Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА №3

З ДИСЦИПЛІНИ “ КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ”

НА ТЕМУ: “ МОДЕЛЮВАННЯ КЛІТИННИХ АВТОМАТІВ ”

Виконав:
Студент III курсу ФІОТ
групи ІО-82
Шендріков Євгеній
Номер у списку - 25

Перевірив:
Радченко К.О.

м. Київ – 2020 р.

Завдання

1. Вивчити теоретичну частину.

2. Ознайомитися з програмою Life в **Matlab**.

3. Вивчити роботу програми *Conway's life*.

4. Для "хаотичної" початкової конфігурації, в якій кожна клітина знаходиться в стані 1 з ймовірністю 50% розглянути часову еволюцію правила 00010010 (правила 18), правила 01001001 (правила 73) і правила 10001000 (правила 136).

Виконання роботи

Для можливості розгляду часової еволюціїї правил (18, 73, 136) було розроблено 2 програми, одна була реалізована на основі інтерфейсу та функціоналу програму *Conway's life* з відповідними змінами, для другої програми був написаний власний інтерфейс та функціонал, який дозволяє будувати будь-яке правило (wolfram rule 1..256).

Перший варіант реалізації

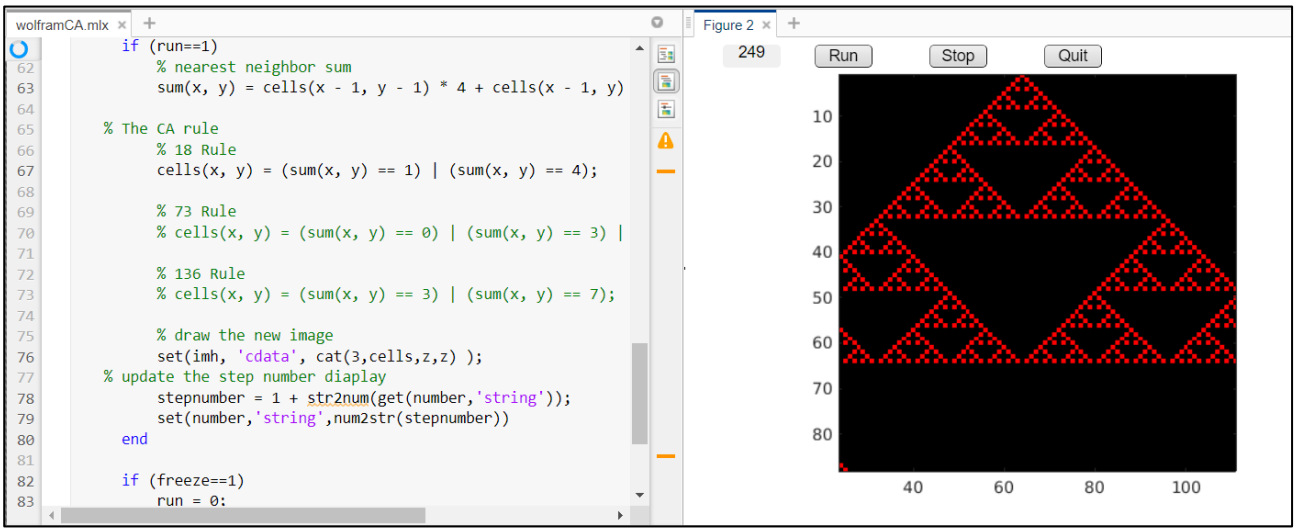
wolframCA.mlx

```
% Conway's life with GUI


clf
clear all


%==========================================
% build the GUI
% define the plot button
plotbutton=uicontrol('style','pushbutton',...
   'string','Run', ...
   'fontsize',12, ...
   'position',[100,400,50,20], ...
   'callback', 'run=1;');

% define the stop button
erasebutton=uicontrol('style','pushbutton',...
   'string','Stop', ...
   'fontsize',12, ...
   'position',[200,400,50,20], ...
   'callback','freeze=1;');

% define the Quit button
```

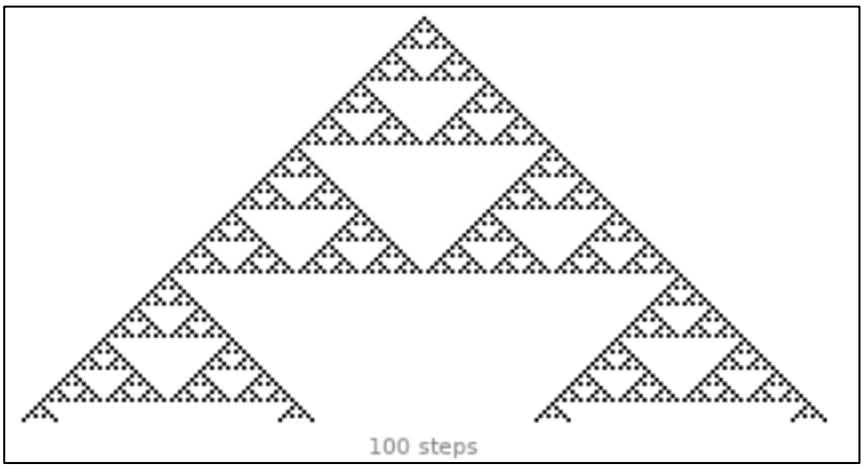
```
quitbutton=uicontrol('style','pushbutton',...
   'string','Quit', ...
   'fontsize',12, ...
   'position',[300,400,50,20], ...
   'callback','stop=1;close;');

number = uicontrol('style','text', ...
    'string','1', ...
   'fontsize',12, ...
   'position',[20,400,50,20]);

%=========================================
% CA setup
n=128;


% initialize the arrays
z = zeros(n,n);
cells = z;
sum = z;


% set a few cells to one
cells(1, n/2) = 1;


% build an image and display it
imh = image(cat(3,cells,z,z));
set(imh, 'erasemode', 'none')
axis equal
axis tight


% index definition for cell update
x = 2:n-1;
y = 2:n-1;


% Main event loop
stop= 0; % wait for a quit button push
run = 0; % wait for a draw
freeze = 0; % wait for a freeze
while (stop==0)
    if (run==1)
        % nearest neighbor sum
        sum(x, y) = cells(x - 1, y - 1) * 4 + cells(x - 1, y) * 2 + cells(x - 1, y
+ 1);

        % The CA rule
        % 18 Rule
        % cells(x, y) = (sum(x, y) == 1) | (sum(x, y) == 4);

        % 73 Rule
        % cells(x, y) = (sum(x, y) == 0) | (sum(x, y) == 3) | (sum(x, y) == 6);

        % 136 Rule
        cells(x, y) = (sum(x, y) == 3) | (sum(x, y) == 7);
```

```
        % draw the new image
        set(imh, 'cdata', cat(3,cells,z,z) );
        % update the step number diaplay
        stepnumber = 1 + str2num(get(number,'string'));
        set(number,'string',num2str(stepnumber))
    end

    if (freeze==1)
        run = 0;
        freeze = 0;
    end


    drawnow % need this in the loop for controls to work
    pause(0.1);
end
```

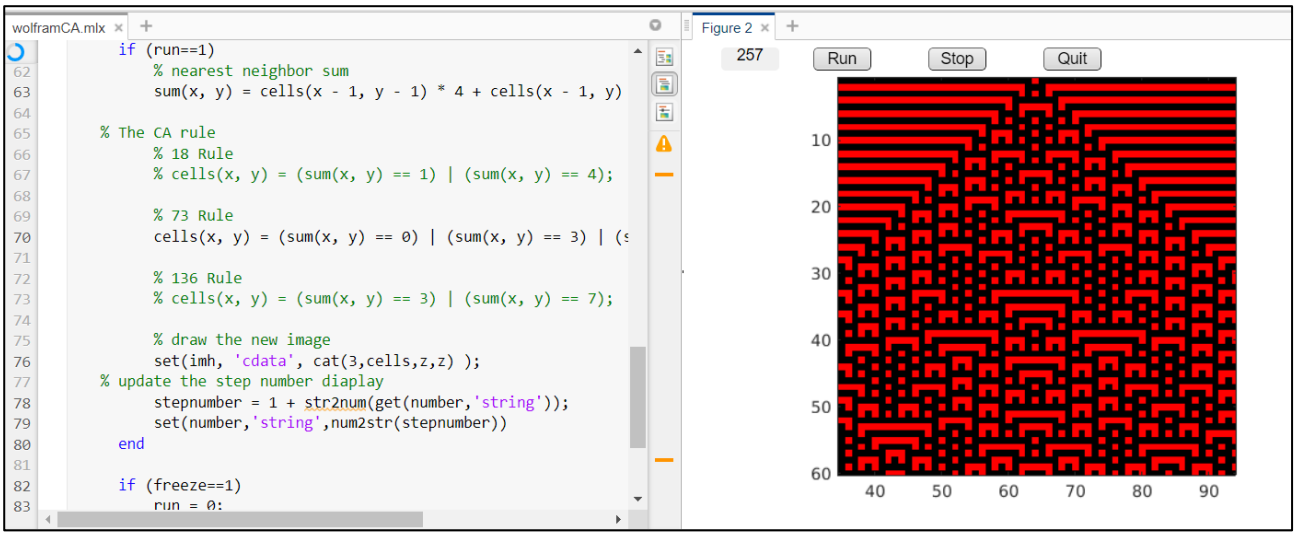
Правило 18



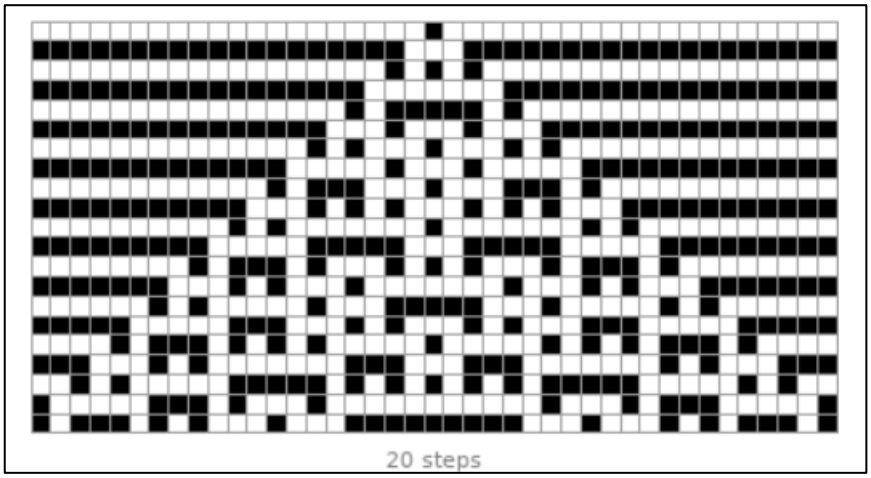
Теоретичний вигляд правила 18



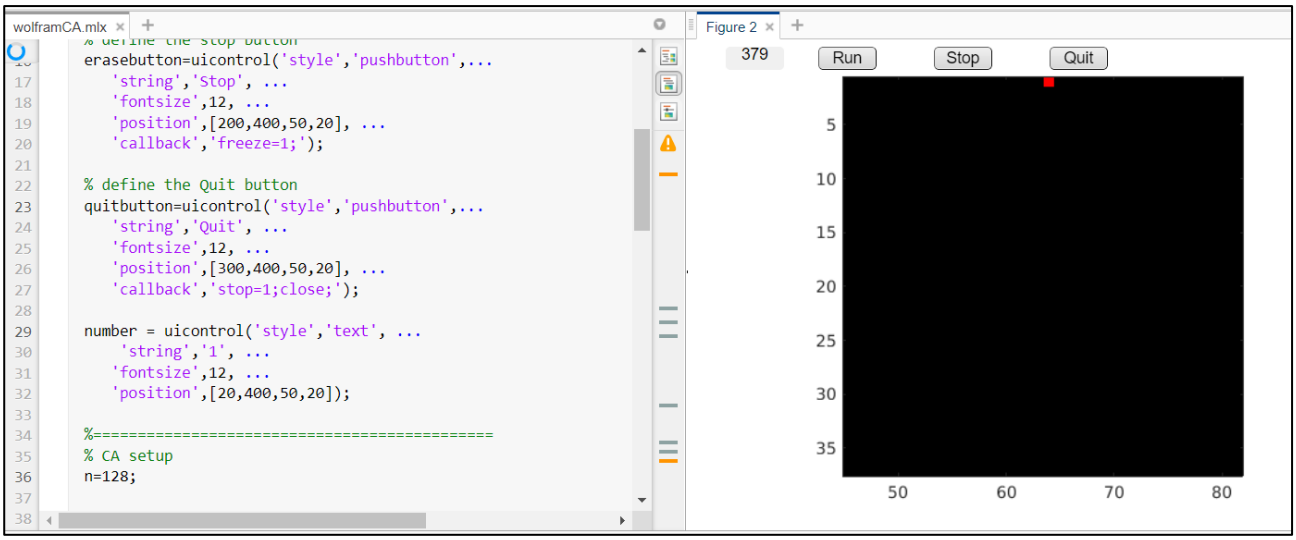
Правило 73



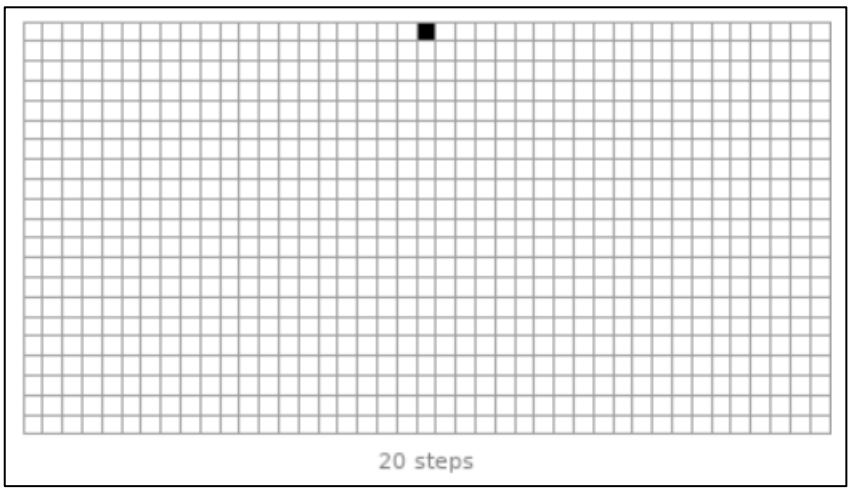
Теоретичний вигляд правила 73



Правило 136



Теоретичний вигляд правила 136



Другий варіант реалізації

wolfram.m

```
% Shendrikov Jack / IO-82 / 01.11.2020
function wolfram(rule, n, width, randfrac)


% check arguments and supply defaults
narginchk(2, 4);
validateattributes(rule, {'numeric'}, {'scalar' 'integer' 'nonnegative' '<=' 255},
'wolfram', 'RULE');
validateattributes(n, {'numeric'}, {'scalar' 'integer' 'positive'}, 'wolfram', 'N');
if nargin < 3 || isempty(width)
    width = 2*n-1;
elseif isscalar(width)
    validateattributes(width, {'numeric'}, {'integer' 'positive'}, 'wolfram', 'WIDTH');
else
    validateattributes(width, {'numeric' 'logical'}, {'binary' 'row'}, 'wolfram',
'START');
end
if nargin < 4 || isempty(randfrac)
    dorand = false;
else
    validateattributes(randfrac, {'double' 'single'}, {'scalar' 'nonnegative' '<=' 1},
'wolfram', 'FNOISE');
    dorand = true;
end


% set up machine
if isscalar(width)
    patt = ones(1, width);
    patt(floor((width+1)/2)) = 2;
else
    patt = width + 1;  % change 0,1 to 1,2 so can use sub2ind
    width = length(patt);
end


% unpack rule
rulearr = (bitget(rule, 1:8) + 1);
```

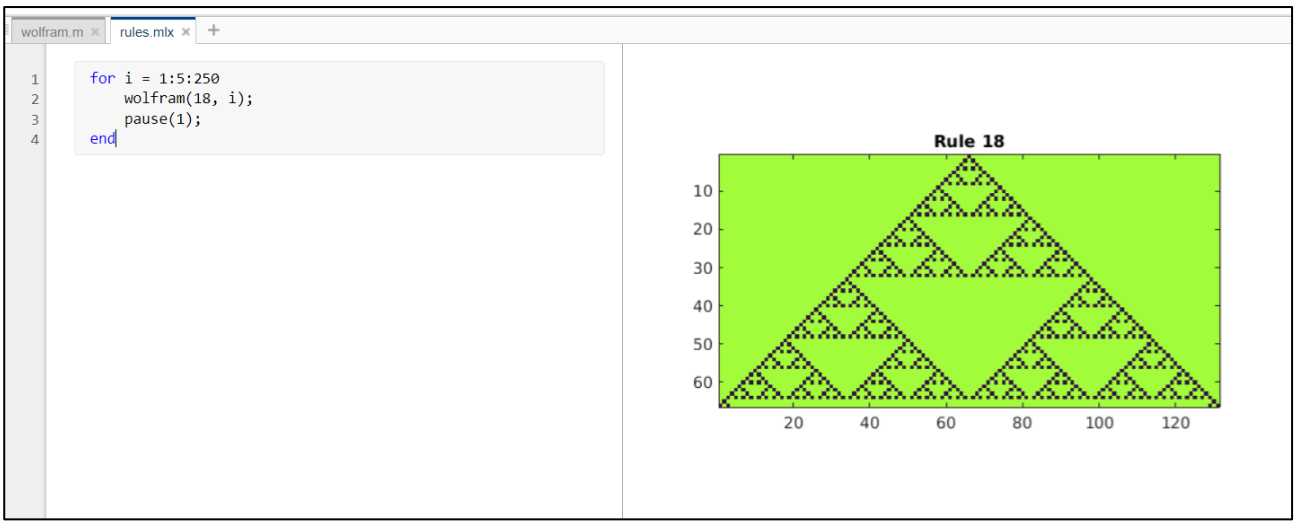
```
% initialise output array
pattern = zeros(n, width);


% iterate to generate rest of pattern
for i = 1:n
    pattern(i, :) = patt;   % record current state in output array

    % core step: apply CA rules to propagate to next 1D pattern
    ind = sub2ind([2 2 2], ...
        [patt(2:end) patt(1)], patt, [patt(end) patt(1:end-1)]);
    patt = rulearr(ind);

    %optional randomisation
    if dorand
flip = rand(1, width) < randfrac;
patt(flip) = 3 - patt(flip);
end
end


% change symbols from 1 and 2 to 0 and 1
pattern = pattern-1;
image(2-pattern); axis image; title("Rule " + rule); colormap(turbo(3));
end
```

Приклад виклику функції

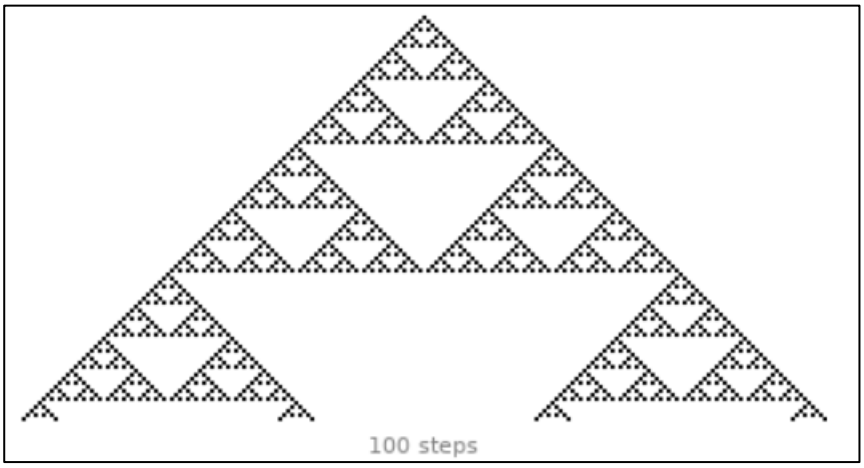
rules.mlx

```
% Shendrikov Jack / IO-82 / 01.11.2020
for i = 1:5:250
    wolfram(18, i); % for Rule 73 -> wolfram(18,i); for Rule 136 -> wolfram(136,i);
    pause(1);
end
```

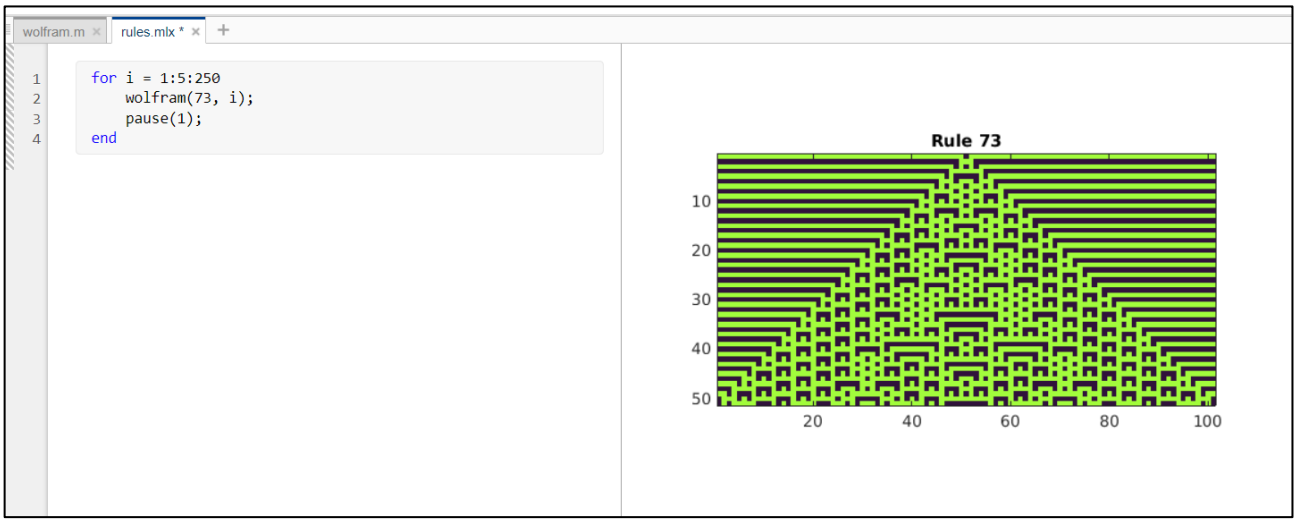
Правило 18



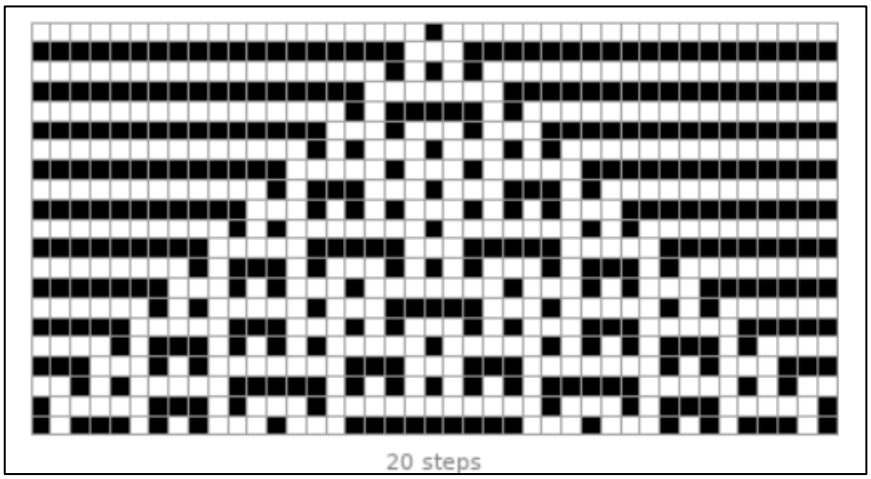
Теоретичний вигляд правила 18



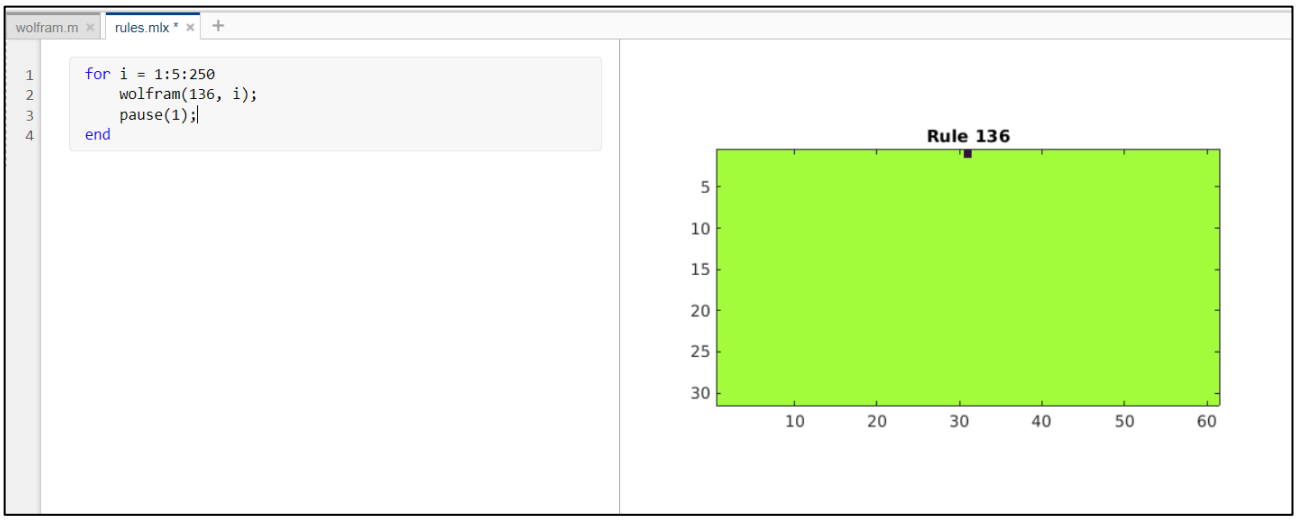
Правило 73



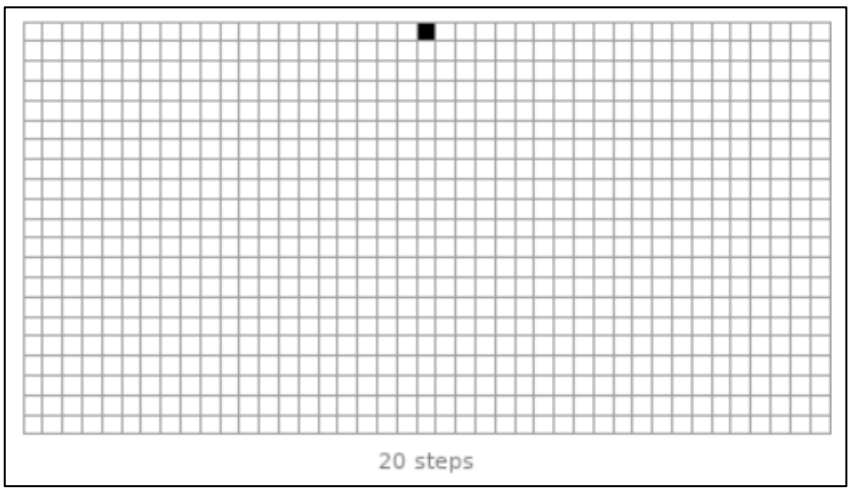
Теоретичний вигляд правила 73



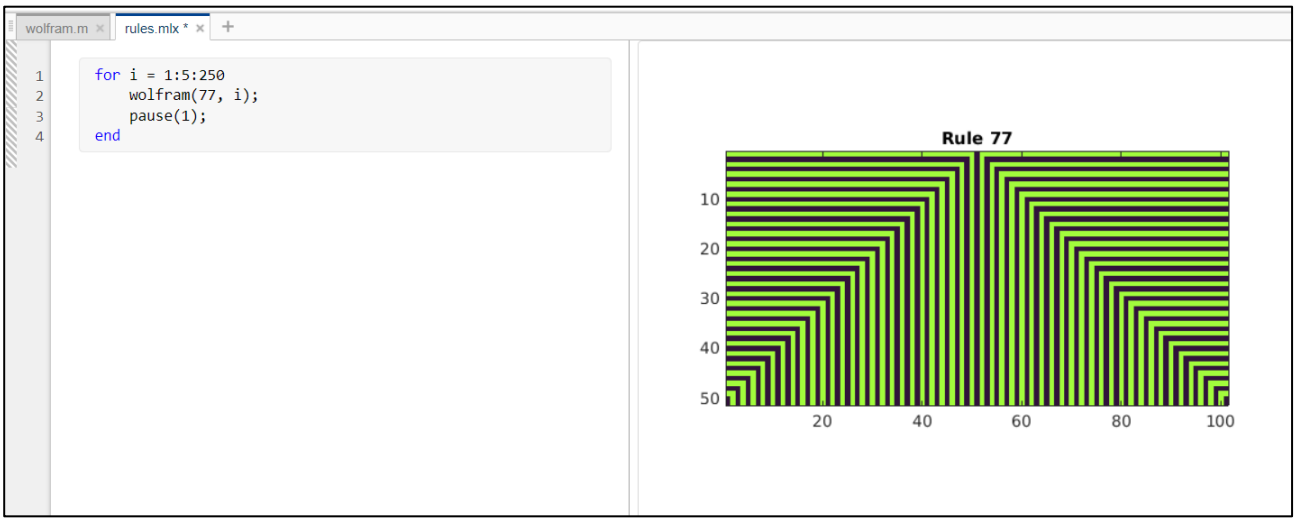
Правило 136



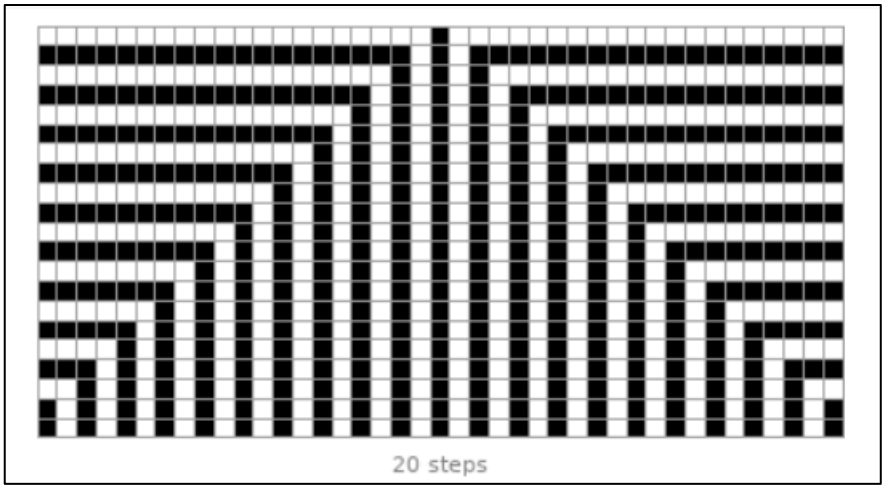
Теоретичний вигляд правила 136



Приклад для правила 77 (додатково для показу функціоналу)



Теоретичний вигляд правила 77



Висновок

В процесі виконання лабораторної роботи було здобуто навички роботи з програмаю Life пакета MatLab, вивчено роботу програми Conway's life, а також розглянуто часову еволюцію правила 00010010 (правила 18), правила 01001001 (правила 73) і правила 10001000 (правила 136).

Також було ознайомлено з процесом моделювання клітинних автоматів. Було запропоновано власний варіант для генерації правил, який виявився набагато ефективнішим, ніж переробка програми Conways`s life, оскільки даний варіант дозволяє будувати будь-яке правило і для цього треба лише змінити число у виклику функції (приклад – *rules.mlx*). Результати збігаються з очікуваними.

Кінцева мета роботи досягнута.