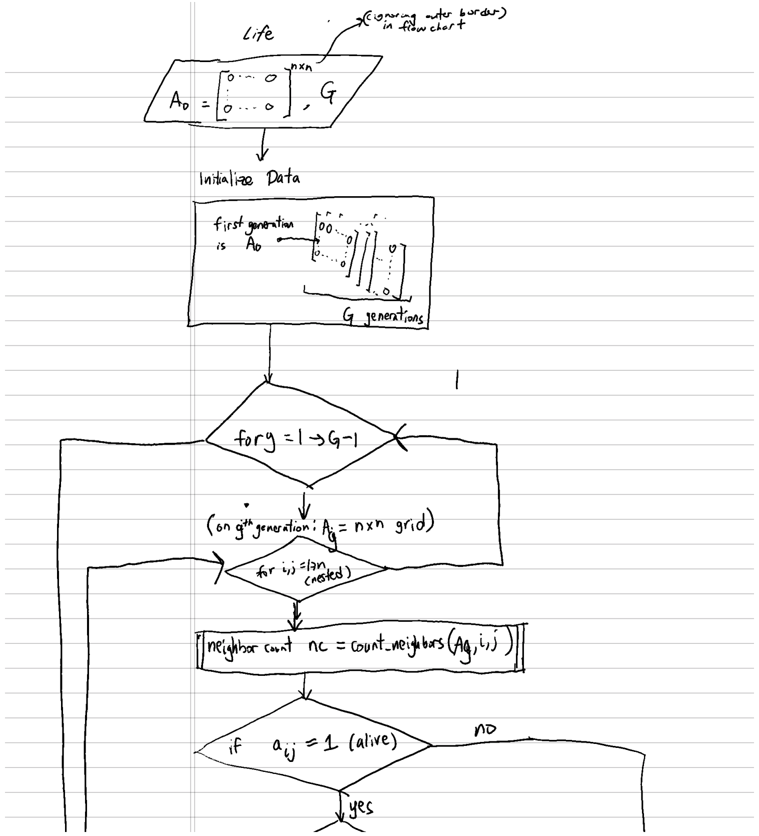
Assignment 02
Game of Life

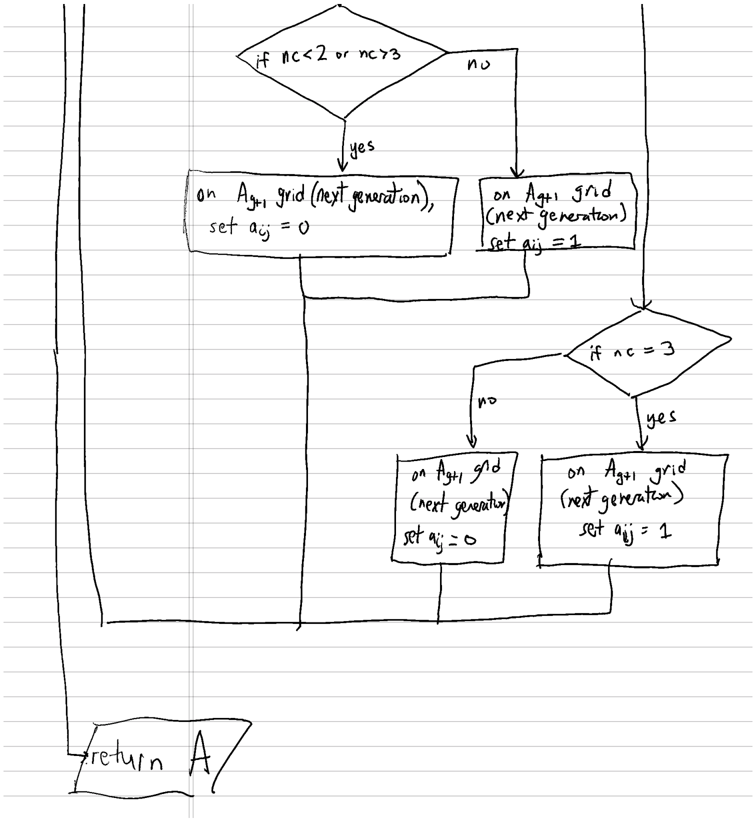
Minyoung Heo

January 25, 2024

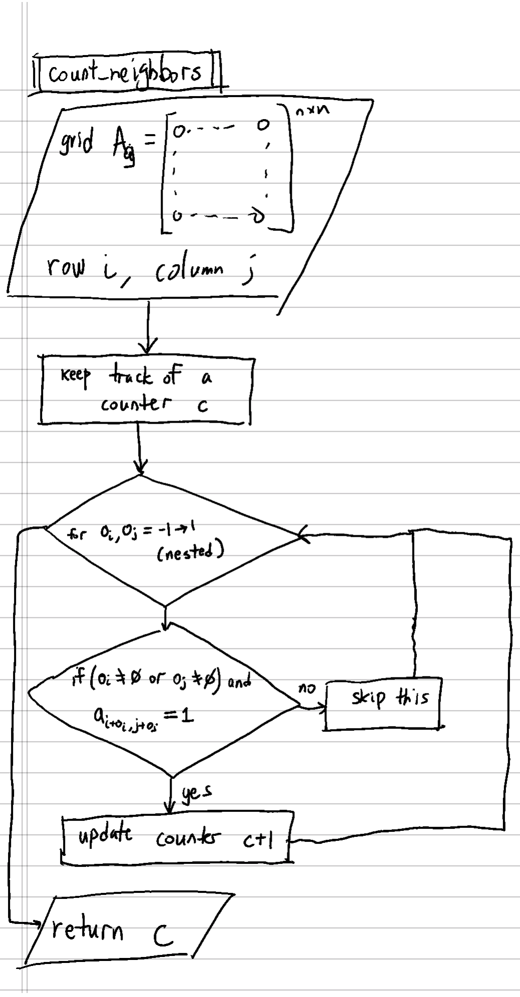
1 Pseudocode

Life


Life
(ignoring outer border) in flow chart
A0 = [0 ⋯ 0 ; 0 ⋯ 0] nxn , G
Initialize Data
first generation is A0
G generations
for g = 1 → G-1
(on gth generation: Ag = nxn grid)
for i,j = 1→n (nested)
neighbor count nc = count_neighbors(Ag, i, j)
if aij = 1 (alive)
no
yes


if nc<2 or nc>3
no
yes
on A_{g+1} grid (next generation), set $a_{ij} = 0$
on A_{g+1} grid (next generation) set $a_{ij} = 1$
if nc = 3
no
yes
on A_{g+1} grid (next generation) set $a_{ij} = 0$
on A_{g+1} grid (next generation) set $a_{ij} = 1$
return A

Counting Neighbors


count_neighbors
grid $A_{ij} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}^{n \times n}$
row i, column j
keep track of a counter c
for $o_i, o_j = -1 \to 1$ (nested)
if $(o_i \neq \emptyset$ or $o_j \neq \emptyset)$ and $a_{i+o_i, j+o_j} = 1$
no
skip this
yes
update counter c+1
return c

2 Code

Life.m

```
%   A Function that Simulates the Game of Life
%
% Input:
%
%   Init_Config: An nxn matrix of cells containing either a 1 (alive) or 0
% (dead). Acts as the initial configuration of the grid and is the first
% one on the series of grids.
%
%   Generations: An integer containing the number of "frames" this
% simulation plays through.
%
% Output:
%
%   Simulation: A 3-dimensional matrix containing Generations "frames" of
% nxn grids (nxnxGenerations) of the Game of Life being simulated on each
% "frame"
function [Simulation] = Life(Init_Config, Generations)
    n = size(Init_Config, 1);
    A = zeros(n+2, n+2, Generations);

    % initializing the first generation to Init_Config
    for i=1:n
        for j=1:n
            A(i+1, j+1, 1) = Init_Config(i, j);
        end
    end

    % the first generation is already set so we only need
    % to set after the first one
    for g=1:Generations-1
        % A_g is the gth generation of A
        A_g = A(:,:,g);

        % need to check (2, n+1) instead of (1, n)
        % because A has a "border" of zeros so we
        % really need to access indices 2-(n+1)
        for i=2:n+1
            for j=2:n+1
                nc = count_neighbors(A_g, i, j);
                if A_g(i, j) == 1
                    if nc < 2 || nc > 3
                        A(i, j, g+1) = 0;
                    else
                        A(i, j,g+1) = 1;
                    end
                else
                    if nc == 3
                        A(i, j, g+1) = 1;
                    else
                        A(i, j, g+1) = 0;
                    end
                end

            end
        end
    end

    % return
    Simulation = A;

end
```

counting_neighbors.m

```
% A function that counts the "neighboring cells" of (i, j) by using a
% nested for loop
%
% Input:
%
%   A_g: A (n+1)x(n+1) matrix that contains the gth "frame" of the
% simulation in Life
%
%   i, j: The coordinates (row, col) of A_g that we want to cout the
% neighboring cells of
function [c] = count_neighbors(A_g, i, j)
    c = 0;

    % for looping through the "offsets":
    % : (-1, -1), (-1, 0), ..., (1, 1)
    % such that for each offset o = (oi, oj)
    % (where i is row and j is column)
    % we can count the neighboring cell of (i, j) with
    % (i + oi, j + oj)
    for oi = -1:1
        for oj = -1:1
            % counting if neighboring cell is alive
            if (oi ~= 0 || oj ~= 0) && A_g(i+oi, j+oj) == 1
                c = c + 1;
            end
        end
    end

    % return (c is already c :P)

end
```

3 Some proof it works

Unfortunately, this doesn't work as well in pdf form, so the gifs are provided in the zip. But the Doc is on the next page.

Game of Life Documentation

Math 466

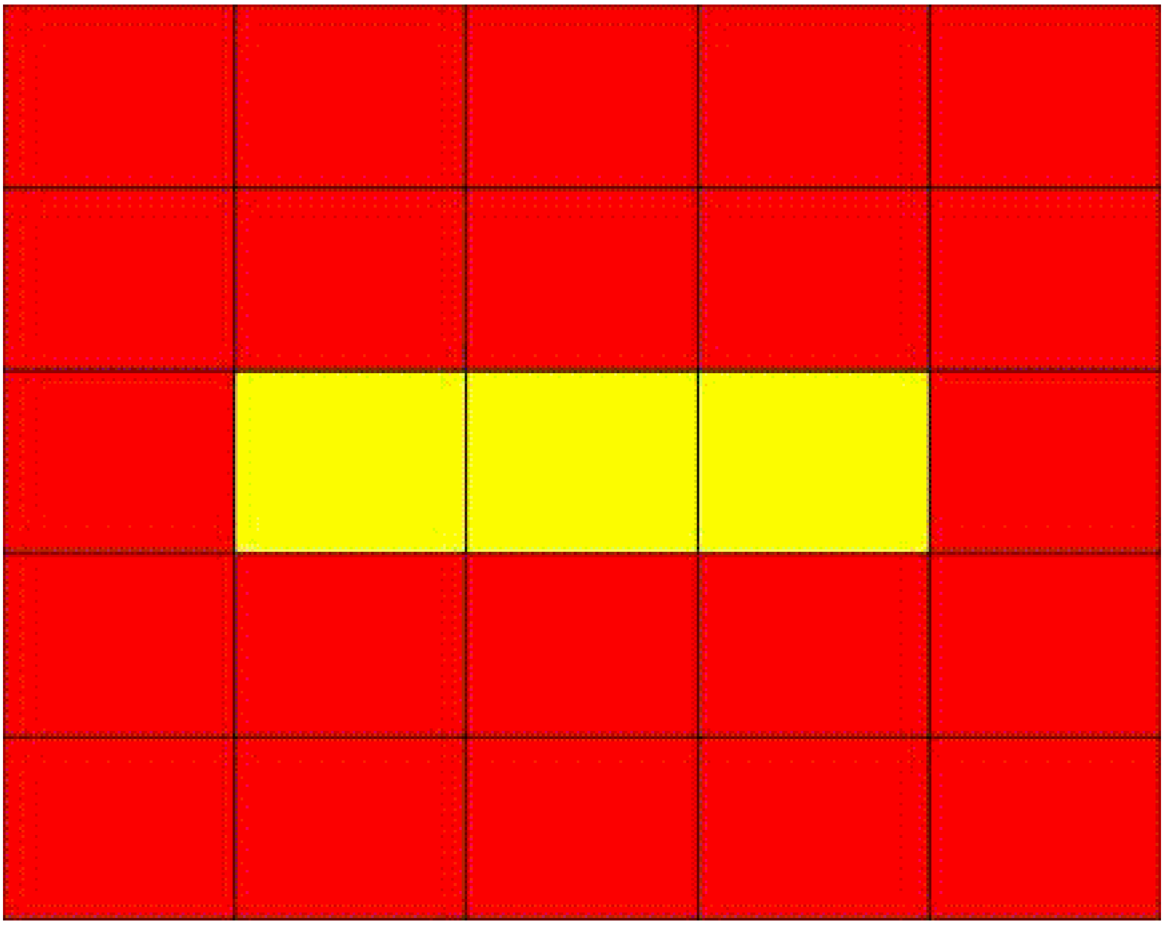
Minyoung Heo

Overview

Made 3 different tests: 1 random, 1 blinker, 1 glider, and 1 glider gun.

Random Test

Using `rand` function, I was able to make a random board of $n \times n$ with 1s and 0s given a certain "density" (the larger the density, the more it is going to place 1s). Here is a simulation of 1000 generations of 100x100 grid with a density of 0.1.



The Test Code

```
gens = 1000;
```

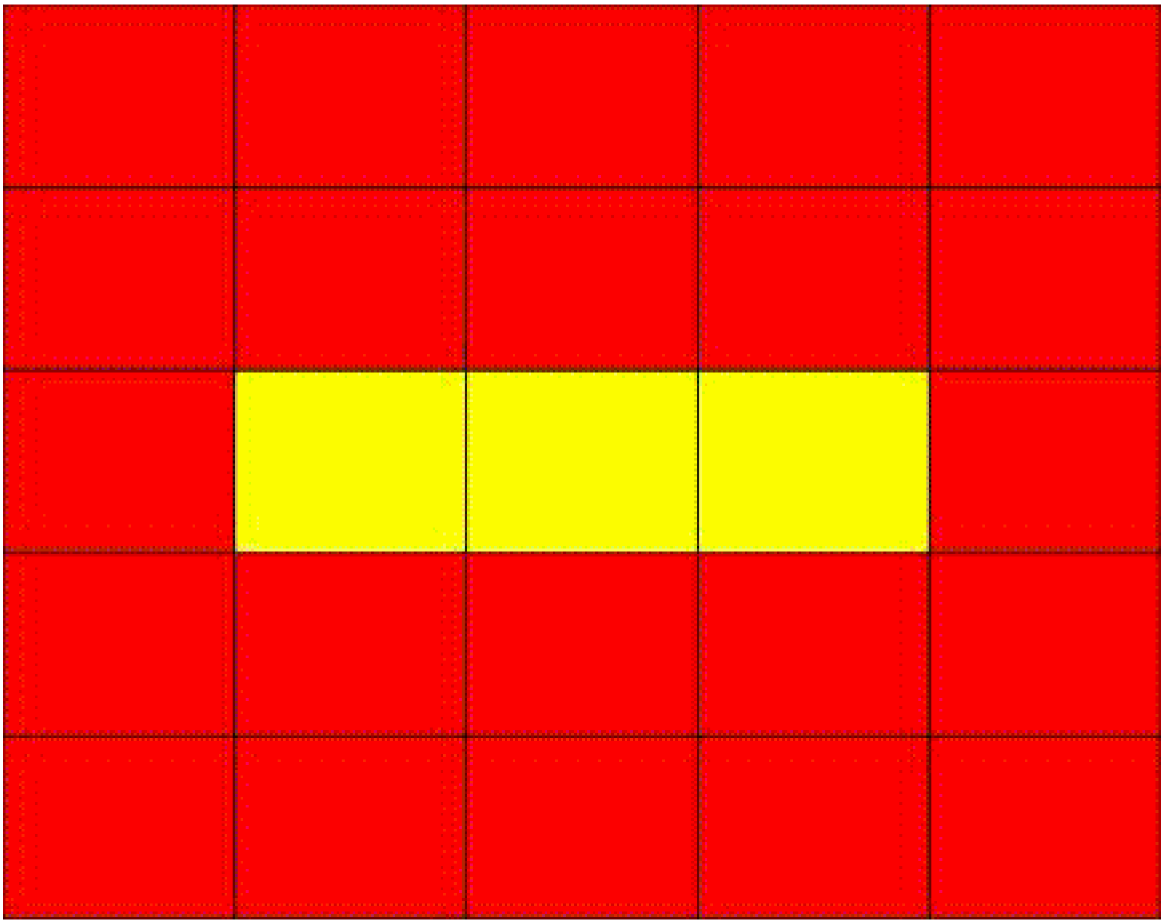
```
n = 100;
Init\_Config = zeros(n);

density = .10;
spawnCount = 0;

for i=1:n
for j=1:n
tospawn = 0;
if rand < density
tospawn = 1;
spawnCount = spawnCount+1;
end
Init\_Config(i,j) = tospawn;
end
end
disp("spawned: " + spawnCount);

% Init\_Config(5, 5) = 1;
% Init\_Config(5, 6) = 1;
% Init\_Config(5, 4) = 1;

%%
global log
log = fopen("outputlog.txt", "w");
fprintf(log, "%d %d %d %d %d %d %d %d %d %d\n", Init\_Config);
```

```
A = Life(Init\_Config, gens);

mov = Life\_Animation\_alt(A, 1);

v = VideoWriter('randomlife.avi');

open(v)

writeVideo(v, mov);

close(v);
```

Blinker Test

The Test Code

```
gens = 10;

n = 5;

Init\_Config = zeros(n);

Init\_Config(3, 3) = 1;

Init\_Config(3, 4) = 1;

Init\_Config(3, 2) = 1;

%%

global log

log = fopen("outputlog.txt", "w");

fprintf(log, "%d %d %d %d %d %d %d %d %d %d\n", Init\_Config);

A = Life(Init\_Config, gens);

mov = Life\_Animation\_alt(A, 1);

v = VideoWriter('blinkerlife.avi');

open(v)

writeVideo(v, mov);

close(v);
```



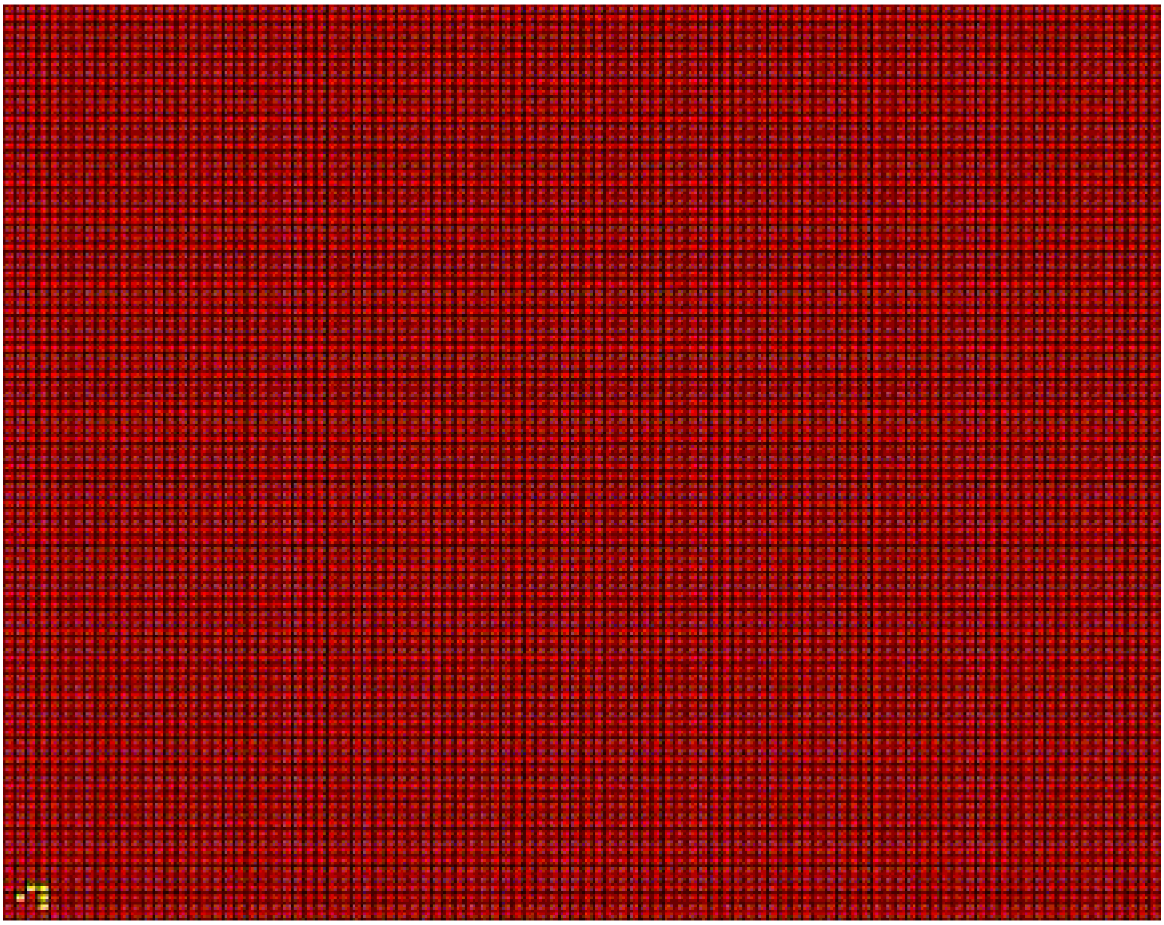
Glider Test

The “glider” that travels infinitely if it’s not stopped.

The Test Code

```
gens = 400;

n = 100;

Init\_Config = zeros(n);

Init\_Config(2, 4) = 1;

Init\_Config(3, 2) = 1;

Init\_Config(3, 4) = 1;

Init\_Config(4, 3) = 1;

Init\_Config(4, 4) = 1;

%%
```

```
global log

log = fopen("outputlog.txt", "w");

fprintf(log, "%d %d %d %d %d %d %d %d %d %d\n", Init\_Config);

A = Life(Init\_Config, gens);

mov = Life\_Animation\_alt(A, 1);

v = VideoWriter('gliderlife.avi');

open(v)

writeVideo(v, mov);

close(v);
```



Glider Gun

Made this one for fun but also to show that this works

Test Code

```
gens = 400;
```

```
n = 100;

Init\_Config = zeros(n);

<!--[if mso]><br><![endif]-->

glider\_gun = \[

% 1, 2 3 4 5 6 7 8 9 10,11,12 13 14 15 16 17 18 19 20

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0

<!--[if mso]><br><![endif]-->

% 1, 2 3 4 5 6 7 8 9 10,11,12 13 14 15 16 17 18 19 20

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0

<!--[if mso]><br><![endif]-->

% 1, 2 3 4 5 6 7 8 9 10,11,12 13 14 15 16 17 18 19 20

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0

<!--[if mso]><br><![endif]-->

% 1, 2 3 4 5 6 7 8 9 10,11,12 13 14 15 16 17 18 19 20

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0

<!--[if mso]><br><![endif]-->

% 1, 2 3 4 5 6 7 8 9 10,11,12 13 14 15 16 17 18 19 20

1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0

<!--[if mso]><br><![endif]-->

% 1, 2 3 4 5 6 7 8 9 10,11,12 13 14 15 16 17 18 19 20

1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0

<!--[if mso]><br><![endif]-->

% 1, 2 3 4 5 6 7 8 9 10,11,12 13 14 15 16 17 18 19 20

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0
<!--[if mso]><br><![endif]-->

% 1, 2 3 4 5 6 7 8 9 10,11,12 13 14 15 16 17 18 19 20

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

<!--[if mso]><br><![endif]-->

% 1, 2 3 4 5 6 7 8 9 10,11,12 13 14 15 16 17 18 19 20

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

];

<!--[if mso]><br><![endif]-->

glider\_gun

<!--[if mso]><br><![endif]-->

for i=1:9

for j=1:40

Init\_Config(i + 5, j + 5) = glider\_gun(i, j);

end

end

<!--[if mso]><br><![endif]-->

%%

global log

log = fopen("outputlog.txt", "w");

fprintf(log, "%d %d %d %d %d %d %d %d %d %d\n", Init\_Config);

<!--[if mso]><br><![endif]-->

A = Life(Init\_Config, gens);

mov = Life\_Animation\_alt(A, 1);

<!--[if mso]><br><![endif]-->
```

```
v = VideoWriter('glidergunlife.avi');

open(v)

writeVideo(v, mov);

close(v);
```