FILE: RW.F90

```
1  program rw
2
3      use global_env
4      use calendar
5      use walker
6      use cell
7
8      implicit none
9
10     integer :: max_events
11     integer :: action, opt1, opt2
12
13     real(DP) :: time, time_limit, time_stat
14     real(DP) :: sigma, rho, xx
15
16     integer, dimension(:,:), allocatable :: p0
17
18     real(DP), dimension(:), allocatable :: rn
19
20     integer :: i, j, k, l, m, n
21     integer :: n_errors
22
23     character(len=10) :: arg
24
25     debug = .false.
26
27     ! The following if statement checks to see whether you supply
           any
28     ! flags to the compiler. The possible flags are "debug" and
29     ! "newseed". The debug flag will output information about the
```

```
30    ! running of the program and will introduce additional stop
31    ! commands. The newseed flag ensures that all the random
        numbers are
32    ! generated from a different seed. This is something that we
        would
33    ! like to turn off during testing.
34
35    if(iargc()>0) then
36       do i=1,iargc()
37          call getarg(i,arg)
38          select case (trim(arg))
39          case ('-debug')
40             debug = .true.
41          case  ('-newseed')
42             call newseed()
43          end select
44       end do
45    end if
46
47    ! Extract the needed data from the configuration file
48
49    open(10,file='config.dat')
50    read(10,*) n_walkers
51    read(10,*) box_size
52    read(10,*) cell_size
53    read(10,*) k_3d, k_2d
54    read(10,*) k_on, k_off
55    read(10,*) time_limit
56    read(10,*) time_stat
57    close(10)
58
```

```
59    ! open log file for output
60
61    open(fmain,file='rw.log')
62
63    write(fmain,*) '*** KMC FRAP *** '
64    write(fmain,*) 'Started at ',time_stamp()
65    write(fmain,'(/a,i5)')  'Number of Random Walkers=',n_walkers
66    write(fmain,'(3(a,i4))') 'Box size=',box_size(X),' x',
         box_size(Y),' x',box_size(Y)
67    write(fmain,'(3(a,i4))') 'Cell size=',cell_size(X),' x',
         cell_size(Y),' x',cell_size(Y)
68    write(fmain,'(2(a,f8.4),a)') 'Transition rate=',k_3d,'(3d)
         ',k_2d,'(2d)'
69    write(fmain,'(2(a,f8.4),a)') 'Transition rate=',k_on,'(on)
         ',k_off,'(off)'
70    write(fmain,'(a,e15.3)') 'Total simulation time=',time_limit
71    write(fmain,*)
72
73    ! open a data file #1
74    write(fmain,*)  'outout: onoff.dat'
75    open(fdat1,file='onoff.dat')
76
77    ! open a data file #2
78    write(fmain,*) 'output: diffusion.dat'
79    open(fdat2,file='diffusion.dat')
80
81    ! open a file for debug information, iff the debug flag is
         issued
82    if(debug) then
83       write(fmain,*) 'output: debug.dat'
84       open(fdbg,file='debug.dat')
```

```
85      endif
86
87      ! set the maximum number of walkers.
88      !(this determines the size of array)
89      max_walkers = 2*n_walkers
90
91      ! set the maximum number of events recorded in the calendar
           at a time.
92      max_events = 2*max_walkers
93
94      ! initialize the event calendar
95      call calendar_init(n_walkers,max_events)
96
97      ! initialize the random walkers
98      call walker_init(n_walkers)
99
100     ! initial position of walkers
101     !(for example uniformally random distribution)
102     allocate(rn(n_walkers)) ! Create a vector for the random
           numbers
103     call random_number(rn)
104     walker_pos(1:n_walkers,X) = ceiling(rn*box_size(X))
105     call random_number(rn)
106     walker_pos(1:n_walkers,Y) = ceiling(rn*box_size(Y))
107     call random_number(rn)
108     walker_pos(1:n_walkers,Z) = ceiling(rn*box_size(Z))
109     deallocate(rn) ! Not needed anymore, so there isn't any sense
           in
110     ! letting rn take up any memory
111
112     ! initialize the cell configuration
```

```
113     call cell_init(n_walkers,max_walkers,walker_pos)
114     ! This is used for determining particle collisions
115
116     if(debug) call cell_test(n_walkers)  ! check the consistency
            in cell assignment
117
118     ! predict the jump time for all walkers
119     do i=1,n_walkers
120          call walker_predict_event(i)
121     end do
122
123     ! save the initial position
124     allocate(p0(max_walkers,3))
125     p0 = walker_pos
126
127     ! schedule the first data output
128     call calendar_schedule_event(11,0,0,time_stat)
129     call calendar_schedule_event(12,0,0,time_stat)
130
131     n_collisions = 0
132     time_current = 0.0_DP
133
134     do while (time_current<=time_limit)
135
136        ! find the next event to happen
137        call calendar_find_event(action,opt1,opt2)
138
139        ! execute the event
140        if (action<=10)  then   ! simple jump event
141
142           ! walker 'opt1' jumps
```

```
143          call walker_action(opt1,action)
144          ! predict next jump of walker 'opt1'
145          call walker_predict_event(opt1)
146
147       else if(action<=20) then   ! non-physical events
148
149          select case(action)
150             case(11)
151                !  evaluate surface density
152                sigma = count(walker_pos(:,Z)>box_size(Z))
153                rho = count(walker_pos(:,Z)==box_size(Z))
154                write(fdat1,'(f10.2,2e13.5)') time_current, rho,
                      sigma
155                !  set the next evaluation time
156                call calendar_schedule_event(11,0,0,time_current+
                      time_stat)
157             case(12)
158                !  evaluate mean square displacement
159                xx = real(sum((walker_pos-p0)**2),kind=DP)/real(
                      n_walkers,kind=DP)
160                write(fdat2,'(f10.2,e13.5)') time_current, xx
161                !  set the next evaluation time
162                call calendar_schedule_event(12,0,0,time_current+
                      time_stat)
163             case default
164                ! unknown event
165                write(fmain,'(f10.2,a)') time_current, 'unknown
                      event'
166                ! emergency stop
167                stop
168          end select
```

```
169
170         else if(action>20) then
171            ! binary event (\such as reaction
172            write(fmain,'(f10.2,a)') time_current, 'binary event'
173            ! binary events have not implimented yet.
174         end if
175
176      end do
177
178      write(fmain,'(a,i10)') 'Number of collisions =',n_collisions
179
180      if(debug) call cell_test(n_walkers)
181
182      ! close output files
183
184      close(fdat1)
185      close(fdat2)
186
187      !
188      open(fdat1,file='walkers.dat')
189      ! creation of particles has not been considered yet.
190      write(fdat1,'(3i5)') (walker_pos(i,:), i=1,n_walkers)
191      close(fdat1)
192
193      if(debug) close(fdbg)
194
195      write(fmain,*) '*** Done at ',time_stamp(), '***'
196  end program rw
```