Game of Life i MATLAB

Simon Sigurdhsson

2 oktober 2008

Innehåll

Kapitel 1

Historia

Inlämningsuppgift 3 är den sista och defenitivt roligaste uppgiften i kursen Matematisk programvara, då den behandlar en simulering som är mycket intressant på väldigt många sätt; *Game of Life*. Denna simulering, som är tänkt att faktiskt simulera verkligt liv på en lite mer abstrakt och matematisk form, uppfanns redan 1970 av den brittiska matematikern John Conway.

Många av de formationer som kan bildas i Game of Life har intressanta egenskaper; det finns figurer som går framåt i all oändlighet, figurer som oscillerar, generatorer som skapar fler figurer och även mönster som växer i all oändlighet. De första figurer som upptäcktes och analyserades var små och relativt enkla, och analyserades med hjälp av papper och penna. Nuförtiden görs sådant lättast med hjälp av datorer, och i det här fallet ska vi implementera Game of Life i MATLAB.

Kapitel 2

Implementation

Implementationen av Game of Life i MATLAB kommer bli något av ett fulhack, då MATLAB egentligen inte är konstruerat för att göra saker som dessa. I vissa avseenden är MATLAB dock mycket väl lämpat för denna uppgift, främst när det gäller representationen av själva världen i spelet, som bekvämt kan göras med hjälp av matriser.

2.1 Regler

Game of Life följer ett bestämt antal regler. I grunden fungerar det så att mellan varje generation elimineras eller uppstår nya celler i rutnätet, baserat på dessa fyra regler:

1. Levande celler med mindre än två levande grannar dör
2. Levande celler med mer än tre grannar dör
3. Levande celler med två eller tre grannar överlever
4. Döda celler med exakt tre levande grannar återuppstår

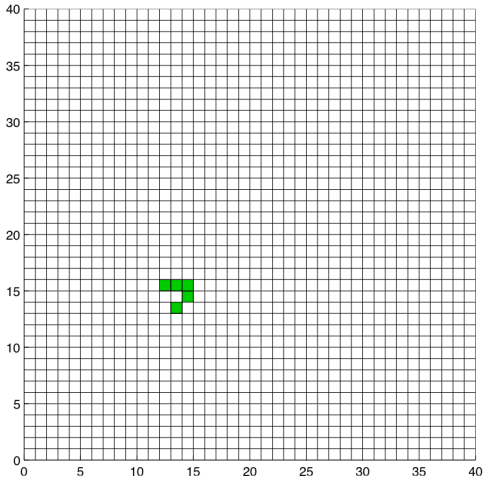
Regel 3 följer helt enkelt av de två första, som ska simulera ensamhet respektive trängsel. Den sista regeln är den enda som faktiskt skapar liv, men detta är fullt tillräckligt. Det bör även nämnas att dessa regler endast gäller för tvådimensionella rutnät med celler som är kvadratiska, dvs. varje cell har nio grannar. Det finns även regeluppsättningar för hexagonala rutnät.

Man kan även definiera vissa begrepp så som ljushastighet (en ruta per generation) och namn på olika grupper av figurer, men det är irrelevant för uppgiften i fråga.

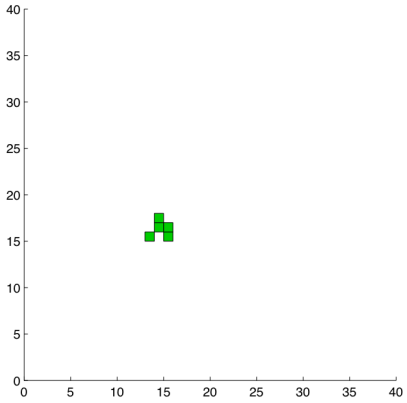
2.2 Exempelkörning

En exempelkörning kommer nu att demonstreras. Vi kör GoL.m i MATLAB, och får upp ett rutnät, 40×40 rutor stort (notera att källkoden har ett större rutnät, detta p.g.a. ytterligare tester efter exempelkörningen). MATLAB säger nu åt oss att mata in vår startgeneration genom att med musen klicka på de rutor vi vill aktivera (eller inaktivera) och därmed skapa en intressant startuppställning, som kan ses i Figur 2.1a.

Vi trycker `enter` för att avsluta inmatningen och därmed starta simuleringen. Efter ett par generationer går vi tillbaks till MATLABs fönster och stoppar exekveringen, då ingen pausfunktion har implementerats. Detta för att kunna spara ännu en bild för att visa att vår implementation verkligen fungerar, vilket den gör. Som man kan se i Figur 2.1b har det gått ett par generationer, och vår *glider* har rört sig diagonalt några rutor.



(a) Inmatning av första generationen



(b) Rutnätet efter några generationer

Figur 2.1: Resultatet av testkörningen som gjordes, då en så kallad *glider* ritats upp som första generation.

Några saker skall påpekas när det gäller källkoden i Appendix A; denna har nämligen utvecklats något sedan den här exempelkörningen utfördes. Rutnätet har utökats till 150×113 celler för att bättre kunna studera lite större mönster, en generationsräknare har lagts till för att kunna föra lite inexakt statistik över mönster, och lite kosmetiska ändringar har gjorts.

Bilaga A

MATLAB-kod

A.1 GoL.m

```
drawlife = @Drawlifegrid;
nextgen = @Nextlifegen;

width=150;
height=113;

life=zeros(width,height);
axis([0 width 0 height]);
axis image;
cla;

hlinex = [0:1:width;0:1:width];
hliney = [zeros(1,width+1);height*ones(1,width+1)];
vlinex = [zeros(1,height+1);width*ones(1,height+1)];
vliney = [0:1:height;0:1:height];
 linec = [.7 .7 .7];

line(hlinex, hliney, 'Color', linec)
line(vlinex, vliney, 'Color', linec)

text(width/2, 0, '\bfVlj punkter genom att klicka,
   och tryck ENTER efter sista punkten.', 'Color',
   'b', 'HorizontalAlignment', 'center',
   'VerticalAlignment', 'bottom');
[x y] = ginput(1);
```

```
life(floor(x)+1,floor(y)+1) =
   not(life(floor(x)+1,floor(y)+1));
drawlife(life);
while(1)
    line(hlinex, hliney, 'Color', linec)
    line(vlinex, vliney, 'Color', linec)
    [x y] = ginput(1);
    if(isempty(x) ~= 1)
        life(floor(x)+1,floor(y)+1) =
           not(life(floor(x)+1,floor(y)+1));
        drawlife(life);
    else
        %saveas(gcf,'Figur1.eps');
        drawlife(life);
        break
    end
end

generations = 0;
while(1)
    life=nextgen(life);
    drawlife(life);
    text(width/200, height, ['\bfGeneration ',
       num2str(generations)], 'Color', 'b',
       'VerticalAlignment', 'top');
    pause(25/(width*height));
    generations = generations + 1;
end
```

A.2 Drawlifegrid.m

```
function a = Drawlifegrid(lgrid)
    cla;
    [w h] = size(lgrid);
    for i=1:w
        for j=1:h
            if lgrid(i,j)==1
                patch([i-1;i;i;i-1],[j-1;j-1;j;j],[0
                    .8 0],'EdgeColor', 'none');
            end
        end
```

```
    end
end
```

A.3 Nextlifegen.m

```
function b=Nextlifegen(lgrid)
    [w h]=size(lgrid);
    l=zeros(w+2,h+2);
    l(2:(w+1),2:(h+1))=lgrid;
    lgrid=l;
    ng=zeros(w,h);
    for i=2:(w+1)
        for j=2:(h+1)
            previ=i-1;
            nexti=i+1;
            prevj=j-1;
            nextj=j+1;
            gr = lgrid(previ:nexti,prevj:nextj);
            gr(2,2)=0;
            s=sum(sum(gr));
            if lgrid(i,j)==1 && (s==2 || s==3)
                ng(previ,prevj)=1;
            elseif lgrid(i,j)==0 && s==3
                ng(previ,prevj)=1;
            else
                ng(previ,prevj)=0;
            end
        end
    end
    b=ng;
end
```