Obliczanie grafu widoczności
Dokumentacja techniczna
Instrukcja użytkownika

Dominik Adamczyk
Szymon Nowak-Trzos
grupa nr 2
Algorytmy geometryczne 2022/2023

Spis treści

1 Opis klas i funkcji programu

W tym dokumencie zostanie położony nacisk na funkcje bezpośrednio związane z implementacją algorytmu. Elementy kodu odpowiadające za działanie interfejsu graficznego nie będą szczegółowo opisywane. Funkcje i klasy odpowiadające za działanie algorytmu zaimplementowane są w plikach Essential_Functions.js, Scene.js, VisibilityGraphs.js oraz folderze avl. Pozostałe pliki odpowiadają za import biblioteki p5.js przy pomocy której został napisany interfejs graficzny, a także za obsługę logiki tego interfejsu - zarówno wizualizacji algorytmu jak i funkcjonowania przycisków. Poszczególne funkcje odpowiadające za działanie algorytmu:

- **Point** - klasa reprezentująca punkt - posiada współrzędne pojedynczego punktu
- **Line** - klasa reprezentująca odcinek - posiada dwa obiekty klasy Point
- **PointsCollection** - klasa przechowująca listę obiektów klasy Point
- **LinesCollection** - klasa przechowująca listę obiektów klasy Line
- **Shape** - klasa reprezentująca wielokąt, przechowuje obiekt PointsCollection i LinesCollection. Kolejne punkty z PointsCollection wyznaczają wielokąt.
- **Scene** - klasa przechowująca wszystkie widoczne na ekranie obiekty z klas wymienionych powyżej.
- **det** - funkcja wyznacznika przyjmująca trzy obiekty klasy point i zwracająca wyznacznik 2x2 tych punktów. Na podstawie znaku wyznacznika wyznacza się położenie ostatniego z punktów względem prostej przechodzącej przez pierwsze dwa. Funkcja ta przyjmuje też opcjonalny argument "zeros", domyślnie ustawiony na 0. W takim przypadku, dla współliniowości, zamiast zer zwracana jest różnica odległości pomiędzy pierwszym a drugim i pierwszym a trzecim punktem. Jeśli zaś argument ten będzie równy 1, w przypadku współliniowości będą zwracane zera.
- **quickSort_angle** - zaimplementowane sortowanie szybkie wykorzystujące wyznacznik do sortowania grupy punktów na podstawie kąta jaki tworzą z dowolnym wektorem i wskazanym punktem środkowym.
- **distance** - funkcja przyjmująca dwa punkty i zwracająca kwadrat odległości między nimi.
- **isTheSameLine** - funkcja sprawdzająca, czy dwa odcinki są takie same - przydatna w przypadku, gdy podczas działania algorytmu zamiatania miotła trafi na punkt będący sąsiadem punktu, z którego wychodzi.
- **hasAtLeastOneSamePoint** - funkcja sprawdzająca, czy dwa odcinki mają ten sam punkt początkowy lub końcowy. Sprawdzanie tego jest potrzebne do wyznaczania przecięć dwóch odcinków.
- **sortByAngle** - funkcja sortująca punkty wejściowe na podstawie ich kąta. Funkcja używa wcześniej zaimplementowany algorytm sortowania szybkiego, ale eliminuje jego wady opisane w sprawozdaniu.
- **LineIntersectionThatOutputsPoint** - funkcja wyznaczająca przecięcie dwóch prostych, przy pomocy rozwiązania odpowiedniego układu równań. Zwraca punkt przecięcia.
- **createHalfLine** - funkcja tworząca półprostą, będącą przedłużeniem zadanego odcinka. Potrzebna do stworzenia miotły na podstawie odcinka pomiędzy dwoma punktami. W praktyce nie jest tworzona półprosta, a odcinek, którego krańce znacznie wykraczają poza obszar, na którym zadane są figury.
- **isPointOnLine** - funkcja sprawdzająca czy wskazany punkt jest punktem początkowym, lub końcowym danego odcinka. Fukcja używana w wyznaczaniu przecięcia dwóch odcinków
- **LineIntersection** - funkcja zwracająca informację, czy dwa odcinki się przecinają. Funkcja umożliwia rozważenie różnych przypadków przecinania się dwóch odcinków (na przykład takich uwzględniających punkty przecięcie odcinków w ich punktach początkowych / końcowych, lub interpretację jednego z odcinków jako półprostej). Podejście do przecięć odcinków różni się w zależności od wykonywanego algorytmu.
- **comparatorT** - funkcja porównująca używana do umieszczania elementów na drzewie.
- **orientationCheck** - funkcja wykonująca pierwsze kroki algorytmu Jarvisa i przy tej pomocy wyznaczająca orientację w jakiej został zadany wielokąt.

- **intersectsInterior** - funkcja sprawdzająca czy dany odcinek przechodzi przez wnętrze wielokąta w otoczeniu zadanego punktu będącego wierzchołkiem tego wielokąta. Funkcja działa w oparciu o metodę opisaną w sprawozdaniu.
- **visible** - funkcja implementująca algorytm **czyWidoczny**
- **visibleVertices** - funkcja implementująca algorytm **widoczneWierzchołki**
- **visibilityGraph** - funkcja implementująca algorytm **grafWidoczności**
- **visibilityGraphNaive** - funkcja wyznaczająca graf widoczności naiwną metodą - wykorzystywana do porównania czasowego.

2 Użytkowanie aplikacji

2.1 Uruchamianie aplikacji

Najprostszą metodą na uruchomienie aplikacji jest wejście w link: `https://editor.p5js.org/Szyntos/sketches/2SYD7g12Z` i kliknięcie przycisku startu znajdującego się w lewym górnym rogu. Jest to strona na której znajduje się umieszczony na serwerze kod źródłowy aplikacji, dzięki czemu możliwe jest jej uruchomienie. Pod przyciskiem "start"znajduje się strzałka, która umożliwia rozwinięcie listy plików źródłowych aplikacji i ich dowolną edycję. Alternatywną metodą uruchomienia aplikacji - bez łączenia się za pośrednictwem internetu - jest stworzenie lokalnego serwera za pośrednictwem którego zostanie otwarty plik index.html. Gdy na potrzeby napisania kodu źródłowego aplikacji potrzebowaliśmy pokrzystać z lokalnego serwera, korzystaliśmy z aplikacji Visual Studio Code i wtyczki "Live Server". W obydwu przypadkach po uruchomieniu aplikacji może być konieczne pomniejszenie okna przeglądarki przy użyciu kombinacji "ctrl -", gdyż aplikacja nie posiada funkcji auto-skalowania.

2.2 Dodawanie wielokąta

Dodawanie wielokąta realizowane jest poprzez przycisk "Dodaj Figurę". Po kliknięciu tego przycisku można wskazać na ekranie punkty stanowiące tworzące wielokąt. Aby zakończyć dodawanie wielokąta należy ponownie kliknąć przycisk "Dodaj Figurę". Podczas dodawania wielokąta należy zwrócić uwagę na jego poprawność. Program nie wykrywa błędnie zadanych figur - do takich należą“wielokąty”, których krawędzi się przecinają, bądź figury gdzie dwukrotnie wskazano ten sam punkt jako jej wierzchołek. Nie jest też sprawdzane, czy jakiekolwiek dwie figury posiadają część wspólną - tego też należy unikać. Jakiekolwiek niepoprawne dane wejściowe mogą skutkować błędnym obliczaniem grafu widoczności. Jeżeli użytkownik chce od nowa zadać wielokąty należy użyć przycisku "Wyczyść Scenę".

2.3 Obliczanie grafu widoczności

Do wyświetlenia grafu widoczności służy przycisk "Oblicz Graf Widoczności". Dla zadanego zestawu danych można wyświetlić graf widoczności klikając w ten przycisk (czasami konieczne jest podwójne kliknięcie). Program wyświetli informację o liczbie znalezionych krawędzi, a następnie pokaże wynik działania algorytmu. Aby zakończyć wyświetlanie grafu widoczności należy ponownie kliknąć przycisk "Oblicz Graf Widoczności". Tak obliczony graf można zapisać do pliku tekstowego przy pomocy przycisku "Zapisz Graf". Wyjściowy plik tekstowy zawiera dwie listy, które w połączeniu reprezentują graf widoczności. Pierwsza lista to lista punktów będących wierzchołkami zadanych wielokątów. Druga lista reprezentuje wynikowy graf widoczności poprzez wskazanie połączeń między punktami z listy pierwszej - wskazania na elementy z pierwszej listy wyznaczone są poprzez indeksy punktów w tej liście.

Możliwe jest również obliczenie grafu z jednoczesnym porównaniem czasu wykonania algorytmu, względem algorytmu naiwnego. Służy do tego przycisk "Porównanie Czasu". Po obliczeniu grafów widoczności dwoma metodami zostanie wyświetlona informacja o długości działania programu, a także zostanie pokazany wynikowy graf widoczności. Aby wyłączyć widok tego grafu należy ponownie kliknąć przycisk "Porównanie Czasu". Dla danych wejściowych z dużą liczbą wierzchołków obliczanie grafu widoczności może trwać pewien czas - wtedy nie należy używać aplikacji aż nie zostanie wyświetlony graf.

2.4 Wyniki algorytmu widoczneWierzchołki

Program umożliwia zobaczenie wyniku działania algorytmu **widoczneWierzchołki** dla dowolnego wierzchołka należącego do danych wejściowych. Jest to domyślna rzecz, którą wyświetla program. Punkt dla którego wyświetlany jest rezultat algorytm zaznaczony jest na zielono. Aby wybrać punkt, dla którego chce się zobaczyć rezultat algorytmu należy użyć przycisku "Następny Punkt". Można również przeprowadzić algorytm dla dowolnego punktu wskazywanego przez myszkę (dla punktów wewnątrz wielokątów mogą występować błędy). W tym celu należy użyć przycisku "Punkt z Myszki", a jeżeli użytkownik chce powrócić do poprzedniego stanu powinien ponownie użyć tego przycisku.

2.5 Animacje

Użytkownik może prześledzić krok po kroku animację głównej pętli algorytmu, która z każdym krokiem dodaje do grafu kolejne krawędzie znalezione poprzez użycie algorytmu **widoczneWierzchołki**. Aby to zrobić należy użyć przycisku "Animacja Grafu", a następnie "Następny Krok", żeby wyświetlać kolejne kroki animacji. Do wyjścia z animacji konieczne jest ponowne naciśnięcie przycisku "Animacja Grafu"

Istotną funkcjonalnością programu jest możliwość wyświetlenia animacji poszczególnych kroków algorytmu zamiatania. Aby to zrobić należy przy pomocy przycisku "Następny punkt"wskazać wierzchołek dla którego będzie wyświetlana animacja. Następnie kliknąć przycisk "Animacja widoczneW."Po animacji można się dowolnie poruszać przy pomocy przycisków "Następny Krok"i "Poprzedni Krok". Animację można zresetować do pierwszego kroku przy pomocy przycisku "Reset Animacji". W trakcie działania animacji możliwa jest zmiana punktu dla którego jest ona wykonywana - do tego należy użyć przycisku "Następny Punkt". Aby wyłączyć animację należy ją zresetować i ponownie kliknąć przycisk "Animacja widoczneW.".

2.6 Zapisywanie i wczytywanie danych

Stworzone przy pomocy narzędzia dane można zapisać do pliku txt (przy pomocy przycisku "Zapisz do txt"). Format z jakim zostaną zapisane, to kolejne współrzędne punktów tworzące dany wielokąt oddzielone od siebie przecinkami. Na końcu takiej listy punktów należących do jednego wielokąta znajduje się średnik. Używając tego samego formatu możliwe jest wczytywanie danych, które zostały ręcznie zapisane do pliku txt, bądź powstały przez zapisanie poprzednio narysowanych wielokątów. Służy do tego przycisk "Wybierz plik"(napis ten może ulec zmianie w zależności od języka systemu operacyjnego), znajdujący się po prawej stronie przycisku "Zapisz do txt". Nie jest możliwe dwukrotne wczytanie tego samego pliku pod rząd. Program umożliwia również zapisywanie i wczytywanie plików do formatu .json. Pliki uzyskane tą metodą są jednak mniej czytelne od tych zapisanych do formatu .txt.

Pewnym ograniczeniem programu jest obszar roboczy, na którym prowadzona jest wizualizacja. Stanowi on kwadrat o wymiarach 1000×1000. Powoduje to, że wprowadzane przy pomocy myszki punkty mają współrzędne z przedziału $[30, 1030]^2$. Nie wpływa to jednak na działanie algorytmu, który jest prawidłowy dla punktów z $[-10000, 10000]^2$. to ograniczenie wynika z implementacji funkcji **createHalfLine**, która tworzy półprostą, jako odcinek zawarty w tym przedziale. Możliwe jest zatem wprowadzanie wielokątów o współrzędnych większych niż rozmiar sceny i obliczanie odpowiadającego im grafu widoczności, ale tylko za pomocą plików tekstowych (należy wczytać graf z pliku, oraz obliczony graf zapisać do pliku). Dla takich zestawów danych wizualizacja nie będzie prawidłowo funkcjonowała.