

```

1  /*-----*
2  *
3  * CAMELot/ Lava flows code - Release 1.0 - September, 21 2000
4  *
5  * File:    sciara.cpt
6  *
7  * Authors:
8  *      Gino Mirocle Crisci      UNICAL - Dept. of Earth Sciences
9  *      Salvatore Di Gregorio   UNICAL - Dept. of Mathematics
10 *      Rocco Rongo              UNICAL - Dept. of Earth Sciences
11 *      William Spataro          UNICAL - Dept. of Mathematics
12 *      Donato D'Ambrosio        UNICAL - Dept. of Mathematics
13 *
14 * Purpose:    Simulation code for lava flows
15 *
16 *-----*
17 */
18
19 #include <stdio.h>
20 #include <math.h>
21
22 #define TRUE 1
23 #define FALSE 0
24
25 // Insert global variables here...
26
27 float tf,trate,twidth, areaes, apothem, el;
28 float randS, rorS;
29 FILE *f;
30 char s[256];
31
32 // CA definition...
33
34 cedef
35 {
36     dimension 2;
37
38     region All_AC(:, :, :);
39
40     radius 1;
41
42     state (float altitude,
43           width,
44           max_width,
45           temperature,
46           flow[4];
47           float vent_rate,
48           real,
49           RandS,
50           RorS);
51
52     neighbor news[4]([0,-1]NORTH,[1,0]EAST, [-1,0]WEST,[0,1]SOUTH);
53
54     parameter(prm_cellside 10.0,
55               prm_clock    60.0,
56               prm_admin    0.0,
57               prm_admid    0.0,
58               prm_admax    0.0,
59               prm_tcrat    0.0,
60               prm_tmids    0.0,
61               prm_tsolid   0.0,
62               prm_cool     0.0,
63               prm_rall     0.0,
64               prm_days     1.0,
65               prm_maxstep 5760.0
66             );
67 }//cedef
68
69
70
71 // Prototypes...
72
73 void calc_width();
74 void calc_temperature();
75 void calc_altitude();

```

```

77 float calc_adherence();
78 void calc_flows();
79 void vent();
80
81 //main
82 {
83     switch (step%2) {
84         case 0:
85             calc_flows();
86             break;
87
88         case 1:
89             calc_width();
90             calc_temperature();
91             calc_altitude();
92             if (step < 2 * prm_clock * 24 * prm_days)
93                 vent();
94             break;
95     } //switch
96
97     if (step == (int) prm_maxstep - 1)
98     {
99         if (cell_max_width > 0.0 && cell_real > 0.0)
100             update(cell_RandS, 1.0);
101         if (cell_max_width > 0.0 || cell_real > 0.0)
102             update(cell_RorS, 1.0);
103     }
104 } //niam
105
106 /*****
107  *Calculate lava emission*/
108 /*****
109
110 void vent()
111 {
112     int i;
113
114     if (cell_vent_rate>0.)
115     {
116         // Calculate residual vent lava...
117         tf=0.;
118         for(i=0;i<4;i++)
119             tf+=cell_flow[i];
120
121         // Convert mc/s to m
122         areas=3.*prm_cellside*apothem;
123         trate=(cell_vent_rate*prm_clock)/(areas);
124
125         // Set vent thickness and temperature...
126         twidth=cell_width - tf + trate;
127
128         update(cell_width,twidth);
129         update(cell_temperature,prm_tcrat);
130     }
131 }
132
133 /*****
134  * Calculate new lava width*/
135 /*****
136
137 void calc_width()
138 {
139     int i;
140     float new_width;
141
142     new_width=cell_width;
143     for (i=0; i<4; i++)
144         new_width+=(cell_flow[i]-news[i]_flow[3-i]);
145     update(cell_width,new_width);
146     if (new_width > cell_max_width)
147         update(cell_max_width, new_width);
148 } // calc_width
149
150 /*****
151  * Calculate new lava temperature*/

```

```

153  /*****
154
155  void calc_temperature()
156  {
157      int    i;
158      float  sumh,
159            sumth,
160            new_temp;
161
162      sumh=cell_width;
163      for (i=0; i<4; i++)
164          sumh-=cell_flow[i];
165
166      sumth=sumh*cell_temperature;
167      //weighted average for the first step of the temperature determination
168      for (i=0; i<4; i++){
169          sumh+=news[i]_flow[3-i];
170          sumth+=(news[i]_temperature*news[i]_flow[3-i]);
171      }//for
172      //last step of the temperature determination: temperature loss by irradiation
173      if (sumh>0.){
174          new_temp=sumth/sumh;
175          new_temp/=pow(1.+pow(new_temp,3.)*prm_cool,1./3.);
176          update(cell_temperature,new_temp);
177      }//if
178  } // calc_temperature
179
180  /*****
181  /* Calculate new lava altitude*/
182  /*****
183
184  void calc_altitude()
185  {
186      if ((cell_temperature<=prm_tsolid)&&(cell_width>0)){ //solidification...
187          update(cell_altitude,cell_altitude+cell_width);
188          update(cell_width,0);
189          update(cell_temperature,prm_tsolid);
190      }//if
191  }//calc_altitude
192
193  /*****
194  /* Calculate lava adherence*/
195  /*****
196
197  float calc_adherence()
198  {
199      float  coeff,
200            ad;
201
202      if ((cell_temperature<=prm_tcrat)&&(cell_temperature>=prm_tmids)){
203          coeff=(prm_admid-prm_admin)/(prm_tmids-prm_tcrat);
204          ad=coeff*(cell_temperature-prm_tcrat)+prm_admin;
205      }//if
206      else{
207          coeff=(prm_admax-prm_admid)/(prm_tsolid-prm_tmids);
208          ad=coeff*(cell_temperature-prm_tmids)+prm_admid;
209      }//else
210      return ad;
211  } //calc_adherence
212
213  /*****
214  /* Calculate outgoing flows*/
215  /*****
216
217  void calc_flows()
218  {
219      {
220          char wlog[5]={TRUE,TRUE,TRUE,TRUE,TRUE},
221                elim;
222          int  i,
223                k,
224                kk;
225          float  adherence,
226                wqc[5],
227                distr,
228                qav,

```

```

229         rall;
230
231     adherence=calc_adherence();
232     if ((adherence<cell_width)&&(cell_width>0))
233     {
234         wqc[0]=cell_altitude+adherence;
235         for (i=1; i<5; i++)
236             wqc[i]=news[i-1]_width+news[i-1]_altitude;
237         distr=cell_width-adherence;
238
239         // calculate outgoing flows
240         do{
241             elim=FALSE;
242             qav=distr;
243             kk=0;
244             for (k=0; k<5; k++)
245                 if (wlog[k]){
246                     qav+=wqc[k];
247                     kk++;
248                 }//if
249             if (kk!=0)
250                 qav/=kk;
251             for (k=0; k<5; k++)
252                 if ((qav<=wqc[k])&&(wlog[k])){
253                     wlog[k]=FALSE;
254                     elim=TRUE;
255                 }//if
256             }while (elim);
257             for (k=1; k<5; k++)
258                 if (wlog[k]){
259                     update(cell_flow[k-1],(qav-wqc[k])*prm_rall);
260                 }//if
261             else
262                 update(cell_flow[k-1],0.);
263         }//if
264         else
265             for (k=1; k<5; k++)
266                 update(cell_flow[k-1],0.);
267     } //calc_flows
268
269     /***** steering */
270     /* steering */
271     /***** steering */
272
273     steering
274     {
275         if (step == 0)
276         {
277             randS = 0.0;
278             rorS = 0.0;
279
280             apothem=0.866025*prm_cellside;
281
282             f = fopen("param.txt", "r");
283             fscanf(f, "%s", s); fscanf(f, "%s", s); cpt_set_param(&prm_admin,  atof(s));
284             fscanf(f, "%s", s); fscanf(f, "%s", s); cpt_set_param(&prm_admid,  atof(s));
285             fscanf(f, "%s", s); fscanf(f, "%s", s); cpt_set_param(&prm_admax,  atof(s));
286             fscanf(f, "%s", s); fscanf(f, "%s", s); cpt_set_param(&prm_tcrat,  atof(s));
287             fscanf(f, "%s", s); fscanf(f, "%s", s); cpt_set_param(&prm_tmidi,  atof(s));
288             fscanf(f, "%s", s); fscanf(f, "%s", s); cpt_set_param(&prm_tsolid, atof(s));
289             fscanf(f, "%s", s); fscanf(f, "%s", s); cpt_set_param(&prm_cool,   atof(s));
290             fscanf(f, "%s", s); fscanf(f, "%s", s); cpt_set_param(&prm_rall,   atof(s));
291             fclose(f);
292         }
293
294         //compute the fitness and write it on file
295         if (step == (int) prm_maxstep - 1)
296         {
297             randS = region_sum(All_AC, RandS);
298             rorS = region_sum(All_AC, RorS);
299             el = sqrt(randS / rorS);
300             f = fopen("fitness.txt", "w");
301             fprintf(f, "%f\n", el);
302             fclose(f);
303         }
304     }

```