Implementacja komórkowa modelu SIR

Zaczniemy od NAJPROSTSZEJ WERSJI MODELU gdzie CHOROBA JEST BARDZO KRÓTKA i BARDZO MAŁO ZARAŹLIWA. Formalnie będzie to dwuwymiarowy, probabilistyczny (kroki MC) automat komórkowy z regułą SIR.

ModelSIR1

```
int WorldSize=400;//Ile chcemy elementów w linii i ile linii (tablica kwadratowa)

int[][] World=new int[WorldSize][WorldSize];//Tworzenie tablicy świata
                                            //- w Processingu zawsze za pomocą alo
float IDens=0.99;//Początkowa gęstość w tablicy

//final to coś w rodzaju stałych
final int Empty=0;
final int Susceptible=1;
final int Infected=2;
final int Recovered=3;

void setup()
{//Empty=-1;//ERROR: The final field cannot be assigned!
 size(400,400);    //Okno kwadratowe
 noSmooth();       //Znacząco przyśpiesza

 if(IDens>0)
  {
   for(int i=0;i<World.length;i++) //Zasiewanie tablicy
    for(int j=0;j<World.length;j++)
      if(random(1.0)<IDens)
        World[i][j]=Susceptible;
      else
        World[i][j]=Empty;//Dla pewności, gdyby Empty nie było zero.
  }

 World[WorldSize/2][WorldSize/2]=Infected;

 frameRate(10);
}
```

Setup odpowiada za zasiewanie tablicy z zadaną gęstością początkową zdrowymi komórkami, oraz jedną pojedynczą komórką zarażoną na środku.
Procedura **draw**() odpowiada za wizualizacje, i za zmianę stanu modelu. Zaczynamy od wizualizacji bo jest ona po prostu odziedziczona po poprzednich automatach komórkowych.

```
38 int t=0;
39
40 void draw()
41 {
42   for(int i=0;i<World.length;i++)//Wizualizacja czyli "rysowanie na ekranie"
43     for(int j=0;j<World.length;j++)
44     {
45       switch(World[i][j]){ //Instrukcja wyboru pozwala nam wybrać
46       case 3:stroke(0,255,0);break;//dowolny kolor w zależności od liczby w komórce
47       case 2:stroke(255,0,0);break;
48       case 1:stroke(0,0,255);break;
49       case 0:stroke(0,0,0);break;
50       default: stroke(255);//To się pojawiac nie powinno
51       break;
52       }
53       point(i,j);
54     }
```

Na razie cztery kolory w zupełności nam wystarczą. Gdyby się pojawił jakiś inny, nieprzewidziany stan to komórka zostanie wyświetlona na biało, za co odpowiada linia 50.

Teraz implementacja samego modelu:

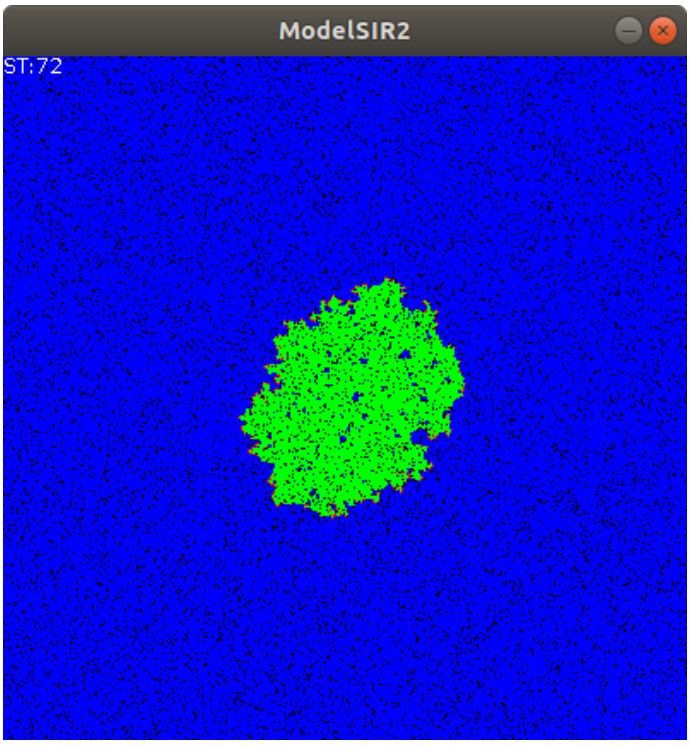
```
57   //STANY: Empty=0; Susceptible=1; Infected=2; Recovered=3;
58   for(int a=0;a<World.length*World.length;a++)//Tyle losowań ile komórek
59   {
60        //Losowanie agenta
61        int i=(int)random(World.length);
62        int j=(int)random(World.length);
63
64        //Jesli pusty lub zdrowy zdrowy to nic nie robi
65        if(World[i][j]!=Infected) continue;
66
67        //Wyliczenie lokalizacji sąsiadów
68        int right = (i+1) % WorldSize;
69        int left  = (WorldSize+i-1) % WorldSize;
70        int dw=(j+1) % WorldSize;
71        int up=(WorldSize+j-1) % WorldSize;
72
73        int neigh=(int)random(4);//który sąsiad
74
75        switch(neigh) //Tu trzeba bardzo uważac żeby się nie pomylić w indeksach
76        {
77        case 0: if(World[left] [j]==Susceptible) World[left][j]=Infected;  break;
78        case 1: if(World[right][j]==Susceptible) World[right][j]=Infected; break;
79        case 2: if(World[i][up]==Susceptible) World[i][up]=Infected;break;
80        case 3: if(World[i][dw]==Susceptible) World[i][dw]=Infected;break;
81        default: println("Ups!",neigh);break;
82        }
83
84        World[i][j]=Recovered;
85    }
86
87    t++;//Kolejne pokolenie/krok/rok
88    text("ST:"+t,0,10);
89 }
```

Nie zdziwcie się, że ta epidemia wcale nie wybucha. Jej aktualne parametry nie dopuszczają do tego.
Jak to zmienić żeby miała szansę wybuchnąć?

Najprostszy sposób to zwiększyć liczbę interakcji:

```
ModelSIR2
74
75        for(int b=0;b<4;b++)//Więcej interakcji
76        {
77          int neigh=(int)random(4);
78
79          switch(neigh) //Tu trzeba bardzo uważac żeby się nie pomylić w indeksach
80          {
81          case 0: if(World[left] [j]==Susceptible) World[left][j]=Infected;  break;
82          case 1: if(World[right][j]==Susceptible) World[right][j]=Infected;  break;
83          case 2: if(World[i][up]==Susceptible) World[i][up]=Infected;break;
84          case 3: if(World[i][dw]==Susceptible) World[i][dw]=Infected;break;
85          default: println("Ups!",neigh);break;
86          }
87        }
88        World[i][j]=Recovered;
```

Zamykamy switch interakcji w pętle. Teraz dostajemy epidemie zarazka, który jest bardzo zaraźliwy.



Tak to mniej więcej powinno wyglądać…

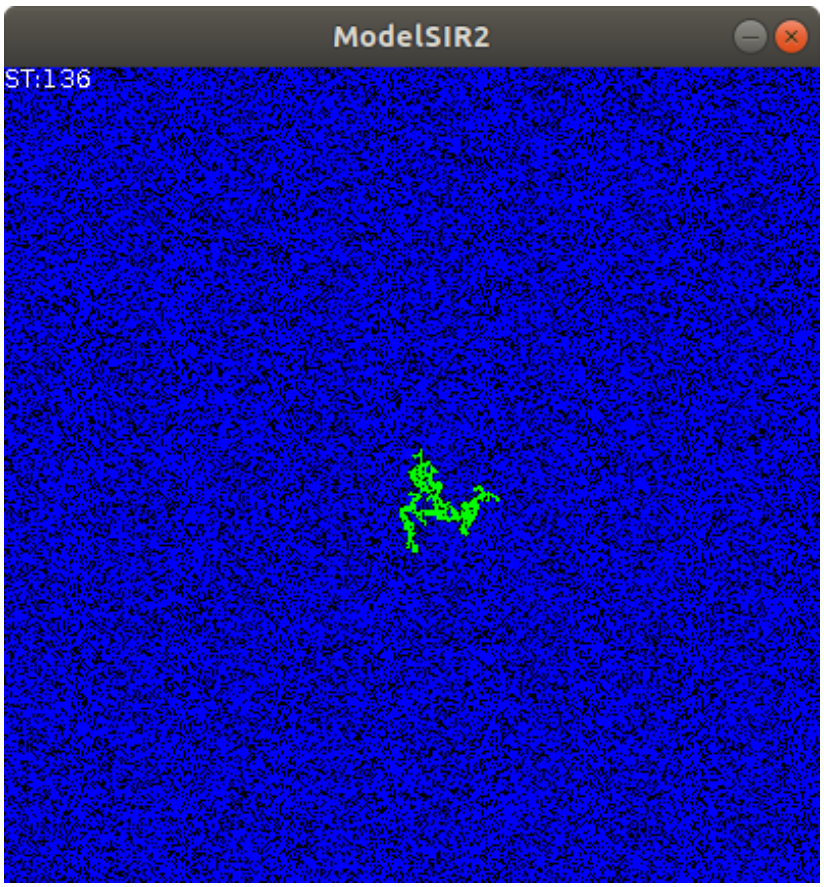
Ale czy zawsze? Co możemy zmienić? Nie zbadaliśmy jeszcze znaczenia parametru

“gęstości zaludnienia” (*IDens*) .

Z praktyki epidemicznej wiadomo że ma on ogromne znaczenie. Poniżej pewnego progu epidemie zawsze są lokalne. A jaki jest ten próg? Spróbujcie go znaleźć:

```
10 float IDens=0.9; //Początkowa gęstość w tablicy - jaka jest gęstość progowa,
11                  //przy której epidemia zaatakuje ZAWSZE cały świat? (o ile już się zacznie)
12                  //Choć mogą być małe rejony które ominęła.
13
```

Można się tu posłużyć algorytmem bisekcji:
Najpierw sprawdza jakąś wartość mocno poniżej aktualnej. Np. 0.5 …
Okazuje się że przy tej wartości parametru epidemia wcale nie wybucha.
W takim razie sprawdzamy wartość pomiędzy 0.5 i 0.9 czyli 0.7 …
Epidemie nadal są tylko lokalne - “pandemii”, czyli epidemii zakażającej więcej niż połowę “świata” nie daje się wywołać.



No w takim razie sprawdzamy wartość pomiędzy 0.7 a 0.9 czyli 0.8 …

Teraz “pandemia” startuje niemal za każdym razem.

Czyli musimy sprawdzić wartość między 0.7 a 0.8. Itd.

Im bliżej jesteśmy wartości krytycznej tym więcej prób trzeba przeprowadzić, bo będzie się zdarzać że pandemia czasem startuje, a czasem nie, ale jest przewaga jednej z możliwości.

No to do roboty :-D

To jaki jest wasz wynik?
Mi wyszło ***0.775*** , ale może wam się uda dokładniej.

No to uzupełnimy teraz nasz model o prawdopodobieństwa zarażenia, wyzdrowienia i śmierci. Są one łatwe do uzyskania empirycznego, ale nie bardzo są zgodne z biologią typowych chorób epidemicznych. Rozwój większości chorób jest uzależniony od etapów odpowiedzi immunologicznej, które mają typowy czas trwania - pierwszy etap odporności po 7 dniach, a pełna odporność po ok 14. Myślmy nad tym w międzyczasie, a teraz implementacja modelu z prawdopodobieństwami.

ModelSIR3

```
20
21 final float PTransfer=0.33;   //Prawdopodobieństwo zarażenia agenta w pojedynczej interakcji
22 final float PRecovery=0.10;   //Średnie prawdopodobieństwo wyzdrowienia w danym dniu
23 final float PDeath=0.01;      //Średnie prawdopodobieństwo śmierci w danym dniu choroby
24                               //PDeath + PRecovery < 1  !!!
25
26 void setup()
27 {
28  assert PDeath + PRecovery < 1 : "Za duże prawdopodobieństwa PDeath + PRecovery";//Asercja
```

Wprowadzamy trzy prawdopodobieństwa - zakażenia w pojedynczej interakcji (*PTransfer*), wyzdrowienia w danym dniu infekcji (*PRecovery*) i śmierci w danym dniu infekcji (*PDeath*). Ponieważ logika wskazuje że nie można jednocześnie wyzdrowieć i umrzeć, suma tych dwu prawdopodobieństw musi być mniejsza od 1. Do sprawdzenia tego założenia używamy specjalnego narzędzia programistycznego nazywanego “**ASERCJĄ”**.

W JAVIE i Processingu asercja ma następującą składnie:

assert *Warunek* : “komunikat”;

Jeśli warunek asercji nie jest spełniony to komunikat jest wyświetlany na konsoli i program się zatrzymuje. W naszym wypadku komunikat jest podkreślony na żółto, ponieważ warunek może być sprawdzony już przez środowisko Processingu, które wykrywa i ostrzega, że ten kod nigdy nie będzie wykonany.

No to teraz pętla zmiany stanu:

```
69  for(int a=0;a<World.length*World.length;a++)//Tyle losowań ile komórek
70  {
71       //Losowanie agenta
72       int i=(int)random(World.length);
73       int j=(int)random(World.length);
74
75       //Jesli pusty lub zdrowy zdrowy to nic nie robi
76       if(World[i][j]!=Infected) continue;
77
78       //Wyliczenie lokalizacji sąsiadów
79       int right = (i+1) % WorldSize;
80       int left  = (WorldSize+i-1) % WorldSize;
81       int dw=(j+1) % WorldSize;
82       int up=(WorldSize+j-1) % WorldSize;
83
84       //PTransfer - Prawdopodobieństwo zarażenia agenta w pojedynczej interakcji
85       //PRecovery - Prawdopodobieństwo wyzdrowienia w danym dniu
86       //PDeath    - Prawdopodobieństwo śmierci w danym dniu choroby
87       //PDeath + PRecovery < 1  !!!
88
89       if(World[left] [j]==Susceptible && random(1) < PTransfer) World[left][j]=Infected;
90       if(World[right][j]==Susceptible && random(1) < PTransfer) World[right][j]=Infected;
91       if(World[i][up]==Susceptible && random(1) < PTransfer) World[i][up]=Infected;
92       if(World[i][dw]==Susceptible && random(1) < PTransfer) World[i][dw]=Infected;
93
94       float prob=random(1);//Los na dany dzień
95
96       if(prob<PDeath) //Albo tego dnia umiera
97          World[i][j]=Empty;
98       else if(prob<PRecovery+PDeath)//Albo jest wyleczony
99              World[i][j]=Recovered;
100          //else //NADAL CIERPI!
101 }
```

Zrezygnowaliśmy z wewnętrznej pętli interakcji. Każdego dnia epidemii (kroku MC) chory ma szansę zarazić czterech sąsiadów, ale czy mu się uda zależy od prawdopodobieństwa transferu.
Chory może też w danym dniu umrzeć, albo wyzdrowieć, a w obu przypadkach przestaje zarażać.
W tym modelu możemy przetestować epidemie różnych znanych chorób. Z braku dostępu do realnych danych opisałem je w tabeli jakościowo,

Choroba / parametr	PTransfer	PDeath	PRecovery
Katar	bardzo wysoki	zero	~1/10
Ebola	bardzo wysoki	wysoki	~/100
Dżuma	wysoki	dosyć wysoki	~1/10
Grypa	wysoki	niski	~1/10
Gruźlica	niski	niski	zero
Żółtaczka B	średni	niski	zero
Broń B!	bardzo wysoki	średni	bliski zera

Oczywiście teraz przydałyby się jakieś statystyki. Minimum to wyświetlane na konsoli. Możemy je, tak jak w poniższym przykładzie zliczać już w trakcie symulacji.

ModelSIR4

```
93
94        if(World[left] [j]==Susceptible && random(1) < PTransfer)
95         {World[left][j]=Infected; kranken++;}
96        if(World[right][j]==Susceptible && random(1) < PTransfer)
97         {World[right][j]=Infected; kranken++;}
98        if(World[i][up]==Susceptible && random(1) < PTransfer)
99         {World[i][up]=Infected; kranken++;}
100       if(World[i][dw]==Susceptible && random(1) < PTransfer)
101        {World[i][dw]=Infected; kranken++;}
102
103       float prob=random(1);//Los na dany dzień
104
105       if(prob<PDeath) //Albo tego dnia umiera
106        {World[i][j]=Empty;starben++;}
107       else if(prob<PRecovery+PDeath)//Albo jest wyleczony
108             {World[i][j]=Recovered;geheilt++;}
109            //else //NADAL CIERPI!
110   }//Koniec petli po wylosowanych agentach
111
112   t++;//Kolejne pokolenie/krok/rok
113   text("ST:"+t+" Zachorowali:"+kranken+" Wyzdrowieli:"+geheilt+" Umarli:"+starben,0,10);
114   println("ST:"+t+"\tZ\t"+kranken+"\tW\t"+geheilt+"\tU\t"+starben);
115 }
```

Jeśli zmienne te będziemy zerować przed każdym krokiem M C, to uzyskamy statystyki z kroku, a jeśli nie, będą to kumulatywne statystyki z całości epidemii.
Dane z konsoli Processingu można skopiować używając skrótów klawiszowych Ctrl-A (zaznacz wszystko) , Ctrl-C (kopiuj) i wkleić (Ctrl-V) gdzieś, gdzie potrafimy wykonać wykres.

SIR z konkretną liczbą dni choroby

Pozostała nam jeszcze wersja modelu, w którym choroba będzie trwała konkretną liczbę dni. Czyli LICZBA INTERAKCJI DZIENNIE wynosi 4, ale prawdopodobieństwo zarażenia

równe *PTransfer*, CHOROBA trwa u zarażonego *Duration* kroków chyba że wcześniej umrze (*PDeath*).

```
ModelSIR5
16 //Coś w rodzaju stałych ;-)
17 final int Duration=7;//Czas trwania infekcji!
18 final int Empty=0;
19 final int Susceptible=1;
20 final int Infected=2;
21 final int Recovered=Infected+Duration;
22 final float PTransfer=0.75;   //Prawdopodobieństwo zarażenia agenta w pojedynczej interakcji
23 final float PDeath=0.01;      //Średnie prawdopodobieństwo śmierci w danym dniu choroby
24
25 //STATYSTYKI LICZONE W TRAKCIE SYMULACJI
26 int sumInfected=0;//Zachorowanie
27 int sumRecovered=0;//Wyzdrowienia
28 int sumDeath=0;//Ci co umarli
```

Znika nam prawdopodobieństwo wyzdrowienia. Wyzdrowienia następuje po konkretnej liczbie losowań (więc jest trochę zmienne). Agent zdrowieja zawsze o ile nie nastąpi śmierć.

Musimy zmodyfikować główną pętlę programu, gdyż teraz choroba będzie reprezentowana przez wiele stanów. Każdy stan to kolejny etap/dzień choroby.

```
ModelSIR5
68  //Zmiana stanu automatu - krok Monte Carlo
69  //STANY: Empty=0; Susceptible=1; Infected=2; Recovered=Infected+Duration;
70  for(int a=0;a<World.length*World.length;a++)//Tyle losowań ile komórek
71  {
72        //Losowanie agenta
73        int i=(int)random(World.length);
74        int j=(int)random(World.length);
75
76        //Jesli pusty lub zdrowy to nic nie robimy
77        if(World[i][j]<Infected || Recovered<=World[i][j]) continue;
78
79        //Wyliczenie lokalizacji sąsiadów
80        int right = (i+1) % WorldSize;
81        int left  = (WorldSize+i-1) % WorldSize;
82        int dw=(j+1) % WorldSize;
83        int up=(WorldSize+j-1) % WorldSize;
84
85        //PTransfer - Prawdopodobieństwo zarażenia agenta w pojedynczej interakcji
86        //PRecovery - Prawdopodobieństwo wyzdrowienia w danym dniu
87        //PDeath    - Prawdopodobieństwo śmierci w danym dniu choroby
88        //PDeath + PRecovery < 1  !!!
89
90        if(World[left] [j]==Susceptible && random(1) < PTransfer)
91         {World[left][j]=Infected; sumInfected++;}
92        if(World[right][j]==Susceptible && random(1) < PTransfer)
93         {World[right][j]=Infected; sumInfected++;}
94        if(World[i][up]==Susceptible && random(1) < PTransfer)
95         {World[i][up]=Infected; sumInfected++;}
96        if(World[i][dw]==Susceptible && random(1) < PTransfer)
97         {World[i][dw]=Infected; sumInfected++;}
```

Stąd w liniach 76 i 77 mamy bardziej złożony warunek, wyłączający komórki zdrowe. Sam sposób infekcji się nie zmienia, jednak zmienia się dalsze traktowanie zakażonego agenta:

```
98
99          float prob=random(1);//Los na dany dzień
100
101         if(prob<PDeath) //Albo tego dnia umiera
102          {World[i][j]=Empty;sumDeath++;}
103          {
104            //Albo jest wyleczony
105            if(++World[i][j]==Recovered)
106                sumRecovered++;
107            //else //NADAL CIERPI!
108          }
109     }
110
111     t++;//Kolejne pokolenie/krok/rok
112     text("ST:"+t+" Zachorowali:"+sumInfected+" Wyzdrowieli:"+sumRecovered+" Umarli:"+sumDeath
113           ,0,10);
114     println("ST:"+t+"\tZ\t"+sumInfected+"\tW\t"+sumRecovered+"\tU\t"+sumDeath);
115 }
```

Warunek w linii 105 wykonuje jednocześnie dwie operacje:

++World[i][j]

to preinkrementacja stanu komórki. Dopiero wynik tej operacji jest porównywany ze stanem *Recovered*, żeby można było zliczyć wyleczonych.
Na koniec wyświetlanie stanu… I już.

Zaraz, zaraz :-D
Pewnie zauważyliście że coś nam się popsuło teraz wyświetlanie.
Nie dziwne, skoro zmieniły się wartości stanów, a pętla wyświetlająca była skopiowana z prostego automatu komórkowego. Musimy to teraz poprawić...

```
50 int t=0;
51
52 void draw()
53 {
54   for(int i=0;i<World.length;i++)//Wizualizacja czyli "rysowanie na ekranie"
55     for(int j=0;j<World.length;j++)
56     {
57       switch(World[i][j]){ //Instrukcja wyboru pozwala nam wybrać dowolny kolor
58       case Recovered:  stroke(0,255,0);break;//Wyleczony
59       case Infected:   stroke(255,0,0);break;//Zachorował
60       case Susceptible:stroke(0,0,255);break;//Podatny
61       case Empty:      stroke(0,0,0);break;//Pusty
62       default:         stroke(random(255),0,random(255));//Chory
63       break;
64       }
65       point(i,j);
66     }
```

… używając zdefiniowanych wcześniej stałych.
Natomiast default: służy nam teraz do kolorowania aktualnie chorych za pomocą losowych odcieni fioletu.
No to starczy… **Pomyślcie jeszcze jak można wprowadzić utratę odporności. Przyda nam się, gdy wrócimy do epidemii w wersji agentowej.**