

# Zraszacze Ogrodowe

Jakub Grenda  
Kamil Sztandur  
Robert Odrowąż-Sypniewski

15/04/2020

## 1 Cel Projektu

Celem naszego projektu było stworzenie programu, który pomagałby ogrodnikom w optymalnym rozstawieniu zraszaczy ogrodowych na terenie swojego trawnika. Program ten uwzględnia różne kształty terenu oraz obecne przeszkody na nim. Jego zadaniem jest rozstawienie odpowiedniej liczby czterech rodzajów zraszaczy ogrodowych tak, aby trawnik był podlany równomiernie na całej powierzchni. Program informuje użytkownika o ilości rozstawionych zraszaczy, ich koordynatach (oraz typach i kierunkach na konkretnych współrzędnych), średni stopień podlania całego trawnika oraz odchylenie standardowe od niego.

Użytkownik wprowadza do programu kształt swojego trawnika oraz oczekiwaną liczbę cykli obrotów zraszaczy formie, którą opiszemy w dalszej części dokumentacji. Program odczytuje wprowadzone przez użytkownika dane i za pomocą trenowanego algorytmu rozstawia odpowiednio zraszacze. Pod koniec swojej pracy zapisuje stan trawnika do pliku typu png (*Portable Network Graphics*), zawierającego heatmapę stopnia podlania poszczególnych fragmentów trawnika. Program generuje także plik .txt, w którym wypisana zostaje ilość rozstawionych zraszaczy, średnią arytmetyczną podlanego trawnika oraz odchylenie standardowe od średniej.

## 2 Dane wejściowe programu

Dla poprawnego działania programu należy go uruchomić w terminalu podając jeden argument wejściowy będący ścieżką do pliku zawierającego liczbę cykli jaką mają wykonać zraszacze oraz kształt trawnika, na którym mają zostać one rozmieszczone. W pierwszej linii pliku określona jest liczba cykli, w kolejnych 40 liniach o zawierających 80 znaków danych oraz znak nowej linii określony jest kształt trawnika. Znaki danych to symbole '-' lub '\*' gdzie '-' oznacza brak trawnika w danym punkcie natomiast '\*' oznacza jego obecność. Każdy znak odpowiada kwadratowi o wymiarach 100x100 pixeli pliku wyjściowego png.

## 21

2

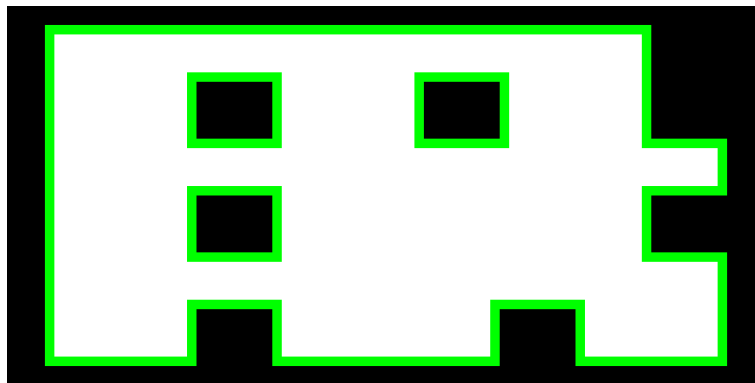
### 3 Przegląd sytuacji wyjątkowych opisywanych przez program.

#### 3.1 Wejście

1. Użytkownik nie podaje żadnego argumentu.
  - Program wyświetla użytkownikowi poprawną instrukcję wywołania programu `help()`.
  - Program przerywa pracę.
2. Użytkownik podaje jako argument nieistniejący plik.
  - Program informuje użytkownika poprzez terminal o niepoprawnym podaniu pliku tekstowego.
  - Program wyświetla użytkownikowi instrukcję `help()`.
  - Program przerywa pracę.
3. Użytkownik podaje jako argument plik niezawierający liczby cykli w pierwszej linijce.
  - Program wyświetla użytkownikowi poprawną instrukcję wywołania programu `help()`.
  - Program przerywa pracę.
4. Użytkownik podaje jako argument plik z ujemną liczbą cykli.
  - Program wyświetla użytkownikowi poprawną instrukcję wywołania programu `help()`.
  - Program przerywa pracę.

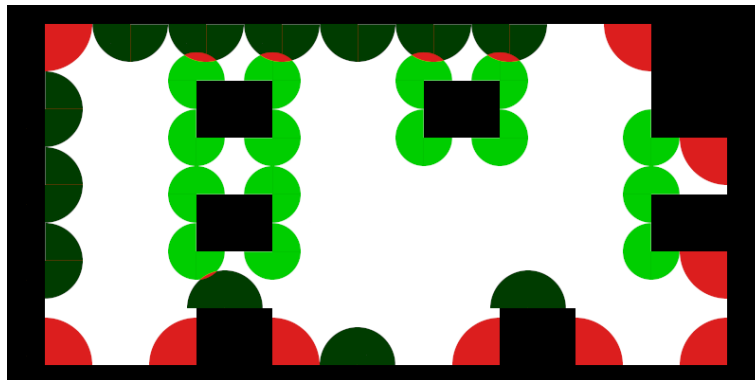
### 4 Algorytm rozmieszczający zraszacze

Działanie algorytmu rozpoczyna się od znalezienia i zaznaczenia w pomocniczej tablicy `int** edges` krawędzi trawnika. Dla zwiększenia wydajności algorytm na tym etapie operuje na danych o rozdzielczości pliku wejściowego tj. 80x40.



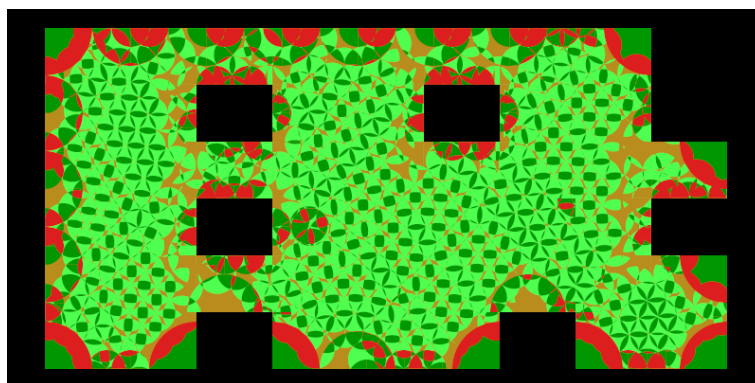
Rysunek 1: Odnalezione krawędzie trawnika

Po odnalezieniu krawędzie algorytm znajduje rogi o kącie  $90^\circ$  i umieszcza w nich zraszacz o takim zakresie podlewania. Następnie w na rogach o kącie  $270^\circ$  umieszczane są odpowiednie zraszacz. Jako ostatnie rozstawiane są zraszacz  $180^\circ$  wzdłuż prostych krawędzi trawnika.



Rysunek 2: Trawnik po rozmieszczeniu krawędziowych zraszaczy

Następnie we wszystkich dotąd niepodlanych punktach ustawiane są zraszacz o zakresie pełnego okręgu rozpoczynając od lewego górnego rogu.



Rysunek 3: Trawnik po wszystkich zraszaczy

W trakcie działania algorytmu wyniki zapisywane są do wektora zawierającego listę zraszaczy oraz stopień podlania trawnika renderowany jest do tablicy, na podstawie której eksportowany jest wyjściowy plik png.

## 5 Opis funkcji udostępnianych przez program

















Program dysponuje wachlarzem funkcji odpowiedzialnych za obsługę plików png, txt, list liniowych, tablic dwuwymiarowych oraz funkcji matematycznych i debugujących.

## 5.1 Lista zraszaczy

1. `location *set_new(location *tab, int x, int y, SprinklerType type, int direction, int *spr_size, int *spr_count)` - funkcja dodając zraszacz do listy na podstawie której generowany jest plik wyjściowy txt zawierający listę pozycji oraz typów zraszaczy.
2. `location *increase(location *tab, int *spr_size)` - funkcja powiększająca dwukrotnie pojemność listy zraszaczy w razie przepełnienia.

## 5.2 Rysowanie zraszaczy po tablicy 2D trawnika

1. `void draw_circle(int **tab, int x, int y, SprinklerType type, Direction direction, int add_value)` - główna funkcja uruchamiająca dany zraszacz. Jej zadaniem jest podlanie znajdującego się w jego zasięgu terenu na podstawie współrzędnych oraz iloczynu kartezjańskiego typu i kierunku zraszacza (*Rysunek 1.*). Funkcja przyjmuje dodatkowo argument `add_value`, który zgodnie ze swoją nazwą wskazuje o ile należy podlać wszystkie komórki w zasięgu zraszacza. Program w głównej funkcji liczy tę wartość na podstawie wskazanej przez użytkownika liczby cykli oraz czasu potrzebnego do obrotu danego typu zraszacza (np. ćwiartka w ciągu 1 cyklu obróci się 4 razy, a pełny kołowy zraszacz tylko raz).

	FULL	THREE QUARTERS	HALF	QUARTER
LEFT				
TOP				
RIGHT				
BOTTOM				

Rysunek 4: Tabelka zraszaczy ze względu na iloczyn kartezjański ich typu i kierunku.

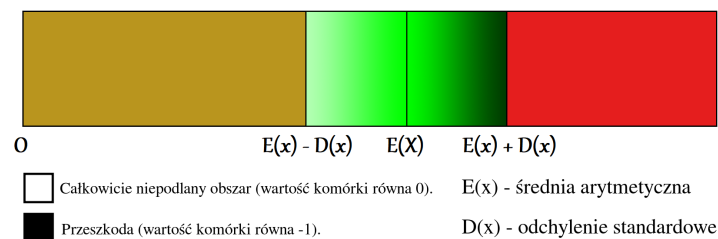
## 5.3 Tworzenie obrazka png

1. `static pixel_t *pixel_at(bitmap_t *bitmap, int r, int c)` - funkcja wskazująca

bieżąco edytowany pixel na obrazku png

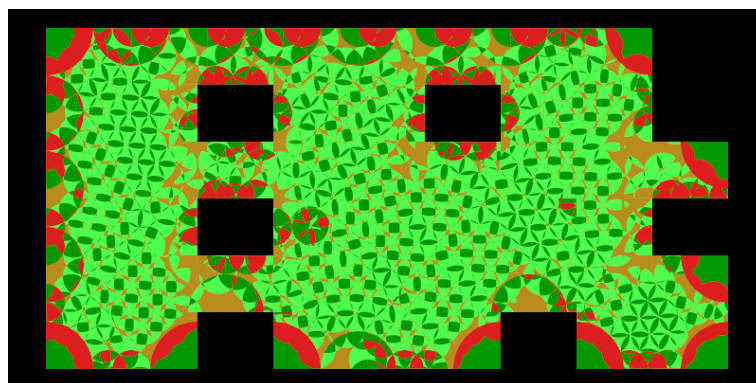
2. `static int save_png_to_file(bitmap_t *bitmap, const char *path)` - funkcja eksportująca pokolorowaną tablicę 2D do pliku PNG
3. `int make_png(int **tab, int height, int width, int count, char *filename, double spr_average, double spr_deviation)` - funkcja główna modułu eksportującego tablicę do pliku png.

Każdy pixel trawnika jest kolorowany w oparciu o wartość odpowiadającą mu komórki w tablicy 2D. Sposób kolorowania jest zależny od wyliczonej średniej arytmetycznej oraz odchylenia standardowego (*Rysunek 2.*).



Rysunek 5: *Metoda kolorowania komórek*

- Wartości pomiędzy przedziału ( $E(x) - D(x)$ ,  $E(x) + D(x)$ ) są gradientem zieleni (jasne od dolnej granicy i ciemne z górnej granicy).
- Wartości poniżej dolnej granicy są koloru zgniło-żółtego ( 185, 140, 30 ) i oznaczają niedostatecznie podlane komórki w porównaniu do całości.
- Wartości powyżej górnej granicy są koloru czerwonego (220, 30, 30) i oznaczają zbyttnio podlane komórki w porównaniu do całości.

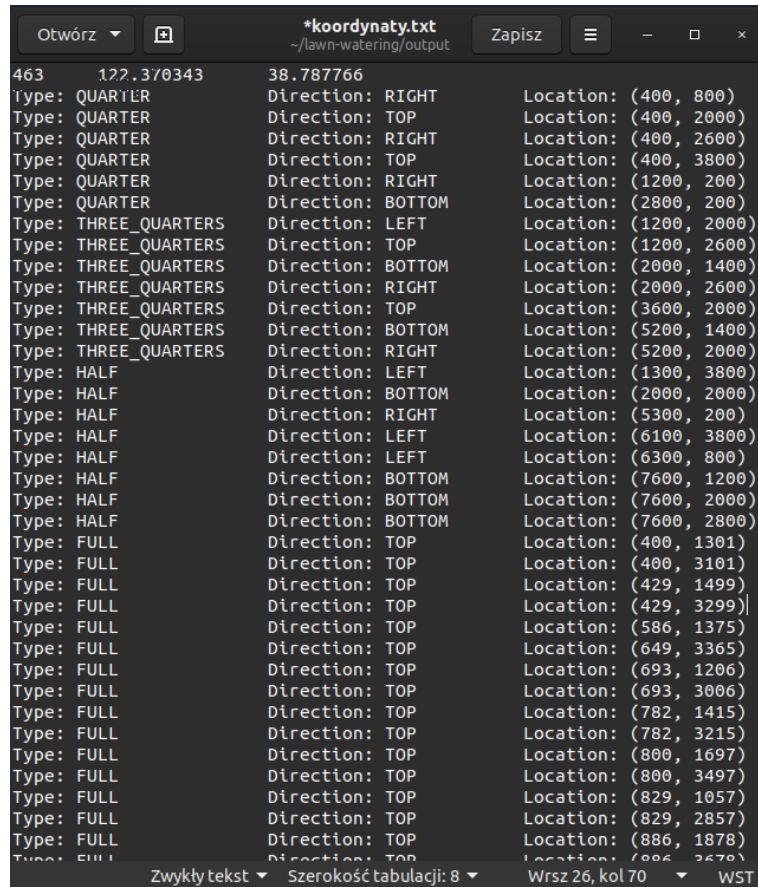


Rysunek 6: Przykładowy wynik renderowania

## 5.4 Tworzenie pliku txt:

1. `void make_txt(location *sprinklers, int count, char *filename, double spr_average, double spr_deviation, int spr_count)` - funkcja wypisująca listę zraszaczy, ich ilość, średnią arytmetyczną poziomu nawodnienia oraz odchylenie standardowe od średniej do pliku txt.

## Przykładowy plik wyjściowy



```
463      122.370343      38.787766
Type: QUARTER      Direction: RIGHT      Location: (400, 800)
Type: QUARTER      Direction: TOP      Location: (400, 2000)
Type: QUARTER      Direction: RIGHT      Location: (400, 2600)
Type: QUARTER      Direction: TOP      Location: (400, 3800)
Type: QUARTER      Direction: RIGHT      Location: (1200, 200)
Type: QUARTER      Direction: BOTTOM      Location: (2800, 200)
Type: THREE_QUARTERS      Direction: LEFT      Location: (1200, 2000)
Type: THREE_QUARTERS      Direction: TOP      Location: (1200, 2600)
Type: THREE_QUARTERS      Direction: BOTTOM      Location: (2000, 1400)
Type: THREE_QUARTERS      Direction: RIGHT      Location: (2000, 2600)
Type: THREE_QUARTERS      Direction: TOP      Location: (3600, 2000)
Type: THREE_QUARTERS      Direction: BOTTOM      Location: (5200, 1400)
Type: THREE_QUARTERS      Direction: RIGHT      Location: (5200, 2000)
Type: HALF      Direction: LEFT      Location: (1300, 3800)
Type: HALF      Direction: BOTTOM      Location: (2000, 2000)
Type: HALF      Direction: RIGHT      Location: (5300, 200)
Type: HALF      Direction: LEFT      Location: (6100, 3800)
Type: HALF      Direction: LEFT      Location: (6300, 800)
Type: HALF      Direction: BOTTOM      Location: (7600, 1200)
Type: HALF      Direction: BOTTOM      Location: (7600, 2000)
Type: HALF      Direction: BOTTOM      Location: (7600, 2800)
Type: FULL      Direction: TOP      Location: (400, 1301)
Type: FULL      Direction: TOP      Location: (400, 3101)
Type: FULL      Direction: TOP      Location: (429, 1499)
Type: FULL      Direction: TOP      Location: (429, 3299)
Type: FULL      Direction: TOP      Location: (586, 1375)
Type: FULL      Direction: TOP      Location: (649, 3365)
Type: FULL      Direction: TOP      Location: (693, 1206)
Type: FULL      Direction: TOP      Location: (693, 3006)
Type: FULL      Direction: TOP      Location: (782, 1415)
Type: FULL      Direction: TOP      Location: (782, 3215)
Type: FULL      Direction: TOP      Location: (800, 1697)
Type: FULL      Direction: TOP      Location: (800, 3497)
Type: FULL      Direction: TOP      Location: (829, 1057)
Type: FULL      Direction: TOP      Location: (829, 2857)
Type: FULL      Direction: TOP      Location: (886, 1878)
Type: FULL      Direction: TOP      Location: (886, 3678)
```

## 5.5 Obliczanie średniej oraz odchylenia standardowego:

1. `average()` - funkcja zwracająca średnią arytmetyczną wartości na tablicy posługująca się wzorem:

$$E(x) = \frac{\sum_{i=1}^n x_i}{n}$$

gdzie  $x_i$  to kolejne wartości komórek tablicy, a  $n$  to liczba wszystkich komórek.

2. `std_deviation()` - funkcja zwracająca odchylenie standardowe na tablicy posługująca się wzorem:

$$D(x) = \sqrt{\frac{\sum_{i=1}^n (x_i - E(x))^2}{n}}$$

gdzie  $E_{(x)}$  to średnia arytmetyczna.

## 5.6 Pomoc:

1. `help()` - funkcja wyświetlająca instrukcję prawidłowego wywołania programu.

## 5.7 Kolorowanie:

1. Wartość podlania komórki wykracza ponad 255 (maksymalna wartość składowej RGB).
  - Nie jest to żadnym problemem, ponieważ program opiera się na średniej arytmetycznej i odchyleniu standardowym. Gradient pomiędzy górną granicą, a dolną ma zabarwienie proporcjonalne. Wartość komórek zostaje przeskalowana względem tego przedziału.
2. Średnia arytmetyczna jest mniejsza niż odchylenie standardowe (wartości dolnego przedziału wchodzą na minus).
  - Program jako dolną granicę przyjmuje 0.