

```

1
2 Supplementary 2A Surface resistant to motility
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <float.h>
7 #include <graphics.h>
8 #include <math.h>
9 #include <conio.h>
10 #include <time.h>
11
12
13 //dimensions must be multiples of 4 for bmp output formatting
14 #define LEFT 0
15 #define RIGHT 120
16 #define TOP 0
17 #define BOTTOM 75 //should be less than right
18
19 //color values
20 #define BLACK 0
21 #define WHITE 255
22 #define RED 100
23 #define GREEN 18
24 #define BLUE 17
25 #define GRAY 200
26 #define ORANGE 150
27 #define YELLOW 30
28 #define BLACKVALUE 0
29 #define GRAYVALUE 0
30 #define REDVALUE 50
31 #define LIGHT 4
32 #define LIGHTR 4
33 #define DARK 1
34 #define DARKR 1
35 #define R 2
36 #define MODMOVE 200
37
38 #define COUNTERMAX 4000
39 #define GRAYSLIME 3000 //starting value of counter in initial spot
40 #define SPREAD1 4
41 #define SPREAD12 1
42 #define SPREAD123 1
43 #define SPREAD1234 1
44
45 #define FILL 20//warning, cellmax
46 #define RIM 3
47
48 #define CELLMAX 1000
49 #define MAXTURN 1000
50
51 //draws a 3x3 pixel
52 void draw_cell(char x, char y, char out)
53 {
54 int e, r;
55 int cx=(x*3+1);
56 int cy=(y*3+1);
57 for(e=-1; e<2; e++)

```

```

1  {
2  for(r=-1; r<2; r++)
3  {
4  putpixel((cx+e),(cy+r), (int) out);
5  }
6  }
7
8  }
9
10
11 void main(void)
12 {
13 //graphics vars
14     int      g_driver, g_mode;
15     int      i, left, top, wide, bottom, deep;
16
17 //counter vars
18     int updatex, updatey;
19     int x, y, a, b;
20 //grids
21     unsigned char occupy[RIGHT+1][BOTTOM+1];
22     int counter[RIGHT+1][BOTTOM+1];
23     unsigned char slime[RIGHT+1][BOTTOM+1];
24 //drop filling one out of x is coverage
25     int fill=FILL;
26     int rim=RIM;
27 //collision detection
28     long turn;
29
30 //cells
31     int cell[CELLMAX][3];
32     long ccount=0;
33     long countcopy=0;
34     long currentcell=0;
35     int cellx, celly;
36     int slimevalue=0;
37     int lightvalue=0;
38
39 //text output
40     char txt[12]="t000000.txt";
41     char * txt1 = txt;
42 //bmp output
43     char bmp[12]="b000000.bmp";
44     char * bmp1 = bmp;
45 //output general
46     FILE * wFile;
47
48
49 //seed rand
50     srand (time(NULL));
51
52 //initial setup
53 //zero arrays for cells
54
55     for (a=0; a<(CELLMAX); a++)
56     {
57         cell[a][0]=0;
58         cell[a][1]=0;

```

```

1      }
2  //
3  //setting up initial drop, zero counter
4      for(x=0; x<(RIGHT); x++)
5          {
6          for(y=0; y<(BOTTOM); y++)
7              {
8              if (((x-(BOTTOM/2))*(x-(BOTTOM/2))+(y-(BOTTOM/2))*(y-(BOTTOM/2)))<(BOTTOM*BOTTOM/9))
9                  {
10                 slime[x][y]=GRAY;
11                 counter[x][y]=(GRAYSLIME);
12                 }
13                 else
14                 {
15                 slime[x][y]=BLACK;
16                 }
17                 }
18                 }
19                 for(x=0; x<(RIGHT); x++)
20                     {
21                     for(y=0; y<(BOTTOM); y++)
22                         {
23                         if ((slime[x][y] ==GRAY && rand()%(fill)==0 ) || (slime[x][y]==GRAY && rand()%(rim)==0 &&
24                         (slime[x-1][y+1]==BLACK ||
25                         slime[x-1][y]==BLACK ||
26                         slime[x-1][y-1]==BLACK ||
27                         slime[x][y+1]==BLACK ||
28                         slime[x][y]==BLACK ||
29                         slime[x][y-1]==BLACK ||
30                         slime[x+1][y+1]==BLACK ||
31                         slime[x+1][y]==BLACK||
32                         slime[x+1][y-1]==BLACK)))
33                             {
34                             occupy[x][y]=GREEN;
35                             cell[ccount][0]=x;
36                             cell[ccount][1]=y;
37                             cell[ccount][2]=0;
38                             ccount++;
39                             }
40                             else
41                             {
42                             occupy[x][y]=BLACK;
43                             }
44                             if(x==0 || y==0 || x==RIGHT-1 || y==BOTTOM-1)
45                                 {
46                                 slime[x][y]=(BLUE);
47                                 }
48                                 counter[x][y]=0;
49                                 }
50                                 }
51
52 //draw initilized
53
54
55 detectgraph(&g_driver, &g_mode); //DRAW
56 initgraph(&g_driver, &g_mode, "..\\bgi"); //DRAW
57
58

```

```

1 //at this point, cells have been placed and counters zeroed
2 //now is the time to run the logic
3
4 for(turn=1; turn<MAXTURN; turn++)
5 {
6 //printf("topofforloop %d, ", turn); //TAG
7
8
9
10 //choosing cells
11 countcopy=(rand()%(ccount));
12 currentcell=(countcopy+1);
13 currentcell=0;
14 while(currentcell < (ccount))
15     {
16 //directionality search
17
18 // 234
19 // 105
20 // 876
21
22 if(cell[currentcell][2]==0)
23     {
24         cell[currentcell][2]=((rand()%8)+1);
25     }
26 if(cell[currentcell][2]==1)
27     {
28         cellx=cell[currentcell][0]-1;
29         celly=cell[currentcell][1];
30         lightvalue=DARK;
31     }
32 if(cell[currentcell][2]==2)
33     {
34         cellx=cell[currentcell][0]-1;
35         celly=cell[currentcell][1]-1;
36         lightvalue=DARKR;
37     }
38 if(cell[currentcell][2]==3)
39     {
40         cellx=cell[currentcell][0];
41         celly=cell[currentcell][1]-1;
42         lightvalue=R;
43     }
44 if(cell[currentcell][2]==4)
45     {
46         cellx=cell[currentcell][0]+1;
47         celly=cell[currentcell][1]-1;
48         lightvalue=LIGHTR;
49     }
50 if(cell[currentcell][2]==5)
51     {
52         cellx=cell[currentcell][0]+1;
53         celly=cell[currentcell][1];
54         lightvalue=LIGHT;
55     }
56 if(cell[currentcell][2]==6)
57     {
58         cellx=cell[currentcell][0]+1;

```

```

1         celly=cell[currentcell][1]+1;
2         lightvalue=LIGHTR;
3         }
4     if(cell[currentcell][2]==7)
5     {
6         cellx=cell[currentcell][0];
7         celly=cell[currentcell][1]+1;
8         lightvalue=R;
9         }
10    if(cell[currentcell][2]==8)
11    {
12        cellx=cell[currentcell][0]-1;
13        celly=cell[currentcell][1]+1;
14        lightvalue=DARKR;
15    }
16    //check for cell to cell contact
17    if(occupy[cellx][celly] != BLACK)
18    {
19        cell[currentcell][2]=0;
20    }
21    else
22    {
23        if(slime[cellx][celly]==GRAY)
24        {
25            slimevalue=GRAYVALUE;
26        }
27        if(slime[cellx][celly]==BLACK)
28        {
29            slimevalue=BLACKVALUE;
30        }
31        if(slime[cellx][celly]==RED)
32        {
33            slimevalue=REDVALUE;
34        }
35        //biasedwalk
36        //printf("bias %d, ", currentcell); //TAG
37
38        if((slimevalue*lightvalue)>(rand()%MODMOVE))
39        {
40            //move
41            occupy[cell[currentcell][0]][cell[currentcell][1]]=BLACK;
42            cell[currentcell][0]=cellx;
43            cell[currentcell][1]=celly;
44            occupy[cellx][celly]=GREEN;
45        }
46        else
47        {
48            //nomove
49            cell[currentcell][2]=0;
50        }
51    }
52    //counter iterations within one radius
53    for(x=(cell[currentcell][0]-1);x<(cell[currentcell][0]+2);x++)
54    {
55        for(y=(cell[currentcell][1]-1);y<(cell[currentcell][1]+2);y++)
56        {
57            ((counter[x][y]))=((counter[x][y]))+SPREAD1;
58        }

```

```

1         }
2
3         //counter iterations within 2 radii
4         for(x=(cell[currentcell][0]-2);x<(cell[currentcell][0]+3);x++)
5         {
6         for(y=(cell[currentcell][1]-2);y<(cell[currentcell][1]+3);y++)
7         {
8         ((counter[x][y])=((counter[x][y])+SPREAD12);
9         }
10        }
11
12        //counter iterations within 3 radii
13        for(x=(cell[currentcell][0]-3);x<(cell[currentcell][0]+4);x++)
14        {
15        for(y=(cell[currentcell][1]-3);y<(cell[currentcell][1]+4);y++)
16        {
17        ((counter[x][y])=((counter[x][y])+SPREAD123);
18        }
19        }
20
21        //counter iterations within 4 radii
22        for(x=(cell[currentcell][0]-4);x<(cell[currentcell][0]+5);x++)
23        {
24        for(y=(cell[currentcell][1]-4);y<(cell[currentcell][1]+5);y++)
25        {
26        ((counter[x][y])=((counter[x][y])+SPREAD1234);
27        }
28        }
29
30        counter[cell[currentcell][0]][cell[currentcell][1]]=COUNTERMAX;
31
32        //    if(currentcell== ccount-1)
33        //        {
34        //            currentcell=0;
35        //        }
36        //    else
37        //        {
38        //            currentcell++;
39        //        }
40    }
41
42
43    //updating slime for draw
44    //printf("beforeslime %d, ", turn); //TAG
45    //getch();//TAG
46
47    // bmp initialize
48    bmp[1]=(char) ((turn/100000)% 10+48) ;
49    bmp[2]=(char) ((turn/10000)% 10+48) ;
50    bmp[3]=(char) ((turn/1000)% 10+48) ;
51    bmp[4]=(char) ((turn/100)% 10+48) ;
52    bmp[5]=(char) ((turn/10)% 10+48) ;
53    bmp[6]=(char) (turn% 10+48) ;
54    wFile = fopen (bmp1 , "w");
55    putc( 66 , wFile);
56    putc( 77 , wFile);
57
58

```

```

1 //putc( -120 , wFile) ; //variable
2 //putc( 70 , wFile) ; //variable
3 //putc( 0 , wFile) ; //variable
4 //putc( 0 , wFile) ; //variable
5 putc( ((BOTTOM*RIGHT*3+55)%256), wFile);
6 putc( ((BOTTOM*RIGHT*3+55)/256 )%256, wFile);
7 putc( (((BOTTOM*RIGHT*3+55)/(256) )/256)%256, wFile);
8 putc( (((((BOTTOM*RIGHT*3+55)/(256) )/256)/256)%256, wFile);
9
10
11 putc( 0 , wFile) ;
12 putc( 0 , wFile) ;
13 putc( 0 , wFile) ;
14 putc( 0 , wFile) ;
15 putc( 54 , wFile) ;
16 putc( 0 , wFile) ;
17 putc( 0 , wFile) ;
18 putc( 0 , wFile) ;
19 putc( 40 , wFile) ;
20 putc( 0 , wFile) ;
21 putc( 0 , wFile) ;
22 putc( 0 , wFile) ;
23
24
25 // putc( 100 , wFile) ; //var len
26 // putc( 0 , wFile) ; //var len
27 // putc( 0 , wFile) ; //var len
28 // putc( 0 , wFile) ; //var len
29 putc( ((RIGHT)%256), wFile);
30 putc( ((RIGHT)/256 )%256, wFile);
31 putc( (((RIGHT)/(256) )/256)%256, wFile);
32 putc( (((((RIGHT)/(256) )/256)/256)%256, wFile);
33
34
35 // putc( 60 , wFile) ; //var wide
36 // putc( 0 , wFile) ; //var wide
37 // putc( 0 , wFile) ; //var wide
38 // putc( 0 , wFile) ; //var wide
39
40 putc( ((BOTTOM)%256), wFile);
41 putc( ((BOTTOM)/256)%256, wFile);
42 putc( (((BOTTOM)/(256) )/256)%256, wFile);
43 putc( (((((BOTTOM)/(256) )/256)/256)%256, wFile);
44
45 putc( 1 , wFile) ;
46 putc( 0 , wFile) ;
47 putc( 24 , wFile) ;
48 putc( 0 , wFile) ;
49 putc( 0 , wFile) ;
50 putc( 0 , wFile) ;
51 putc( 0 , wFile) ;
52 putc( 0 , wFile) ;
53 putc( 0 , wFile) ; //imsize82
54 putc( 0 , wFile) ; //imsize70
55 putc( 0 , wFile) ; //imsize
56 putc( 0 , wFile) ; //imsize
57 putc( 19 , wFile) ;
58 putc( 3 , wFile) ;

```

```

1  putc( 0 , wFile)  ;
2  putc( 0 , wFile)  ;
3  putc( 19 , wFile)  ;
4  putc( 3 , wFile)  ;
5  putc( 0 , wFile)  ;
6  putc( 0 , wFile)  ;
7  putc( 0 , wFile) ;
8  putc( 0 , wFile) ;
9  putc( 0 , wFile) ;
10 putc( 0 , wFile) ;
11 putc( 0 , wFile) ;
12 putc( 0 , wFile) ;
13 putc( 0 , wFile) ;
14 putc( 0 , wFile) ;
15
16
17
18
19
20 for(updatey=0; updatey<(BOTTOM); updatey++)
21 {
22 for(updatex=0; updatex<(RIGHT); updatex++)
23 {
24 if(counter[updatex][updatey]>(COUNTERMAX-1))
25 {
26 slime[updatex][updatey]=RED;
27 counter[updatex][updatey]=COUNTERMAX;
28 }
29 if(occupy[updatex][updatey]>BLACK)
30 {
31 slime[updatex][updatey]=occupy[updatex][updatey];
32 }
33 //to screen
34 draw_cell(updatex, updatey, slime[updatex][updatey]); //DRAW
35
36 //to bmp
37 if(slime[updatex][updatey]==BLACK)
38 {
39 putc( 0 , wFile);
40 putc( 0 , wFile);
41 putc( 0 , wFile);
42 }
43 if(slime[updatex][updatey]==GRAY)
44 {
45 putc( 51 , wFile);
46 putc( 51 , wFile);
47 putc( 51 , wFile);
48 }
49 if(slime[updatex][updatey]==YELLOW)
50 {
51 putc( 0 , wFile);
52 putc( -1 , wFile);
53 putc( -1 , wFile);
54 }
55 if(slime[updatex][updatey]==RED)
56 {
57 putc( 0 , wFile);
58 putc( 0 , wFile);

```



```

1  putc( -1 , wFile);
2  }
3  if(slime[updatex][updatey]==BLUE)
4  {
5  putc( -1 , wFile);
6  putc( 0 , wFile);
7  putc( 0 , wFile);
8  }
9  if(slime[updatex][updatey]==GREEN)
10 {
11 putc( 0 , wFile);
12 putc( -1 , wFile);
13 putc( 0 , wFile);
14 }
15 if(occupy[updatex][updatey]>BLACK)
16 {
17 slime[updatex][updatey]=RED;
18 }
19 }
20 }
21 putc( -1, wFile); //EOF
22
23 fclose (wFile); //close output to bmp
24 draw_cell(RIGHT-1, BOTTOM-1, (char) turn);
25 //to txt
26 //header
27 txt[1]=(char) ((turn/100000)%10+48) ;
28 txt[2]=(char) ((turn/10000)%10+48) ;
29 txt[3]=(char) ((turn/1000)%10+48) ;
30 txt[4]=(char) ((turn/100)%10+48) ;
31 txt[5]=(char) ((turn/10)%10+48) ;
32 txt[6]=(char) (turn%10+48) ;
33 wFile = fopen (txt1 , "w");
34
35 putc( (char) 78, wFile);
36 putc( (char) 85, wFile);
37 putc( (char) 77, wFile);
38 putc( (char) 44, wFile);
39 putc( (char) 88, wFile);
40 putc( (char) 44, wFile);
41 putc( (char) 89, wFile);
42 putc( (char) 44, wFile);
43 putc( txt[0] , wFile);
44 putc( txt[1] , wFile);
45 putc( txt[2] , wFile);
46 putc( txt[3] , wFile);
47 putc( txt[4] , wFile);
48 putc( txt[5] , wFile);
49 putc( txt[6] , wFile);
50 putc( txt[7] , wFile);
51 putc( txt[8] , wFile);
52 putc( txt[9] , wFile);
53 putc( txt[10] , wFile);
54 putc( (char) 13, wFile);
55 putc( (char) 10, wFile);
56
57 //data
58 for(currentcell=0; currentcell<(ccount+1); currentcell++)

```

```

1  {
2  putc ((char)((currentcell/100000)% 10+48), wFile );
3  putc ((char)((currentcell/10000)% 10+48), wFile );
4  putc ((char)((currentcell/1000)% 10+48), wFile );
5  putc ((char)((currentcell/100)% 10+48), wFile );
6  putc ((char)((currentcell/10)% 10+48), wFile );
7  putc ((char)((currentcell/1)% 10+48), wFile );
8  putc( (char) 44, wFile);
9  putc ((char)((cell[currentcell][0]/100000)% 10+48), wFile );
10 putc ((char)((cell[currentcell][0]/10000)% 10+48), wFile );
11 putc ((char)((cell[currentcell][0]/1000)% 10+48), wFile );
12 putc ((char)((cell[currentcell][0]/100)% 10+48), wFile );
13 putc ((char)((cell[currentcell][0]/10)% 10+48), wFile );
14 putc ((char)((cell[currentcell][0]/1)% 10+48), wFile );
15 putc( (char) 44, wFile);
16 putc ((char)((cell[currentcell][1]/100000)% 10+48), wFile );
17 putc ((char)((cell[currentcell][1]/10000)% 10+48), wFile );
18 putc ((char)((cell[currentcell][1]/1000)% 10+48), wFile );
19 putc ((char)((cell[currentcell][1]/100)% 10+48), wFile );
20 putc ((char)((cell[currentcell][1]/10)% 10+48), wFile );
21 putc ((char)((cell[currentcell][1]/1)% 10+48), wFile );
22 putc( (char) 44, wFile);
23 putc ((char)((cell[currentcell][2]/100000)% 10+48), wFile );
24 putc ((char)((cell[currentcell][2]/10000)% 10+48), wFile );
25 putc ((char)((cell[currentcell][2]/1000)% 10+48), wFile );
26 putc ((char)((cell[currentcell][2]/100)% 10+48), wFile );
27 putc ((char)((cell[currentcell][2]/10)% 10+48), wFile );
28 putc ((char)((cell[currentcell][2]/1)% 10+48), wFile );
29 putc( (char) 44, wFile);
30 putc( (char) 13, wFile);
31 putc( (char) 10, wFile);
32 }
33 fclose (wFile); //close output to text
34 //printf("end loop %d, ", turn); //TAG
35 //getch(); //TAG
36 } //end of the turn for() loop
37 draw_cell(0, 0, YELLOW); //DRAW DONE
38 getch();
39 closegraph(); //DRAW
40 //printf("end program %d, ", turn);
41 //getch();
42
43 return;
44
45 }
46
47

```

```
1
2 Supplementary 2B Surface permissive to motility
3
4 #define LEFT 0
5 #define RIGHT 120
6 #define TOP 0
7 #define BOTTOM 75 //should be less than right
8
9 //color values
10 #define BLACK 0
11 #define WHITE 255
12 #define RED 100
13 #define GREEN 18
14 #define BLUE 17
15 #define GRAY 200
16 #define ORANGE 150
17 #define YELLOW 30
18 #define BLACKVALUE 0
19 #define GRAYVALUE 0
20 #define REDVALUE 25
21 #define LIGHT 4
22 #define LIGHTR 4
23 #define DARK 1
24 #define DARKR 1
25 #define R 2
26 #define MODMOVE 200
27
28 #define COUNTERMAX 1000
29 #define GRAYSLIME 50 //starting value of counter in initial spot
30 #define SPREAD1 4
31 #define SPREAD12 1
32 #define SPREAD123 1
33 #define SPREAD1234 1
34
35 #define FILL 20//warning, cellmax
36 #define RIM 3
37
38 #define CELLMAX 2000
39 #define MAXTURN 800
40
41
42
```