Supplementary 2A Surface resistant to motility

```
#include	<stdio.h>
#include	<stdlib.h>
#include	<float.h>
#include	<graphics.h>
#include	<math.h>
#include	<conio.h>
#include	<time.h>


//dimensions must be multiples of 4 for bmp output formatting
#define LEFT 0
#define RIGHT 120
#define TOP 0
#define BOTTOM 75 //should be less than right

//color values
#define BLACK 0
#define WHITE 255
#define RED 100
#define GREEN 18
#define BLUE 17
#define GRAY 200
#define ORANGE 150
#define YELLOW 30
#define BLACKVALUE 0
#define GRAYVALUE 0
#define REDVALUE 50
#define LIGHT 4
#define LIGHTR 4
#define DARK 1
#define DARKR 1
#define R 2
#define MODMOVE 200

#define COUNTERMAX 4000
#define GRAYSLIME 3000  //starting value of counter in initial spot
#define SPREAD1 4
#define SPREAD12 1
#define SPREAD123 1
#define SPREAD1234 1

#define FILL 20//warning, cellmax
#define RIM 3

#define CELLMAX 1000
#define MAXTURN 1000

//draws a 3x3 pixel
void draw_cell(char x, char y, char out)
{
int e, r;
int cx=(x*3+1);
int cy=(y*3+1);
for(e=-1; e<2; e++)
```

```
{
for(r=-1; r<2; r++)
{
putpixel((cx+e),(cy+r), (int) out);
}
}

}


void main(void)
{
//graphics vars
  int           g_driver, g_mode;
  int           i, left, top, wide, bottom, deep;

//counter vars
  int updatex, updatey;
  int x, y, a, b;
//grids
        unsigned char occupy[RIGHT+1][BOTTOM+1];
        int counter[RIGHT+1][BOTTOM+1];
        unsigned char slime[RIGHT+1][BOTTOM+1];
//drop filling one out of x is coverage
     int fill=FILL;
     int rim=RIM;
//collision detection
     long turn;

//cells
        int cell[CELLMAX][3];
        long ccount=0;
        long countcopy=0;
        long currentcell=0;
        int cellx, celly;
        int slimevalue=0;
        int lightvalue=0;

//text output
 char txt[12]="t000000.txt";
 char * txt1 = txt;
//bmp output
 char bmp[12]="b000000.bmp";
 char * bmp1 = bmp;
//output general
 FILE * wFile;


//seed rand
        srand (time(NULL));

//initial setup
//zero arrays for cells

        for (a=0; a<(CELLMAX); a++)
        {
        cell[a][0]=0;
        cell[a][1]=0;
```

```
            }
//
//setting up initial drop, zero counter
      for(x=0; x<(RIGHT); x++)
      {
      for(y=0; y<(BOTTOM); y++)
      {
      if (((x-(BOTTOM/2))*(x-(BOTTOM/2))+(y-(BOTTOM/2))*(y-(BOTTOM/2)))<(BOTTOM*BOTTOM/9))
      {
      slime[x][y]=GRAY;
      counter[x][y]=(GRAYSLIME);
      }
      else
      {
      slime[x][y]=BLACK;
      }
      }
      }
      for(x=0; x<(RIGHT); x++)
      {
      for(y=0; y<(BOTTOM); y++)
      {
      if ((slime[x][y] ==GRAY && rand()%(fill)==0 ) || (slime[x][y]==GRAY && rand()%(rim)==0 &&
      (slime[x-1][y+1]==BLACK ||
       slime[x-1][y]==BLACK ||
       slime[x-1][y-1]==BLACK ||
       slime[x][y+1]==BLACK ||
       slime[x][y]==BLACK ||
       slime[x][y-1]==BLACK ||
       slime[x+1][y+1]==BLACK ||
       slime[x+1][y]==BLACK||
       slime[x+1][y-1]==BLACK)))
     {
      occupy[x][y]=GREEN;
      cell[ccount][0]=x;
      cell[ccount][1]=y;
      cell[ccount][2]=0;
      ccount++;
      }
      else
      {
      occupy[x][y]=BLACK;
      }
      if(x==0 || y==0 || x==RIGHT-1 || y==BOTTOM-1)
      {
      slime[x][y]=(BLUE);
      }
      counter[x][y]=0;
      }
      }

//draw initilized


detectgraph(&g_driver, &g_mode);    //DRAW
initgraph(&g_driver, &g_mode, "..\\bgi");  //DRAW

```

```
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;
```

```
                celly=cell[currentcell][1]+1;
                lightvalue=LIGHTR;
                }
        if(cell[currentcell][2]==7)
                {
                cellx=cell[currentcell][0];
                celly=cell[currentcell][1]+1;
                lightvalue=R;
                }
        if(cell[currentcell][2]==8)
                {
                cellx=cell[currentcell][0]-1;
                celly=cell[currentcell][1]+1;
                lightvalue=DARKR;
                }
        //check for cell to cell contact
        if(occupy[cellx][celly] != BLACK)
                {
                cell[currentcell][2]=0;
                }
        else
                {
                if(slime[cellx][celly]==GRAY)
                        {
                        slimevalue=GRAYVALUE;
                        }
                if(slime[cellx][celly]==BLACK)
                        {
                        slimevalue=BLACKVALUE;
                        }
                if(slime[cellx][celly]==RED)
                        {
                        slimevalue=REDVALUE;
                        }
                //biasedwalk
//printf("bias  %d, ", currentcell); //TAG

                if((slimevalue*lightvalue)>(rand()%MODMOVE))
                        {
                        //move
                        occupy[cell[currentcell][0]][cell[currentcell][1]]=BLACK;
                        cell[currentcell][0]=cellx;
                        cell[currentcell][1]=celly;
                        occupy[cellx][celly]=GREEN;
                        }
                else
                        {
                        //nomove
                        cell[currentcell][2]=0;
                        }
                }
                //counter iterations within one radius
                for(x=(cell[currentcell][0]-1);x<(cell[currentcell][0]+2);x++)
                {
                for(y=(cell[currentcell][1]-1);y<(cell[currentcell][1]+2);y++)
                {
                ((counter[x][y]))=((counter[x][y]))+SPREAD1;
                }
```

```
                        }

                        //counter iterations within 2 radii
                        for(x=(cell[currentcell][0]-2);x<(cell[currentcell][0]+3);x++)
                        {
                        for(y=(cell[currentcell][1]-2);y<(cell[currentcell][1]+3);y++)
                        {
                        ((counter[x][y]))=((counter[x][y]))+SPREAD12;
                        }
                        }

                        //counter iterations within 3 radii
                        for(x=(cell[currentcell][0]-3);x<(cell[currentcell][0]+4);x++)
                        {
                        for(y=(cell[currentcell][1]-3);y<(cell[currentcell][1]+4);y++)
                        {
                        ((counter[x][y]))=((counter[x][y]))+SPREAD123;
                        }
                        }

                        //counter iterations within 4 radii
                        for(x=(cell[currentcell][0]-4);x<(cell[currentcell][0]+5);x++)
                        {
                        for(y=(cell[currentcell][1]-4);y<(cell[currentcell][1]+5);y++)
                        {
                        ((counter[x][y]))=((counter[x][y]))+SPREAD1234;
                        }
                        }

                        counter[cell[currentcell][0]][cell[currentcell][1]]=COUNTERMAX;

    //      if(currentcell== ccount-1)
     //                  {
    //                   currentcell=0;
    //                   }
    //      else
    //                   {
                        currentcell++;
    //                   }
                }


  //updating slime for draw
  //printf("beforeslime  %d, ", turn);    //TAG
  //getch();//TAG

  // bmp initialize
    bmp[1]=(char) ((turn/100000)%10+48) ;
    bmp[2]=(char) ((turn/10000)%10+48) ;
    bmp[3]=(char) ((turn/1000)%10+48) ;
    bmp[4]=(char) ((turn/100)%10+48) ;
    bmp[5]=(char) ((turn/10)%10+48) ;
    bmp[6]=(char) (turn%10+48) ;
    wFile = fopen (bmp1 , "w");
    putc( 66  , wFile);
    putc( 77  , wFile);
```

```
1       //putc( -120  , wFile) ; //variable
2       //putc( 70  , wFile)     ;      //variable
3       //putc( 0  , wFile)      ;     //variablle
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);
```

```
putc( -1 , wFile);
}
if(slime[updatex][updatey]==BLUE)
{
putc( -1 , wFile);
putc( 0 , wFile);
putc( 0 , wFile);
}
if(slime[updatex][updatey]==GREEN)
{
putc( 0 , wFile);
putc( -1 , wFile);
putc( 0 , wFile);
}
if(occupy[updatex][updatey]>BLACK)
{
slime[updatex][updatey]=RED;
}
}
}
  putc( -1, wFile);   //EOF

 fclose (wFile); //close output to bmp
draw_cell(RIGHT-1, BOTTOM-1, (char) turn);
//to txt
//header
  txt[1]=(char) ((turn/100000)%10+48) ;
  txt[2]=(char) ((turn/10000)%10+48) ;
  txt[3]=(char) ((turn/1000)%10+48) ;
  txt[4]=(char) ((turn/100)%10+48) ;
  txt[5]=(char) ((turn/10)%10+48) ;
  txt[6]=(char) (turn%10+48) ;
  wFile = fopen (txt1 , "w");

  putc( (char) 78, wFile);
  putc( (char) 85, wFile);
  putc( (char) 77, wFile);
  putc( (char) 44, wFile);
  putc( (char) 88, wFile);
  putc( (char) 44, wFile);
  putc( (char) 89, wFile);
  putc( (char) 44, wFile);
  putc( txt[0]  , wFile);
  putc( txt[1]  , wFile);
  putc( txt[2]  , wFile);
  putc( txt[3]  , wFile);
  putc( txt[4]  , wFile);
  putc( txt[5]  , wFile);
  putc( txt[6]  , wFile);
  putc( txt[7]  , wFile);
  putc( txt[8]  , wFile);
  putc( txt[9]  , wFile);
  putc( txt[10]  , wFile);
  putc( (char) 13, wFile);
  putc( (char) 10, wFile);

//data
for(currentcell=0; currentcell<(ccount+1); currentcell++)
```

```
{
  putc ((char)((currentcell/100000)%10+48), wFile );
  putc ((char)((currentcell/10000)%10+48), wFile );
  putc ((char)((currentcell/1000)%10+48), wFile );
  putc ((char)((currentcell/100)%10+48), wFile );
  putc ((char)((currentcell/10)%10+48), wFile );
  putc ((char)((currentcell/1)%10+48), wFile );
  putc( (char) 44, wFile);
  putc ((char)((cell[currentcell][0]/100000)%10+48), wFile );
  putc ((char)((cell[currentcell][0]/10000)%10+48), wFile );
  putc ((char)((cell[currentcell][0]/1000)%10+48), wFile );
  putc ((char)((cell[currentcell][0]/100)%10+48), wFile );
  putc ((char)((cell[currentcell][0]/10)%10+48), wFile );
  putc ((char)((cell[currentcell][0]/1)%10+48), wFile );
  putc( (char) 44, wFile);
  putc ((char)((cell[currentcell][1]/100000)%10+48), wFile );
  putc ((char)((cell[currentcell][1]/10000)%10+48), wFile );
  putc ((char)((cell[currentcell][1]/1000)%10+48), wFile );
  putc ((char)((cell[currentcell][1]/100)%10+48), wFile );
  putc ((char)((cell[currentcell][1]/10)%10+48), wFile );
  putc ((char)((cell[currentcell][1]/1)%10+48), wFile );
  putc( (char) 44, wFile);
  putc ((char)((cell[currentcell][2]/100000)%10+48), wFile );
  putc ((char)((cell[currentcell][2]/10000)%10+48), wFile );
  putc ((char)((cell[currentcell][2]/1000)%10+48), wFile );
  putc ((char)((cell[currentcell][2]/100)%10+48), wFile );
  putc ((char)((cell[currentcell][2]/10)%10+48), wFile );
  putc ((char)((cell[currentcell][2]/1)%10+48), wFile );
  putc( (char) 44, wFile);
  putc( (char) 13, wFile);
  putc( (char) 10, wFile);
}
   fclose (wFile); //close output to text
//printf("end loop %d, ", turn); //TAG
//getch(); //TAG
}//end of the turn for() loop
draw_cell(0, 0, YELLOW); //DRAW DONE
  getch();
  closegraph();      //DRAW
  //printf("end program %d, ", turn);
  //getch();

  return;

}
```

Supplementary 2B Surface permissive to motility

```
#define LEFT 0
#define RIGHT 120
#define TOP 0
#define BOTTOM 75 //should be less than right

//color values
#define BLACK 0
#define WHITE 255
#define RED 100
#define GREEN 18
#define BLUE 17
#define GRAY 200
#define ORANGE 150
#define YELLOW 30
#define BLACKVALUE 0
#define GRAYVALUE 0
#define REDVALUE 25
#define LIGHT 4
#define LIGHTR 4
#define DARK 1
#define DARKR 1
#define R 2
#define MODMOVE 200

#define COUNTERMAX 1000
#define GRAYSLIME 50  //starting value of counter in initial spot
#define SPREAD1 4
#define SPREAD12 1
#define SPREAD123 1
#define SPREAD1234 1

#define FILL 20//warning, cellmax
#define RIM 3

#define CELLMAX 2000
#define MAXTURN 800
```