

CAD/CAE

Zadanie 2 - Labirynt

Autor: Jan Augustyn

1. Modyfikacja kodu źródłowego

```
% Code modifications

coeff_nrows = size(coeffs, 1);
coeff_ncols = size(coeffs, 2);
if (coeff_nrows ~= nry || coeff_ncols ~= nrx)
    disp("[Input Error]: It's definitely one of the coeffs of all time:")
    fprintf("\tGot: nrows: %d, ncolumns: %d\n", coeff_nrows, coeff_ncols)
    fprintf("\tExpected: nrows: %d, ncolumns: %d\n\n", nry, nrx)
    return
end

M = zeros(size(X));
for i=1:nrx
    vx=compute_spline(knot_vectorx,px,i,X);
    for j=1:nry
        vy=compute_spline(knot_vectory,py,j,Y);
        M = M + coeffs(nry-j+1, i) * vx .* vy;
    end
end

figure('Name','Maze 3D view');
surf(X,Y, M);

figure('Name','Maze 3D view cropped');
surf(X,Y, M);
zlim([0.7, 1])

% Code modifications end
```

Kod 1 - Modyfikacja kodu funkcji 'spline2Duniform' rysująca trójwymiarowy wykres labiryntu.

2. Wczytywanie danych wejściowych z pliku

```
function spline2D_comp()
data = load('101x101.mat');
spline2Duniform(data.knot_vectorx, data.knot_vectory, data.coeffs);
return
```

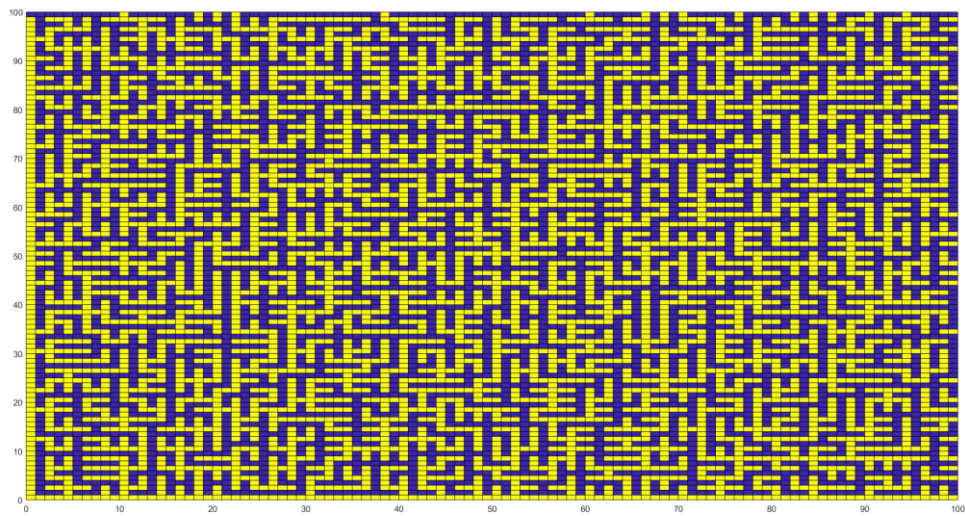
Kod 2 - Kod funkcji spline2D_comp pozwalający na wczytanie danych wejściowych z pliku 'mat'.

3. Kod skryptu generującego macierz i zapisującego dane wejściowe

```
1 from mazelib import Maze
2 from mazelib.generate.Prims import Prims
3 import numpy as np
4 import scipy.io
5 import sys
6
7 def get_knot_vectors(dimension: int, lenght: int):
8     beg = [0 for _ in range(dimension)]
9     mid = [i for i in range(1, lenght-1)]
10    end = [lenght-1 for _ in range(dimension)]
11    return beg+mid+end
12
13 def add_outer_entrances(grid: list[list[int]]):
14     grid[1][0] = 0
15     grid[-2][-1] = 0
16
17 def main():
18     if len(sys.argv) not in (3, 4):
19         print(f"[Usage]: python {sys.argv[0]} [row halls] [col halls] [optional: file name]")
20         return
21
22     row_halls = int(sys.argv[1])
23     col_halls = int(sys.argv[2])
24
25     file_name = "data"
26     if len(sys.argv) == 4:
27         file_name = sys.argv[3]
28
29     m = Maze()
30     m.generator = Prims(row_halls, col_halls)
31     m.generate()
32
33     add_outer_entrances(m.grid)
34
35     vectorx_len = 2 * col_halls + 1
36     vectory_len = 2 * row_halls + 1
37     knot_vectorx = get_knot_vectors(2, vectorx_len)
38     knot_vectory = get_knot_vectors(2, vectory_len)
39
40     with open(file_name + ".txt", "w") as file:
41         file.write(f"knot_vectorx = {knot_vectorx};\n\n")
42         file.write(f"knot_vectory = {knot_vectory};\n\n")
43         file.write(f"coeffs = {np.array2string(np.array(m.grid), separator=' ', threshold=10**10, max_line_width=10*10)};\n")
44
45     knot_vectorx = np.array(knot_vectorx, dtype=np.float64)
46     knot_vectory = np.array(knot_vectory, dtype=np.float64)
47     coeffs = np.array(m.grid, dtype=np.float64)
48
49     scipy.io.savemat(f'{file_name}.mat', {
50         'knot_vectorx': knot_vectorx,
51         'knot_vectory': knot_vectory,
52         'coeffs': coeffs})
53
54     print(f"Maze data successfully saved to {file_name}.txt and {file_name}.mat")
55
56 if __name__ == "__main__":
57     main()
```

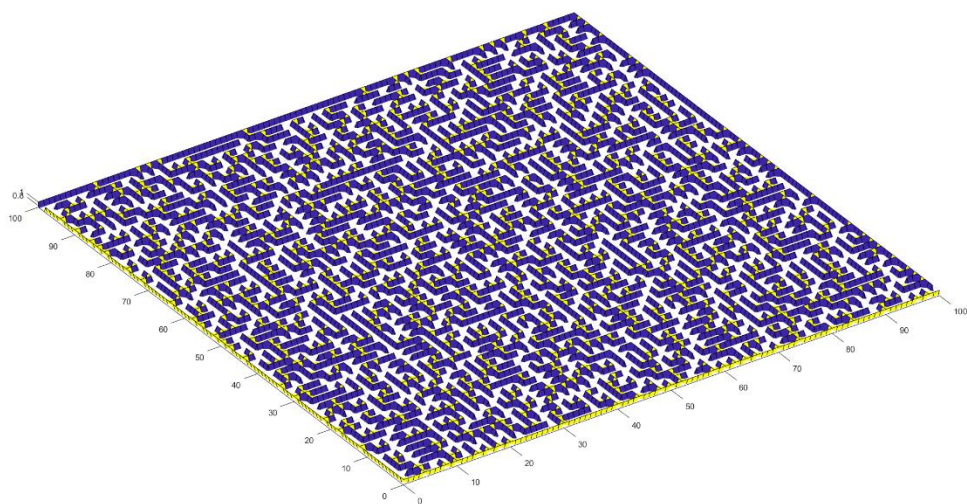
Kod 3 - Skrypt generujący losową macierz o zadanym rozmiarze i zapisujący dane wejściowe w plikach 'mat' i 'txt'.

4. Wygenerowany wykres labiryntu od góry



Wykres 1 - Trójwymiarowy wykres otrzymanego labiryntu z widokiem od góry.

5. Wygenerowany wykres labiryntu od boku



Wykres 2 - Trójwymiarowy wykres otrzymanego labiryntu z widokiem od boku.