(i+starthour)//” = {50+(“//str(i)//”*((ttablewidth!-100) div (endclock! div
60))) ,20}
                hour”//str(i+starthour)//”label  =  label(\””//str(i+starthour)//”\”,hour”//
str(i+starthour)//”)
       “);
     }
    }

    proc showtablesintimetable : R_numberoftables
    {
    auto i,j;

     for (i=1; i<=R_numberoftables; i++)
     {
      execute(“%donald
       viewport TTABLEDISPLAY
       point ptable”//str(i)//”
        rectangle table”//str(i)//”
       label table”//str(i)//”label
        ptable”//str(i)//”=timeline_topleft+{0,20+(blockspacing*”//str(i)//”)+(blockheight*”/
/str(i-1)//”)}
            table”//str(i)//”  =  rectangle(ptable”//str(i)//”,ptable”//str(i)//”+{ttablewidth!-
100,blockheight})
        table”//str(i)//”label=label(\”Table”//str(i)//”\”,tp+{0,20+(blockspacing*”//str(i)//
”)+(blockheight*”//str(i-1)//”)})
       “);
     }
    }

    proc showbookings : booking_list
    {
    auto i,j;

     for (i=1; i<=booking_list#; i++)
```

```
    {
     /* `id` turns the string id into an identifier */
     if (`"b" / / str(i) / / "cancelled"` == FALSE)
     {
      execute("%donald
       viewport TTABLEDISPLAY
       point pcust" / / str(i) / / "start,pcust" / / str(i) / / "end
       rectangle cust" / / str(i) / / "rect
       label cust" / / str(i) / / "label
         pcust" / / str(i) / / "start = timeline_topleft+{b" / / str(i) / / "arrivetime!*(ttablewidth!-100)
div endclock!,20+(blockspacing*b" / / str(i) / / "table!)+(blockheight*(b" / / str(i) / / "table!-1))}
       pcust" / / str(i) / / "end = pcust" / / str(i) / / "start + {booking_length!*(ttablewidth!-100) div
endclock!,blockheight}
        cust" / / str(i) / / "rect = rectangle(pcust" / / str(i) / / "start+{0,1},pcust" / / str(i) / / "end+{0,-
1})
          cust" / / str(i) / / "label = label(b" / / str(i) / / "name!,(pcust" / / str(i) / / "start+pcust" / /
str(i) / / "end) div 2)
       ");

       execute("%eden
        A_cust" / / str(i) / / "rect is
   ((_pcust" / / str(i) / / "start[2] < _timeline_topleft[2]+(clock*(ttablewidth-100)) / endclock)
   && (_pcust" / / str(i) / / "end[2] > _timeline_topleft[2]+(clock*(ttablewidth-100)) / endclock))
   ? \"outlinecolor=green\"
   : \"color=grey,fill=hollow\";

      A_cust" / / str(i) / / "label is
   ((_pcust" / / str(i) / / "start[2] < _timeline_topleft[2]+(clock*(ttablewidth-100)) / endclock)
   && (_pcust" / / str(i) / / "end[2] > _timeline_topleft[2]+(clock*(ttablewidth-100)) / endclock))
   ? \"color=green\"
   : \"color=black\";
     ");
     }
     else
     {
      execute("%donald
       ?A_cust" / / str(i) / / "rect is \"color=grey,outlinecolor=grey,fill=hollow\";
       ?A_cust" / / str(i) / / "label is \"color=grey\";
      ");
     }
    }
   }

   /* Now the routines for accepting / rejecting a new booking */

   func choosetable
   {
    para name,people,arrivetime;
    auto tableno,i,j;

    possibletables = [];
    for (i=1; i<=R_numberoftables; i++)
    {
    append possibletables,i;
    }

   /* check capacity of table with people in party */
    people = int(people); arrivetime=int(arrivetime);

    for (i=1; i<=R_numberoftables; i++) {
```

```
      /* writeln(R_tablecapacity[i],” “,people,” “,possibletables[i],i); */
      if (R_tablecapacity[i] < people)
      {
    possibletables[i] = -1;
    /* writeln(“fred”); */
    }
    else
    {
    /* writeln(“wilma”); */
    }
     }

     /* writeln(possibletables); */

    /* now check we can fit their booking in where they want */
     for (i=1; i<=R_numberoftables; i++)
     {
      for (j=1; j<=booking_list#; j++)
      {
       if ((booking_list[j][2]==i)&&(booking_list[j][5]==FALSE)) /*i.e not cancelled */
       {
    if                                    ((arrivetime>booking_list[j][4]+booking_length)||(arrive-
time+booking_length<booking_list[j][4]))
         {
       /* we’re ok with this as it fits into the slot */
    } else {possibletables[i] = -1;}
        }
      }
     }

    /* possible tables now shows a list which includes some -1’s. */
     writeln(“Possible tables :”,possibletables);
     resulttables = [];
     for (i=1; i<=R_numberoftables; i++)
     {
     if (possibletables[i]>0)
    {
    append resulttables,i;
    }
     }
     writeln(“Result Tables :”,resulttables);

    /* scope for a lot more improvement to the next bit! instead of being random should check
the amount of time it leaves */
    /* blank before or after the closest booking on that table */
     if (resulttables#!=0)
     {
      tableno = random(resulttables#)+1;
      writeln(“Allocated table number “,resulttables[tableno]);
      return resulttables[tableno];

     }
     else
     {
     tableno=0;
    return tableno;
     }
    }
```

```
    Pname is “Anyone”;
    Pnopeople is 1;
    Parrivetime is 0;
    Ptable is choosetable(Pname,Pnopeople,Parrivetime);

    proc acceptabooking : bookwinaccept_mouse_1
    {
     auto nextbooking,selectedtable;
     auto temp,temp2,temp3,resstr;

     if (bookwinaccept_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click
*/

      _bwt = “New booking form”;

      Pname = bookwinnamefield_getText();
      Pnopeople = bookwinnopeoplefield_getText();
      Parrivetime = bookwinarrivetimefield_getText();
      selectedtable = bookwinallocatedtablefield_getText();

      /* remove the new line characters */
      Pname = substr(Pname,1,Pname#-1);
      Pnopeople = substr(Pnopeople,1,Pnopeople#-1);
      Parrivetime = substr(Parrivetime,1,Parrivetime#-1);
      selectedtable = int(selectedtable);

      writeln(“User Selected table is “,selectedtable);

      /*
      Make the following checks :
     That the selectedtable has the correct capacity
      That the selectedtable is available to be booked
      */

        if ((possibletables[selectedtable] > 0) && (selectedtable != 0) && (int(Pnopeople) <=
(R_tablecapacity[selectedtable])))
      {
       writeln(“Booking made by “,Pname,” for a party of “,Pnopeople, “ at a time of “,Parrive-
time);

        /* Now have to decide which table to put it at and then put them there */
        nextbooking = booking_list#+1;
        execute(“
         b”//str(nextbooking)//”name = \””//str(Pname)//”\”;
         b”//str(nextbooking)//”table = “//str(selectedtable)//”;
         b”//str(nextbooking)//”nopeople = “//str(Pnopeople)//”;
         b”//str(nextbooking)//”arrivetime = “//str(Parrivetime)//”;
         b”//str(nextbooking)//”cancelled = FALSE;
         temp = symboldetail(\”booking_list\”)[“//str(3)//”];
         temp2 = substr(temp,”//str(1)//”,temp#-”//str(2)//”);
                                                                    temp3 =
strcat(\”,[b\”,str(booking_list#+1),\”name,b\”,str(booking_list#+1),\”table,b\”,str(booking_li
st#+1),
              \”nopeople,b\”,str(booking_list#+1),\”arrivetime,b\” ,str(booking_list#+1),\”can-
celled]]\” );
         resstr is strcat(temp2,temp3);
         writeln(\”a\”);
         include(\”f.e\”);
        “);
```

```
      bookwinnamefield_setText("");
      bookwinnopeoplefield_setText("");
      bookwinarrivetimefield_setText("");
      bookwinallocatedtablefield_setText("");

      allow_clock--;
      }
      else
      {
       writeln("No space can found in the restaurant.");
      }
    }

    proc rejectabooking : bookwinreject_mouse_1
    {
     auto x,y,z;

     if (bookwinreject_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click
*/

     _bwt = "New booking form";
     writeln("Booking rejected by the manager.");
     x = bookwinnamefield_getText();
     y = bookwinnopeoplefield_getText();
     z = bookwinarrivetimefield_getText();

     writeln("Unreasonable demand made by ",x," for his party of ",y, " at a time of ",z);

     bookwinnamefield_setText("");
     bookwinnopeoplefield_setText("");
     bookwinarrivetimefield_setText("");
     bookwinallocatedtablefield_setText("");

     allow_clock--;
    }

    proc beep
    {
      writeln("\07");
    }

    proc clockstart : starttheclock_mouse_1
    {
      if (starttheclock_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click */

       /* clock is only allowed to continue when zero events are left to deal with */
      if (allow_clock ==0)
      {
      stopClock = FALSE;
      }
      else
      {
      beep();
    writeln("Please deal with events.");

    /* HACK: Reduce the events by 1 so we can continue if we really want to */
    allow_clock--;
      }
    }
```

```
proc beep_on_stop : stopClock
{
if (stopClock == TRUE)
{
beep();
}
}

proc clockstop : stoptheclock_mouse_1
{
  if (stoptheclock_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click */

  stopClock = TRUE;
}

/* To accept a cancellation we need to be able to uniquely identify the customer, so should fill in as */
/* many of the fields required (name, party number, arrival time) */

proc acceptacancellation : cancelwinaccept_mouse_1
{
 auto name,numberinparty,arrivaltime;
 auto matchingcustomers,i;

 if (cancelwinaccept_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click */

 name = cancelwinnamefield_getText();
 numberinparty = cancelwinnopeoplefield_getText();
 arrivaltime = cancelwinarrivetimefield_getText();
 name = substr(name,1,name#-1);
 numberinparty = substr(numberinparty,1,numberinparty#-1);
 arrivaltime = substr(arrivaltime,1,arrivaltime#-1);
 writeln(name," ",numberinparty," ",arrivaltime);
 writeln(booking_list#," = number of bookings");
 matchingcustomers = [];
 for (i=1;i<=booking_list#; i++)
 {
   writeln(booking_list[i][1],name,booking_list[i][3],numberinparty,booking_list[i][4],arrivaltime);
  if (numberinparty==booking_list[i][3]) {writeln("A OK");}
     if  ((name==booking_list[i][1])&&(numberinparty==str(booking_list[i][3]))&&(arrivaltime==str(booking_list[i][4])))
  {
   append matchingcustomers,i;
  }
 }
 writeln(matchingcustomers);
 if (matchingcustomers#==0)
 {
  cancelwininfostring = "\nNo matching customers";
  writeln("No cancellation - incorrect data entered");
 }
 if (matchingcustomers#>1)
 {
  cancelwininfostring = "\nNo unique customer";
  writeln("No cancellation - incorrect data entered");
 }
   if  (matchingcustomers#==1)  {execute("b"//str(matchingcustomers[1])//"cancelled = TRUE;");}
```

```
    cancelwinnamefield_setText("");
    cancelwinnopeoplefield_setText("");
    cancelwinarrivetimefield_setText("");

    allow_clock--;

    _cwt = "Cancellation form";
   }

   proc customer_leaving_restaurant_usedata : cleavewinaccept_mouse_1
   {
    auto name,partysize,arrtime,tableno,deptime,moneyspent;

    if (cleavewinaccept_mouse_1[2] != 4) { return; } /* Make sure we only accept the down click
*/
    _clwt = "Departure information";

    name = cleavewinnamefield_getText();
    partysize = cleavewinnopeoplefield_getText();
    arrtime = cleavewinarrivetimefield_getText();
    tableno = cleavewinallocatedtablefield_getText();
    deptime = cleavewindeparttimefield_getText();
    moneyspent = cleavewinmoneyspentfield_getText();

    /* Remove trailing new-line character */
    name = substr(name,1,name#-1);
    partysize = substr(partysize,1,partysize#-1);
    arrtime = substr(arrtime,1,arrtime#-1);
    tableno = substr(tableno,1,tableno#-1);
    deptime = substr(deptime,1,deptime#-1);
    moneyspent = substr(moneyspent,1,moneyspent#-1);

      append recorded_customer_data,[name,partysize,tableno,arrtime,int(deptime)-int(arr-
time),moneyspent];

    /* Clear fields now they have been accepted */
    cleavewinnamefield_setText("");
    cleavewinnopeoplefield_setText("");
    cleavewinarrivetimefield_setText("");
    cleavewinallocatedtablefield_setText("");
    cleavewindeparttimefield_setText("");
    cleavewinmoneyspentfield_setText("");

    allow_clock--;
   }

   func random
   {
   para x;
   return rand(x)%x;
   }

   /*
   Pre-defined events
   Fields are :
   Time of event
   Type of event {ENTER,TELEPHONE,CANCEL}
   Name of booker
   Number of people
   Time to book          */
```

```
customer_events is [
[30,"ENTER","Joy","3","120"],
[40,"TELEPHONE","Russ","2","150"],
[50,"CANCEL","Russ","2","150"]
];

proc customer_enters_restaurant : clock
{
 /* Find events that should occur at this time */
 auto i,current_event;

 for (i=1; i<=customer_events#; i++)
 {
 current_event = customer_events[i];
if (current_event[1] == clock && current_event[2]=="ENTER")
{
_bwt = "Customer Enters Restaurant";
bookwinnamefield_setText(strcat(current_event[3]," "));
bookwinnopeoplefield_setText(strcat(current_event[4]," ")); /* add blank space cause this
needs chopping if entered into text box */
bookwinarrivetimefield_setText(strcat(current_event[5]," "));

Pname = current_event[3];
Pnopeople = current_event[4];
Parrivetime = current_event[5];
bookwinallocatedtablefield_setText(strcat(str(Ptable)," "));

allow_clock++;

potential_arrive_time=int(Parrivetime);

stopClock = TRUE;
}
 }
}

proc telephone_booking_restaurant : clock
{
 /* Find events that should occur at this time */
 auto i,current_event;

 for (i=1; i<=customer_events#; i++)
 {
 current_event = customer_events[i];
if (current_event[1] == clock && current_event[2]=="TELEPHONE")
{
_bwt = "Telephone Enquiry";
bookwinnamefield_setText(strcat(current_event[3]," "));
bookwinnopeoplefield_setText(strcat(current_event[4]," ")); /* add blank space cause this
needs chopping if entered into text box */
bookwinarrivetimefield_setText(strcat(current_event[5]," "));

Pname = current_event[3];
Pnopeople = current_event[4];
Parrivetime = current_event[5];
bookwinallocatedtablefield_setText(strcat(str(Ptable)," "));

potential_arrive_time=int(Parrivetime);

allow_clock++;
```

```
    stopClock = TRUE;
    }
     }
    }

    proc telephone_cancellation_restaurant : clock
    {
      /* Find events that should occur at this time */
      auto i,current_event;

      for (i=1; i<=customer_events#; i++)
      {
      current_event = customer_events[i];
    if (current_event[1] == clock && current_event[2]==”CANCEL”)
    {
    _cwt = “Telephone Cancellation”;
    cancelwinnamefield_setText(strcat(current_event[3],” “));
    cancelwinnopeoplefield_setText(strcat(current_event[4],” “)); /* add blank space cause this
needs chopping if entered into text box */
    cancelwinarrivetimefield_setText(strcat(current_event[5],” “));

    allow_clock++;

    stopClock = TRUE;
    }
     }
```

/*NEWrestwindows.s*/

```
%donald

# NOW put this restaurant into a scout window

viewport RESTTITLE
label resttitlestring
resttitlestring = label("View of restaurant",{restwidth! div 2,titleheight! div 2})

%scout

integer restwidth,restheight,titleheight,bw;
point base;
string rest_string,borderColor;
base = {20,50};
restwidth = 500;
restheight = 300;
titleheight = 15;
bw = 5;
borderColor = "Maroon";

window restTitleBar = {
      type: DONALD,
      pict : "RESTTITLE",
      box:[base-{0,15},base+{restwidth,0}],
      xmin :0,
      ymin :0,
      xmax :restwidth,
      ymax : titleheight,
      border: 1,
      fgcolor: "white",
      bgcolor: borderColor,
      sensitive: MOTION
};

window restdisp = {
            type: DONALD,
 pict : "RESTVIEW",
 box:[base,base+{restwidth,restheight}],
 xmin : 0,
 ymin : 0,
 xmax : 100,
 ymax : 100,
            bgcolor : "grey",
            border: 1,
            relief: "raised",
            sensitive : ON
};

window restBack = {
      type: TEXT,
      string: "",
 frame:([base-{bw,15+bw},base+{bw+restwidth,bw+restheight}]),
      bgcolor: borderColor,
      border: 1
};
```

```
%donald
viewport CLOCKTITLE
label clocktitlestring
clocktitlestring = label("Clock",{clockwidth! div 2,titleheight! div 2})

%scout

integer clockwidth,clockheight;
point base_2;
string clock_string;
base_2 = base+{restwidth+50,0};
clockwidth = 150;
clockheight = 50;

window clockTitleBar = {
      type: DONALD,
      pict : "CLOCKTITLE",
      box:[base_2-{0,15},base_2+{clockwidth,0}],
      xmin :0,
      ymin :0,
      xmax :clockwidth,
      ymax : titleheight,
      border: 1,
      fgcolor: "white",
      bgcolor: borderColor,
      sensitive: MOTION
};
window clockdisp = {
            type: TEXT,
 frame:([base_2,base_2+{clockwidth,clockheight}]),
            string : clock_string,
            bgcolor : "grey",
            border: 1,
            relief: "raised",
            sensitive : ON
};

window clockBack = {
      type: TEXT,
      string: "",
frame:([base_2-{bw,15+bw},base_2+{bw+clockwidth,bw+clockheight}]),
      bgcolor: borderColor,
      border: 1
};

point base_4;
base_4 = base_2 + {50,0};
string stoptheclockcolor,starttheclockcolor;

window stoptheclock = {
type : TEXT
string : "\nStop clock",
frame :([base_4,base_4+{100,clockheight/2}]),
bgcolor : stoptheclockcolor,
      border : 1,
sensitive : ON
};

point base_5;
base_5 = base_4+{0,clockheight/2};
```

```
window starttheclock = {
type : TEXT
string : "\nStart clock",
frame :([base_5,base_5+{100,clockheight/2}]),
bgcolor : starttheclockcolor,
      border : 1,
sensitive : ON
};

%eden
stoptheclockcolor is (stopClock==TRUE) ? "red" : DFbgcolor;
starttheclockcolor is (stopClock==FALSE) ? "green" : DFbgcolor;

%scout

# Booking timetable window

integer ttablewidth,ttableheight;
point base_3;
base_3 = {20,50+restheight+50};
ttablewidth = 500;
ttableheight = 300;

window ttabledisp = {
            type: DONALD,
            box:[base_3,base_3+{ttablewidth,ttableheight}],
            pict: "TTABLEDISPLAY",
            xmin: 0,
            xmax: ttablewidth,
            ymin: 0,
            ymax: ttableheight,
            bgcolor : "grey",
            bdcolor : borderColor,
            border: 1,
            relief: "raised",
            sensitive : MOTION
};

window ttableBack = {
      type: TEXT,
      string: "",
frame:([base_3-{bw,15+bw},base_3+{bw+ttablewidth,bw+ttableheight}]),
      bgcolor: borderColor,
      border: 1
};

%donald
viewport TTABLETITLE
label ttablestring
ttablestring = label("Booking information",{ttablewidth! div 2,titleheight! div 2})

%scout

window ttableTitleBar = {
      type: DONALD,
      pict : "TTABLETITLE",
      box:[base_3-{0,15},base_3+{ttablewidth,0}],
      xmin :0,
      ymin :0,
```

```
        xmax :ttablewidth,
        ymax : titleheight,
        border: 1,
        fgcolor: “white”,
        bgcolor: borderColor,
        sensitive: MOTION
    };

    # New booking window

    integer bookwinwidth,bookwinheight;
    string bookwinfields;
    point base_6;
    base_6 = {20+restwidth+50,50+clockheight+50};
    bookwinwidth = 300;
    bookwinheight = 150;
    bookwinfields = “\n Name       : \n\n People      : \n\n Arrival time : \n\n Table number
:”;

    window bookwindisp = {
                type: TEXT,
                frame:([base_6,base_6+{bookwinwidth,bookwinheight}]),
     string : bookwinfields,
                bgcolor : “grey”,
                bdcolor : borderColor,
                border: 1,
                relief: “raised”,
                sensitive : MOTION
    };

    window bookwinBack = {
        type: TEXT,
        string: ““,
    frame:([base_6-{bw,15+bw},base_6+{bw+bookwinwidth,bw+bookwinheight}]),
        bgcolor: borderColor,
        border: 1
    };

    %donald
    viewport BOOKWINTITLE
    label bookwinstring
    char bwt
    bwt = “New booking form”
    bookwinstring = label(bwt,{bookwinwidth! div 2,bookwinheight! div 2})

    %scout

    window bookwinTitleBar = {
        type: DONALD,
        pict : “BOOKWINTITLE”,
        box:[base_6-{0,15},base_6+{bookwinwidth,0}],
        xmin :0,
        ymin :0,
        xmax :bookwinwidth,
        ymax : bookwinheight,
        border: 1,
        fgcolor: “white”,
        bgcolor: borderColor,
        sensitive: MOTION
    };
```

```
string bookwinacceptcolor,bookwinrejectcolor;
bookwinacceptcolor = “green”;
bookwinrejectcolor = “red”;

window bookwinaccept = {
type : TEXT
string : “\n Accept this booking”,
frame :([base_6+{0,120},base_6+{0,bookwinheight}+{bookwinwidth / 2,0}]),
bgcolor : bookwinacceptcolor,
      border : 1,
sensitive : ON
};

window bookwinreject = {
type : TEXT
string : “\n Reject this booking”,
frame      :([base_6+{0,120}+{bookwinwidth      / 2,0},base_6+{0,bookwinheight}+{bookwin-
width,0}]),
bgcolor : bookwinrejectcolor,
      border : 1,
sensitive : ON
};

window bookwinnamefield = {
type : TEXTBOX
frame : ([base_6+{120,10},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};
window bookwinnopeoplefield = {
type : TEXTBOX
frame : ([base_6+{120,37},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};
window bookwinarrivetimefield = {
type : TEXTBOX
frame : ([base_6+{120,64},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};

window bookwinallocatedtablefield = {
type : TEXTBOX
frame : ([base_6+{120,91},{25,1}]),
bgcolor : “DarkGrey”,
relief : “sunken”,
border : 1,
sensitive : ON
};
```

```
    # New cancellation window

    integer cancelwinwidth,cancelwinheight;
    string cancelwinfields;
    point base_7;
    base_7 = base_6+{0,50+bookwinheight};
    cancelwinwidth = 300;
    cancelwinheight = 150;
    cancelwinfields = "\n Name       : \n\n People     : \n\n Arrival time : \n\n Table number
:";

    window cancelwindisp = {
                type: TEXT,
                frame:([base_7,base_7+{cancelwinwidth,cancelwinheight}]),
     string : cancelwinfields,
                bgcolor : "grey",
                bdcolor : borderColor,
                border: 1,
                relief: "raised",
                sensitive : MOTION
    };

    window cancelwinBack = {
          type: TEXT,
          string: "",
    frame:([base_7-{bw,15+bw},base_7+{bw+cancelwinwidth,bw+cancelwinheight}]),
          bgcolor: borderColor,
          border: 1
    };

    %donald
    viewport CANCELWINTITLE
    label cancelwinstring
    char cwt
    cwt = "Cancellation form"
    cancelwinstring = label(cwt,{cancelwinwidth! div 2,cancelwinheight! div 2})

    %scout

    window cancelwinTitleBar = {
          type: DONALD,
          pict : "CANCELWINTITLE",
          box:[base_7-{0,15},base_7+{cancelwinwidth,0}],
          xmin :0,
          ymin :0,
          xmax :cancelwinwidth,
          ymax : cancelwinheight,
          border: 1,
          fgcolor: "white",
          bgcolor: borderColor,
          sensitive: MOTION
    };

    string cancelwinacceptcolor,cancelwininfostring;
    cancelwinacceptcolor = "green";
    cancelwininfostring = "";
```

```
window cancelwinaccept = {
type : TEXT
string : “\n Accept cancellation”,
frame :([base_7+{0,120},base_7+{0,cancelwinheight}+{cancelwinwidth /2,0}]),
alignment : CENTRE,
bgcolor : cancelwinacceptcolor,
      border : 1,
sensitive : ON
};

window cancelwininfo = {
type : TEXT
string : cancelwininfostring,
frame :([base_7+{cancelwinwidth /2,120},base_7+{cancelwinwidth / 2,cancelwin-
height}+{cancelwinwidth/2,0}]),
alignment : CENTRE,
bgcolor : “grey”,
      border : 1,
sensitive : ON
};

window cancelwinnamefield = {
type : TEXTBOX
frame : ([base_7+{120,10},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};
window cancelwinnopeoplefield = {
type : TEXTBOX
frame : ([base_7+{120,37},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};
window cancelwinarrivetimefield = {
type : TEXTBOX
frame : ([base_7+{120,64},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};

window cancelwinallocatedtablefield = {
type : TEXTBOX
frame : ([base_7+{120,91},{25,1}]),
bgcolor : “DarkGrey”,
relief : “sunken”,
border : 1,
sensitive : ON
};
```

```
    # Customer leaves restaurant window

    integer cleavewinwidth,cleavewinheight;
    string cleavewinfields;
    point base_8;
    base_8 = base_7+{0,50+cancelwinheight};
    cleavewinwidth = 300;
    cleavewinheight = 210;
    cleavewinfields = "\n Name      : \n\n People     : \n\n Arrival time : \n\n Table number
: \n\n Depart time  : \n\n Money spent  :";

    window cleavewindisp = {
                type: TEXT,
                frame:([base_8,base_8+{cleavewinwidth,cleavewinheight}]),
     string : cleavewinfields,
                bgcolor : "grey",
                bdcolor : borderColor,
                border: 1,
                relief: "raised",
                sensitive : MOTION
    };

    window cleavewinBack = {
          type: TEXT,
          string: "",
    frame:([base_8-{bw,15+bw},base_8+{bw+cleavewinwidth,bw+cleavewinheight}]),
          bgcolor: borderColor,
          border: 1
    };

    %donald
    viewport CLEAVEWINTITLE
    label cleavewinstring
    char clwt
    clwt = "Departure information"
    cleavewinstring = label(clwt,{cleavewinwidth! div 2,cleavewinheight! div 2})

    %scout

    window cleavewinTitleBar = {
          type: DONALD,
          pict : "CLEAVEWINTITLE",
          box:[base_8-{0,15},base_8+{cleavewinwidth,0}],
          xmin :0,
          ymin :0,
          xmax :cleavewinwidth,
          ymax : cleavewinheight,
          border: 1,
          fgcolor: "white",
          bgcolor: borderColor,
          sensitive: MOTION
    };

    string cleavewinacceptcolor,cleavewinrejectcolor;
    cleavewinacceptcolor = "green";
    cleavewinrejectcolor = "red";
```

```
window cleavewinaccept = {
type : TEXT
string : “\n Accept customer data”,
frame :([base_8+{0,180},base_8+{0,cleavewinheight}+{cleavewinwidth /2,0}]),
alignment : CENTRE,
bgcolor : cleavewinacceptcolor,
      border : 1,
sensitive : ON
};

window cleavewinnamefield = {
type : TEXTBOX
frame : ([base_8+{120,10},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};
window cleavewinnopeoplefield = {
type : TEXTBOX
frame : ([base_8+{120,37},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};
window cleavewinarrivetimefield = {
type : TEXTBOX
frame : ([base_8+{120,64},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};

window cleavewinallocatedtablefield = {
type : TEXTBOX
frame : ([base_8+{120,91},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};

window cleavewindeparttimefield = {
type : TEXTBOX
frame : ([base_8+{120,118},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};
window cleavewinmoneyspentfield = {
type : TEXTBOX
frame : ([base_8+{120,145},{25,1}]),
bgcolor : “grey”,
relief : “sunken”,
border : 1,
sensitive : ON
};
```

display timetablewindow = <ttableTitleBar / ttabledisp / ttableBack>;
display restaurantwindow = <restTitleBar / restdisp / restBack>;
display clockwindow = <clockTitleBar / stoptheclock / starttheclock / clockdisp / clockBack>;
display bookingwindow=<bookwinTitleBar / bookwinaccept / bookwinreject / bookwinnamefield / bookwinnopeoplefield / bookwinarrivetimefield / bookwinallocatedtablefield / bookwindisp / bookwinBack>;
display cancelwindow=<cancelwinTitleBar / cancelwinaccept / cancelwininfo / cancelwinnamefield / cancelwinnopeoplefield / cancelwinarrivetimefield / cancelwinallocatedtablefield / cancelwindisp / cancelwinBack>;
display customerleavewindow=<cleavewinTitleBar / cleavewinaccept / cleavewinnamefield / cleavewinnopeoplefield / cleavewinarrivetimefield / cleavewinallocatedtablefield / cleavewindeparttimefield / cleavewinmoneyspentfield / cleavewindisp / cleavewinBack>;
screen = restaurantwindow & timetablewindow & bookingwindow & cancelwindow & customerleavewindow & clockwindow;