

Zraszacze Ogrodowe

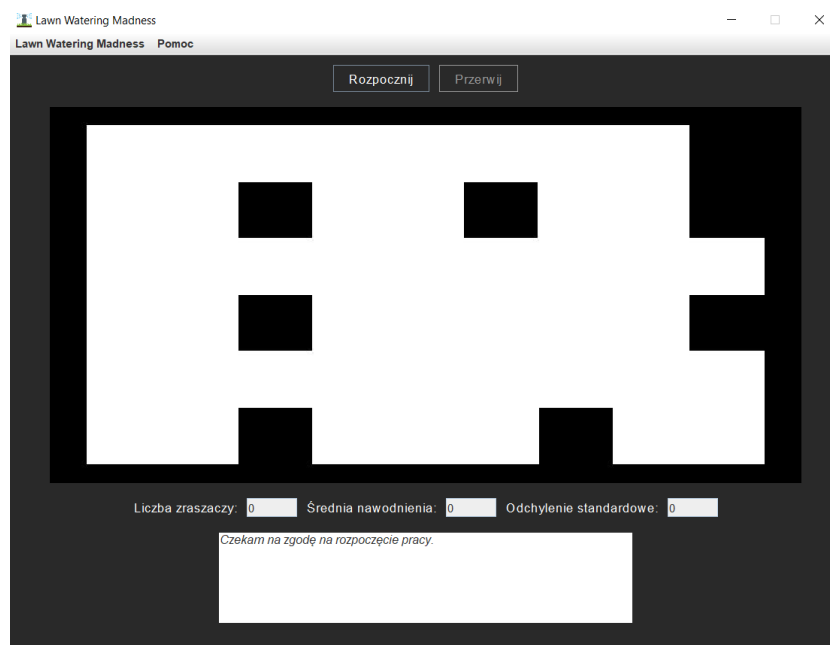
Jakub Grenda
Kamil Sztandur
Robert Odrowąż-Sypniewski

31/05/2020

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.

Najnowsza wersja aplikacji pozwala także na utworzenie specjalnej animacji, symulującej proces podlewania całego trawnika, która jest wyświetlana w czasie pracy programu, a następnie zapisywana w formacie GIF.



Rysunek 1: Podgląd okna aplikacji

Użytkownik wprowadza do programu dane potrzebne do przeprowadzenia symulacji i pracy algorytmu rozstawiającego zraszacz. W czasie pracy programu użytkownik w specjalnym oknie jest cały czas informowany o procesach przebiegających w programie - bieżący etap pracy, dane właśnie rozstawionego zraszacza, numer aktualnie wyświetlanego cyklu (klatki), ewentualne komunikaty o problemach - oraz w specjalnych komórkach widoczna jest ilość rozstawionych przez algorytm zraszaczy, aktualna średnia arytmetyczna podlania całego trawnika oraz odchylenie standardowe od niej. Po zakończeniu pracy program generuje dane dot. rozstawienia zraszaczy, stanu trawnika oraz pozwalające na ponowne prześledzenie symulacji w razie potrzeby. Pomoc znajduje się w zakładce "Pomoc" na pasku menu w oknie aplikacji.

2 Dane wejściowe programu.

Po uruchomieniu aplikacji użytkownik wskazuje w oknie ustawień plik tekstowy na dysku, który zawiera kształt trawnika. Następnie wprowadza **czas trwania pojedynczych cykli**, które są jednostką czasu pracy naszych zraszaczy oraz okresem wyświetlania kolejnych klatek animacji. W przypadku potrzeby podania wartości niecałkowitej część ułamkową od jedności należy oddzielić **kropką**. Kolejnym parametrem jest **ilość cykli**, które mają być symulowane (co jest równoznaczne z ilością klatek animacji). Każdy typ zraszacza potrzebuje inną ilość czasu dla swojego pełnego obrotu. Istnieje możliwość zaznaczenia opcji, aby program uwzględniał w symulacji **odbicia wytryskiwanej wody od powierzchni przeszkód**, co zwiększa wiarygodność wyników symulacji, ale również poważnie wydłuża czas pracy aplikacji i renderowania kolejnych klatek, co może spowodować opóźnienia wyświetlania kolejnych klatek. Aby rozwiązać ten problem, po zakończeniu pracy program generuje **animację GIF**, w której czas pomiędzy kolejnymi klatkami jest zawsze zgodny z czasem wprowadzonym przez użytkownika.

W kolejnych **40 liniach** wejściowego pliku tekstowego zawierających **80 znaków** danych oraz znak nowej linii określony jest kształt trawnika. Znaki danych to symbole '-' lub '*', gdzie '-' oznacza brak trawnika (lub przeszkodę) w danym punkcie natomiast '*' oznacza jego obecność. Każdy znak odpowiada kwadratowi o wymiarach 100x100 pixeli pliku wyjściowego jpg.

[illegible]

3 Przegląd sytuacji wyjątkowych opisywanych przez program.

3.1 Wejście.

1. Użytkownik podaje nieprawidłową wartość ilości lub długości cykli.
 - Program wyświetla użytkownikowi okno z upomnieniem i informacją o prawidłowym uzupełnieniu parametrów.
 - Po zamknięciu okienka z upomnieniem program wraca do okna ustawień.
2. Użytkownik zapomina podać plik tekstowy z trawnikiem.
 - Program informuje użytkownika poprzez okienko o potrzebie podania pliku tekstowego.
 - Po zamknięciu okienka z upomnieniem program wraca do okna ustawień.

3.2 Przetwarzanie danych wprowadzonych przez użytkownika.

1. Po zaakceptowaniu podania pliku przez użytkownika, plik zostaje usunięty lub występują problemy z jego rozczytaniem.
 - Program wyświetla użytkownikowi komunikat o zaistniałym wariancie błędu.
 - Program przerywa pracę, zachowując pliki konfiguracyjne dla przyszłej analizy.

3.3 Renderowanie animacji.

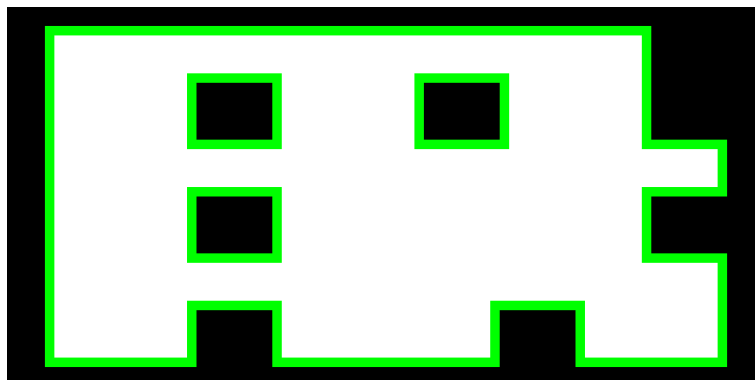
1. Komputer użytkownika nie nadąża z renderowaniem kolejnych klatek animacji w porównaniu do czasu, jaką mu na to użytkownik wyznaczył.
 - Program wyświetla komunikat o zaistniałym opóźnieniu w okienku logów.
 - Program kontynuuje pracę.

3.4 Ogólna praca programu.

1. W czasie pracy program z nieprzewidzianych przez programistów przyczyn napotkał krytyczny błąd i został zamknięty w czasie pracy.
 - Program przerywa pracę.
 - Dotychczasowe wyrenderowane klatki animacji znajdują się w katalogu zawierającym plik .jar w folderze resources w podfolderze temp.

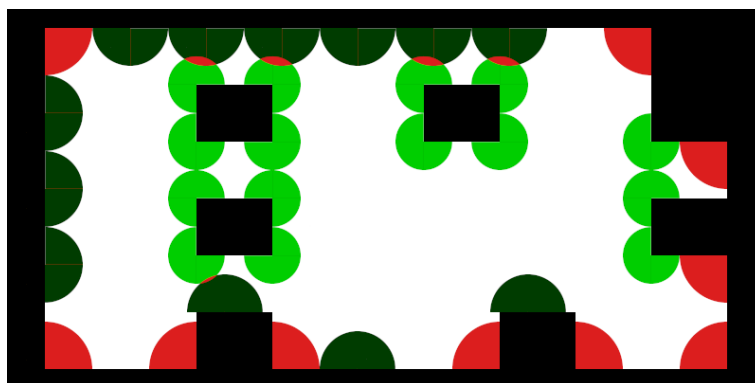
4 Algorytm rozmieszczający zraszacze

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.



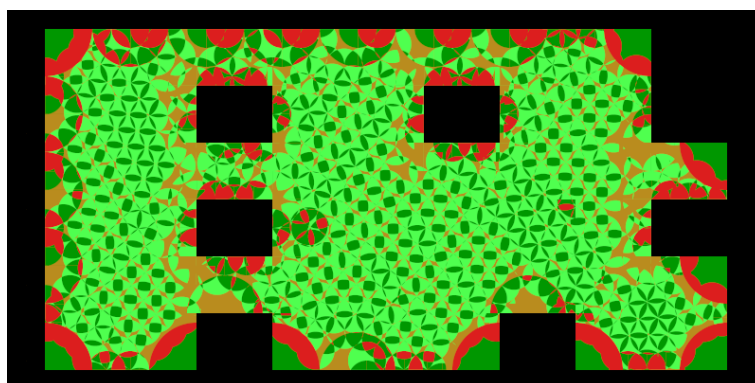
Rysunek 2: Odnalezione krawędzie trawnika

Po odnalezieniu krawędzi algorytm znajduje rogi o kącie 90° i umieszcza w nich zraszacze o takim zakresie podlewania. Następnie w na rogach o kącie 270° umieszczane są odpowiednie zraszacze. Jako ostatnie rozstawiane są zraszacze 180° wzdłuż prostych krawędzi trawnika.



Rysunek 3: Trawnik po rozmieszczeniu krawędziowych zraszaczy

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



Rysunek 4: Trawnik po wszystkich zraszaczach

W trakcie działania algorytmu kolejne zraszacze są dopisywane do `ArrayList` zawierającej obiekty typu **Sprinkler** określające położenie (**koordynaty x, y**) zraszacza, jego **typ** (`SprinkleType`) oraz **kierunek** (`Direction`). Następnie program na jej podstawie rozpoczyna podlewanie trawnika oraz po ukończeniu pracy wypisuje ją do pliku tekstowego.

5 Opis funkcji udostępnianych przez program.

















Program dysponuje wachlarzem obiektów odpowiedzialnych za obsługę plików jpg, txt, gif, interfejsów graficznych Java Swing oraz potrzebnych funkcji matematycznych.

5.1 Config.

Najważniejszy obiekt informacyjny, którego wymagają prawie wszystkie obiekty w programie do prawidłowej pracy. Tworzy on i zarządza plikiem konfiguracyjnym, który zawiera ścieżki, ustawienia użytkownika i inne ważne informacje dla prawie wszystkich pozostałych obiektów w programie. Obiekt zawiera wyłącznie funkcje statyczne.

1. **void createNewProperties()** - Funkcja tworząca plik konfiguracyjny typu Properties i przypisująca mu początkowe wartości.
2. **void saveToFile(Properties config)** - Funkcja zapisująca właściwości do pliku tekstowego (używana w createNewProperties).
3. **void deleteFile()** - Funkcja kasująca tekstowym plik konfiguracyjny z katalogu programu.
4. **Properties readFile()** - Funkcja czytająca tekstowy plik konfiguracyjny i zwracająca obiekt typu Properties w nim zawarty.
5. **void set(String key, String value)** - Funkcja zapisująca lub nadpisująca pozycję ustawień w tekstowym pliku konfiguracyjnym.
6. **void get(String key)** - Funkcja pobierająca pozycję ustawień w tekstowym pliku konfiguracyjnym.
7. **String getConfigFilename()** - Funkcja zwracająca nazwę tekstowego pliku konfiguracyjnego.

5.2 Sprinkler.

	FULL	THREE QUARTERS	HALF	QUARTER
LEFT				
TOP				
RIGHT				
BOTTOM				

Rysunek 5: *Tabela zraszaczy w postaci iloczynu kartezjańskiego ich typu i kierunku.*

Obiekt odpowiadający za zarządzanie zraszaczami, listą zraszaczy i zapewniający interfejs ułatwiający operacje na nich np. przy algorytmie.

1. **Sprinklers** - Interfejs zawierający typy wyliczeniowe dla typów (SprinklerType) i kierunków (Direction) zraszaczy, przyporządkujący im wartości numeryczne oraz zawierający informacji o okresach obrotu i promieniach poszczególnych zraszaczy.
2. **Sprinkler** - Pojedynczy obiekt zraszacza. Przechowuje jego koordynaty (x, y), kierunek (Direction) oraz typ (SprinklerType)
3. **SprinklersList** - Obiekt stanowiący listę zraszaczy oparty na ArrayList oraz opatrzony funkcją "add", która oprócz dodania zraszacza do ArrayListy wykonuje wszystkie inne potrzebne w programie akcje przy dodaniu zraszacza (np. wypisanie informacji o tym na interfejs graficzny).

5.3 Stats.

Obiekt zawierający wszystkie bardziej zaawansowane funkcje matematyczne dostosowane pod podawane im przez ten program obiekty głównie tablice dwuwymiarowe.

1. **double avg (int lawn)** - Funkcja obliczająca średnią arytmetyczną z przyjmowanej tablicy dwuwymiarowej, zwraca -1 w przypadku wystąpienia wyjątku ArithmeticException. Posługuje 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. **double std_deviation (int[][] lawn, double average)** - Funkcja obliczająca odchylenie standardowe z przyjętej tablicy dwuwymiarowej i średniej arytmetycznej. Posługuje się wzorem:

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

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

3. **double std_deviation(int[][] lawn)** - Przeciążenie funkcji obliczającej odchylenie standardowe z przyjętej tablicy dwuwymiarowej, ale przy braku średniej arytmetycznej.

5.4 WindowManager.

Statyczny obiekt stanowiący ogólny podręczny menedżer okien interfejsu graficznego użytkownika

1. **void waitUntilClosed(JFrame Window)** - statyczna funkcja, która każe bieżącemu wątkowi czekać dopóki użytkownik nie zamknie okna interfejsu graficznego Java Swing podanego jako parametr.

5.5 WindowMenuBar.

Obiekt paska menu w oknach interfejsu graficznego na bazie Java Swing.

1. **WindowMenuBar** - Interfejs graficzny panelu menu w oknie programu wraz z całą logiką i nasłuchiwaniami.
2. **InfoWindow** - Niepubliczna klasa przechowująca w obiekcie wyskakujące okna z informacjami po wybraniu którejkolwiek z opcji w pasku menu. Jako argument przyjmuje tytuł okna, informacja która ma zostać wypisana w oknie oraz ścieżkę do obrazka przykładowego (może być null - wtedy nie będzie obrazka).

5.6 SettingsWindow.

Obiekt przechowujący okno ustawień wyskakujące na początku programu. Współpracuje z obiektem Settings, która obsługuje nasłuchiwanie i komunikuje się z plikami konfiguracyjnymi. Zawiera szereg funkcji renderujących poszczególne panele okna.

5.7 Settings.

Obiekt przechowujący logikę okna ustawień SettingsWindow. Obsługuje nasłuchiwanie, weryfikuje poprawność i zapisuje wprowadzone przez użytkownika dane do pliku konfiguracyjnego config.txt.

1. **void actionPerformed((ActionEvent event)** - Funkcja obsługująca naciśnięcie przez użytkownika przyciski w oknie ustawień.
2. **void itemStateChanged(ItemEvent e)** - Funkcja obsługująca zaptaszkowanie checkboxa przez użytkownika.
3. **private void modifyField(JTextField field, int mode)** - Funkcja modyfikowania pola w oknie z ilością cykli za pomocą przycisków wokół.
4. **private void getFile()** - Funkcja otwierająca okno wyszukiwarki plików w systemie operacyjnym, pozwalająca w wygodny sposób wskazać użytkownikowi plik wejściowy.

5.8 ResultWindow.

Obiekt przechowujący okno finałowe wyskakujące na zakończenie pracy programu, informująca o jego przebiegu i wyniku. Zawiera szereg funkcji renderujących poszczególne panele okna.

5.9 Result.

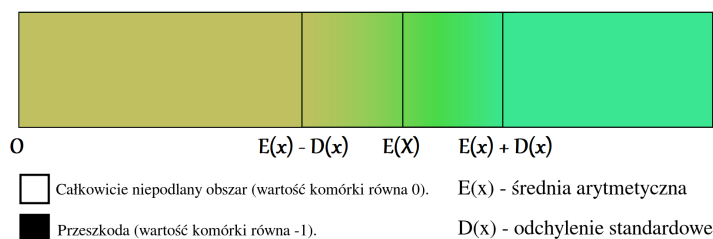
Obiekt przechowujący logikę okna finałowego ResultWindow. Obsługuje nasłuchiwanie kliknięcia przycisku przez użytkownika.

1. **void actionPerformed((ActionEvent event)** - Funkcja obsługująca naciśnięcie przez użytkownika przyciski w oknie finałowym.

5.10 Output.

Obiekt obsługujący całe potrzebne wyjście w programie.


1. **void export_jpg(int[], double average, double stdv, String img_filename)** - Funkcja eksportująca obrazek jpg z tablicy dwuwymiarowej, korzystająca z obiektu tłumaczącego kolory HSV na RGB i kolorująca w oparciu o średnią arytmetyczną i odchylenie standardowe, a następnie tworząca obrazek jpg o podanej nazwie.
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 6: *Metoda kolorowania komórek*

- Wartości spomiędzy przedziału $(E(x) - D(x), E(x) + D(x))$ są gradientem zieleni i oznaczają prawidłowe wartości nawodnienia trawnika.

- Wartości poniżej dolnej granicy są koloru zgniło-żółtego i oznaczają niedostatecznie podlane komórki w porównaniu do całości.
 - Wartości powyżej górnej granicy są koloru mokrej zieleni i oznaczają zbyt dużo podlane komórki w porównaniu do całości.
2. **void export_txt(ArrayList<Sprinkler> sprinklers_list)** - Funkcja eksportująca plik tekstowy "koordynaty.txt" z listą zraszaczy podaną jako parametr.



Type: FULL	Direction: LEFT	Location: (5360, 892)
Type: FULL	Direction: TOP	Location: (6178, 2711)
Type: FULL	Direction: LEFT	Location: (5407, 892)
Type: FULL	Direction: TOP	Location: (6128, 2300)
Type: HALF	Direction: RIGHT	Location: (3190, 1010)
Type: FULL	Direction: RIGHT	Location: (2094, 1844)
Type: FULL	Direction: LEFT	Location: (577, 969)
Type: THREE_QUARTERS	Direction: TOP	Location: (2978, 1126)
Type: HALF	Direction: TOP	Location: (6558, 1797)
Type: HALF	Direction: TOP	Location: (5053, 695)
Type: THREE_QUARTERS	Direction: LEFT	Location: (7236, 3181)
Type: FULL	Direction: TOP	Location: (4472, 1447)
Type: HALF	Direction: RIGHT	Location: (2956, 1213)
Type: FULL	Direction: RIGHT	Location: (4465, 1418)
Type: HALF	Direction: LEFT	Location: (5075, 2586)
Type: FULL	Direction: RIGHT	Location: (2875, 3359)
Type: THREE_QUARTERS	Direction: TOP	Location: (6279, 1135)

Rysunek 7: Przykładowy tekstowy plik wyjściowy.

3. **private void writeLine(FileOutputStream stream, String lineToWrite)** - Funkcja prywatna pomocna przy wypisywaniu listy do pliku.
4. **boolean create_dir(String dirname)** - Funkcja tworząca folder o podanej nazwie (jeżeli jeszcze taki nie istnieje) i zwracająca true, jeżeli udało się utworzyć folder.

5.11 MainWindowEvents.

Obiekt obsługujący logikę głównego okna interfejsu graficznego, gdzie wyświetlana jest animacja, logi programu oraz wartości liczbowe w programie. Użytkownik może rozpocząć, wstrzymać i wznowić pracę programu z tego poziomu. To stąd startuje wątek animacji. Obsługuje nasłuchiwanie MainWindow.

1. **void actionPerformed((ActionEvent event)** - Funkcja obsługująca naciśnięcie przez użytkownika przyciski w oknie. Pierwsze uruchomienie animacji z komendy "Rozpocznij" rozpoczyna rozstawianie zraszaczy przez algorytm, a następnie startuje wątek animacji. Przyciski "Wznów" i "Zatrzymaj" pozwalają na zarządzanie programem.

5.12 MainWindow.

Obiekt przechowujący główne okno interfejsu graficznego, gdzie wyświetlana jest animacja, logi programu oraz wartości liczbowe w programie. Zawiera nasłuchiwanie, które są obsługiwane przez MainWindowEvents, funkcje renderujące kolejne panele w oknie oraz funkcje pomocnicze do modyfikowania wartości znajdujących się w oknie (średnia, odchylenie, ilość zraszaczy, dodawanie logów programu).

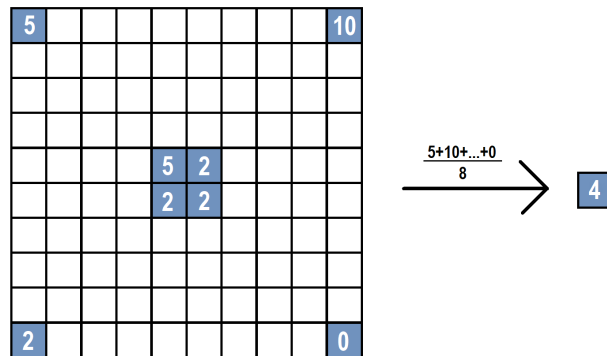
1. **void printNewAction(String action)** - Funkcja dopisująca do logów w oknie przekazaną jako parametr typu String akcję.
2. **void updateSprinklerAmount(int n)** - Funkcja ustawiająca liczbę przekazaną poprzez parametr jako wyświetlaną ilość zraszaczy w oknie.
3. **void updateAverage()** - Funkcja aktualizująca wyświetlaną w oknie średnią arytmetyczną z tablicy dwuwymiarowej trawnika, korzystająca z obiektu Lawn, którego wskaźnik jest zapisany w tym obiekcie MainWindow.
4. **void updateStdv()** - Funkcja aktualizująca wyświetlane w oknie odchylenie standardowe z tablicy dwuwymiarowej trawnika, korzystająca z obiektu Lawn, którego wskaźnik jest zapisany w tym obiekcie MainWindow. Warto poprzedzić tę funkcję powyższą funkcją updateAverage podczas użycia, ponieważ zakłada ona aktualność średniej arytmetycznej.

5.13 Lawn.

Obiekt trawnika, zawierający tablicę dwuwymiarową 4000x8000 jego stanu, 40x80 jego kształtu (na potrzeby algorytmu) oraz zmniejszonych rozmiarów kopię 400x800 na potrzeby wyświetlania animacji ze względu na problemy wydajnościowe przy renderingu animacji 8k. Oprócz tego przechowuje także stałe wielkości poszczególnych tablic oraz średnią i odchylenie standardowe z trawnika 4000x8000.

Obiekt inicjalizuje się bez żadnych parametrów. Korzysta on informacji pliku config.exe zmodyfikowanego przez obiekt Settings.

1. **void parseShape()** - Tworząca tablicę dwuwymiarową 40x80 w obiekcie Lawn, będąca odwzorowaniem kształtu trawnika. Potrzebne do algorytmu.
2. **void upscaleShape()** - Funkcja skalująca tablicę dwuwymiarową 40x80 kształtu trawnika na rzeczywistą wielkość 4000x8000, na której będzie pracować program.
3. **void update_gif_lawn()** - Funkcja aktualizująca tablicę 400x800 trawnika dla animacji na podstawie oryginalnej tablicy 4000x8000. Wykorzystuje do algorytm dzielący oryginalną tablicę na pomniejsze tablice 10x10 i liczący średnią z ich środkowych i narożnikowych komórek. Następnie przypisuje tę średnią jednej komórce odpowiadającej im położeniem na tablicy 400x800. W ten sposób ta komórka w przybliżony sposób reprezentuje tamtą tablicę 10x10 w optymalny sposób, ponieważ liczenie średniej z całej tablicy 10x10 zajmuje zbyt dużo czasu jak na trwającą animację.



Rysunek 8: Sposób skalowania dużej tablicy trawnika na mniejszą, używaną do wyświetlania animacji.

4. **void updateAverage()** - Funkcja aktualizująca przechowywaną w tym obiekcie średnią arytmetyczną z tablicy dwuwymiarowej trawnika, korzystająca z obiektu Stats zawierającego potrzebne obliczenia matematyczne.
5. **void updateStdv()** - Funkcja aktualizująca przechowywane w tym obiekcie odchylenie standardowe z tablicy dwuwymiarowej trawnika, korzystająca z obiektu Stats zawierającego potrzebne obliczenia matematyczne. Warto poprzedzić tę funkcję powyższą funkcją updateAverage podczas użycia, ponieważ zakłada ona aktualność średniej arytmetycznej.

5.14 HSVtoRGB.

Statyczny obiekt tłumaczący wartości HSV na RGB.

1. **int[] calc(double h, double s, double v)** - Funkcja tłumacząca wartości HSV na RGB i zwracająca jednowymiarową tablicę. Parametr h przyjmuje wartości z przedziału [0, 360], parametr s z przedziału [0, 1] oraz parametr v z przedziału [0, 1].

5.15 GifSequenceWriter.

Biblioteka zawierająca metody tworzące animację GIF z utworzonych klatek w trakcie pracy programu. Autorem tej biblioteki jest twórca strony <https://memorynotfound.com>. Zamieścił ją tam do pobrania i użytku publicznego.

5.16 GIF.

Obiekt przechowujący wątek odpowiedzialny za wyświetlanie animacji w oknie aplikacji i tworzenie kolejnych klatek animacji, zawierający również narzędzia do utworzenia pliku GIF na końcu pracy programu.

1. **void run()** - Funkcja główna wątku animacji, odpowiedzialna za jej prawidłową pracę i wyświetlanie. Może być wstrzymany z poziomu interfejsu graficznego okna głównego MainWindow.

2. **void generateGIF()** - Funkcja tworząca animację GIF z klatek utworzonych w czasie pracy wątku animacji. Korzysta z obiektów `GifSequenceWriter` oraz `HSVtoRGB`.
3. **void createWorkdir()** - Funkcja tworząca tymczasowy folder roboczy, w której program zapisuje kolejne wyrenderowane klatki animacji.
4. **void cleanWorkdir()** - Funkcja kasująca tymczasowy folder roboczy, w której program zapisuje kolejne wyrenderowane klatki animacji.
5. **void drawCircles()** - Funkcja wprowadzająca zmiany w wyglądzie trawnika wywoływana przy każdej kolejnej klatce animacji. Korzysta z obiektu `Drawer` do wprowadzania zmian oraz obiektu `SprinklerList` zawierającego listę rozstawionych zraszaczy.

5.17 ErrorWindowEvents oraz Error.

Klasa zawierająca obiekt obsługujący nasłuchiwanie akcji na interfejsie graficznym okna wyskakującego błędu `ErrorWindow`. Zawiera również obiekt `Error` zawierający metodę do uruchamiania takiego okna i zaleca się uruchamianie tego okna tylko za pomocą tej metody.

1. **void popErrorWindow(String error_message, int mode, boolean wait)** - Statyczna funkcja w obiekcie `Error` wyświetlająca na ekran okno z powiadomieniem o błędzie. Przyjmuje jako parametry treść wiadomości o błędzie, informację w jakim trybie wyskakuje to okno (`JFrame.DISPOSE_ON_CLOSE` albo `JFrame.EXIT_ON_CLOSE`) oraz boolowską wartość, czy program ma czekać aż okno zostanie zamknięte przez użytkownika.

5.18 ErrorWindow.

Obiekt przechowujący okno błędu wyskakujące za wywołaniem metody `popErrorWindow` z obiektu `Error`, informująca o przyczynie błędu i decydująca o dalszej pracy programu. Zawiera szereg funkcji renderujących poszczególne panele okna.

5.19 Drawer.

Klasa odpowiedzialna za wprowadzanie zmian na tablicy dwuwymiarowej trawnika poprzez "rysowanie" kół zraszaczy, podwyższając wartości obejmowanych komórek o 1. Potrzebuje tablicy dwuwymiarowej trawnika oraz obiektu `Sprinkler`, na podstawie którego wprowadzi zmiany w tablicy. Zalecane jest zapoznanie się z interfejsem `Sprinklers` i numerycznymi odpowiednikami typów i kierunków zraszaczy.

1. **boolean contains_quarter(int quarter, SprinklerType type, Direction direction)** - Funkcja sprawdzająca, czy przekazana ćwiartka koła znajduje się w obrębie aktualnie rysowanego zraszacza.
2. **void fix_value_artefacts(int[][] tab, Sprinkler sprinkler, int r)** - Funkcja przyjmująca tablicę dwuwymiarową trawnika, zraszacz oraz promień zraszacza, poprawiająca artefakty takie jak komórki, które przypadkiem powiększyły się kilka razy przy rysowaniu lub zostały pominięte.
3. **void draw_circle_with_reflections(int[][] tab, Sprinkler sprinkler)** - Funkcja rysująca na przyjętej tablicy dwuwymiarowej trawnika zraszacz przyjęty jako drugi parametr z uwzględnieniem odbić wody od przeszkód (znacznie pogorsza wydajność programu).

4. **void draw_circle_without_reflections(int[] tab, Sprinkler sprinkler)** - Funkcja rysująca na przyjętej tablicy dwuwymiarowej trawnika zraszacz przyjęty jako drugi parametr z pominięciem odbić wody od przeszkód.

5.20 Algorithm.

Obiekt odpowiedzialny za rozstawienie zraszaczy na trawniku. Potrzebuje obiektu interfejsu graficznego okna, na którym wyświetla się animacja, obiektu Lawn trawnika oraz listy zraszaczy w obiekcie SprinklersList.

1. **void setSprinklers()** - Funkcja główna algorytmu rozstawiająca zraszacze.