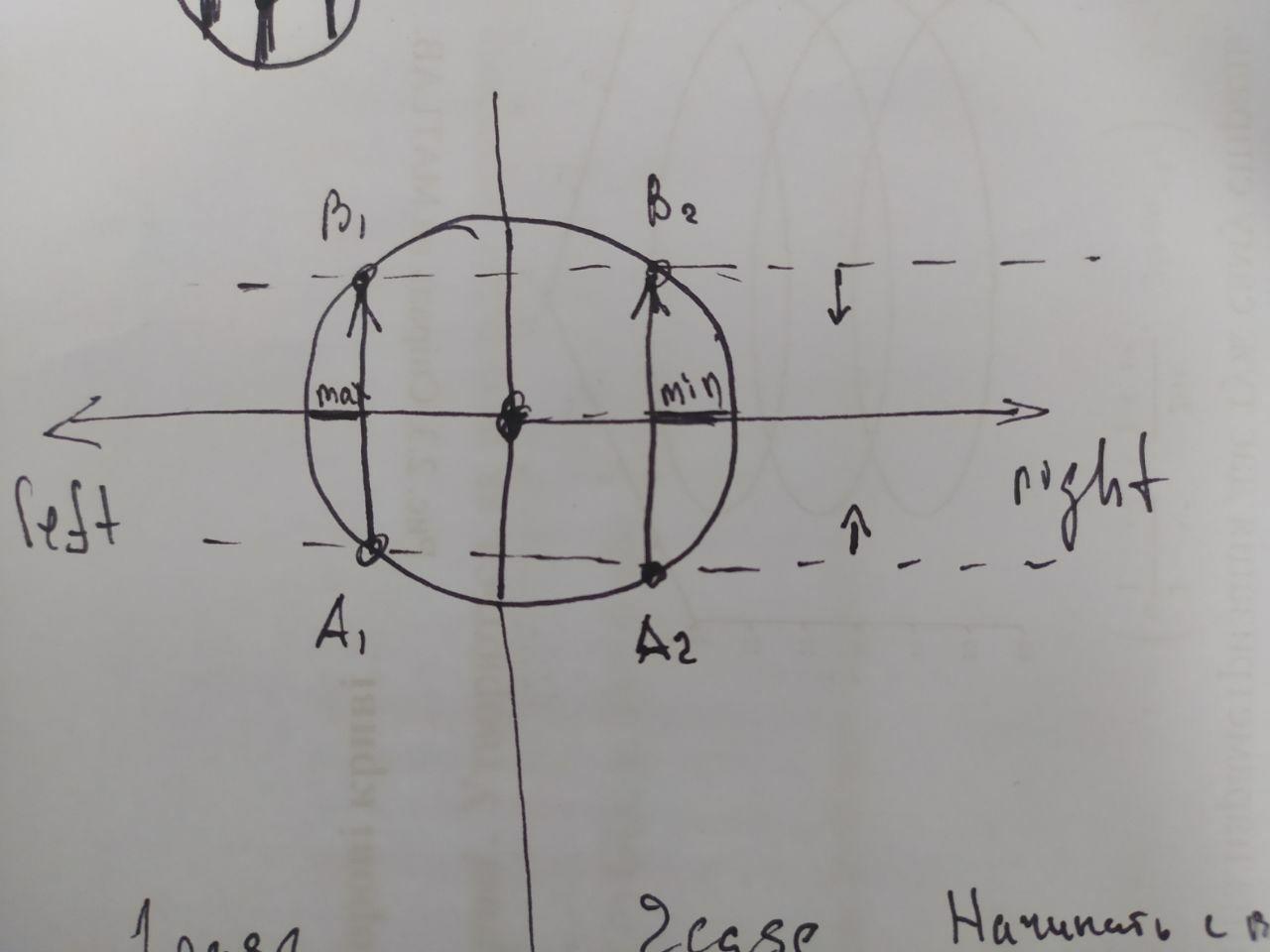
Звіт з лабораторної роботи №8  
на тему «Тріангуляція Делоне»  
з дисципліни «Комп’ютерна графіка»  
студента 3-го курсу Факультету комп’ютерних наук та кібернетики   
групи ІПС-32  
Поліщук Єгора Даниловича

**Постановка задачі.**  
В просторі E2 задана множина точок S, потужності N. Необхідно з’єднати точки множини відрізками, що не перетинаються таким чином, щоб кожна область всередині опуклої оболонки цієї множини точок була трикутником, причому накладається обмеження, що всередині описаного навколо будь-якого з цих трикутників кола не має бути точок цієї множини, але допускається їх знаходження на самому колі.  
Тріангуляція з такими обмеженнями називається тріангуляцією Делоне.

**Розв’язання.**Вважатимемо ребра напрямленими, у позначенні ребра AB будемо розуміти, що А – початок, В – кінець ребра, а тому можна визначити сторони таких ребер – ліву та праву.  
На початку алгоритму необхідно мати ребро опуклої оболонки множини S. Його можна знайти за лінійний час, вибравши за початок деяку ектримальну точку (з максимальною або мінімальною координатою) A, та виконавши крок алгоритму Джарвіса для цієї вершини та інших точок множини. Нехай отримали точку B.  
  
Отже, маємо ребро AB. Причому направимо ребро AB (можна змінити найменування знайдених двох точок) таким чином, щоб усі точки множини S знаходилися справа від AB.  
  
Тепер необхідно ввести класифікацію ребер, отримуваних у холі виконання алгоритму.  
Ребра можуть бути 3 видів:  
- сплячі (ребро, яке ще не було знайдено алгоритмом тріангуляції);  
- живі (ребро, яке було знайдене, але відома лиш одна суміжна з ним область);  
- мертві (ребро, для якого відомі дві суміжні області).  
Ребро може переходити з одного в стану в інший тільки так: спляче ->живе -> мертве.  
Перше знайдене ребро алгоритму (у нашому випадку AB) є живим, адже відомо, що з одного боку воно межує з нескінченною областю (без точок множини S).  
Класифікація ребер важлива для означення поняття границі. Границя на кожному кроці алгоритму складається з набору живих ребер.  
Саме для ребер з множини границі й будемо шукати точки, які утворять трикутник з ребром. Після знаходження найкращої точки для утворення трикутника, поточке ребро стає мертвим.   
  
Отже, у ході виконання алгоритму необхідно підтримувати множину живих ребер. Та оброблювати кожне з них. Після того, як ребро стає мертвим, воно вилучається з множини.  
  
Нехай маємо структуру даних для зберігання поточного списку живих ребер LiveEdges, тоді

LiveEdges <- AB  
DeadEdges = empty();  
while (LiveEdges is not empty){  
 cur\_edge <- LiveEdges;  
 pointC <- findPoint(cur\_edge);  
 LiveEdges <- new Edge (cur\_edge.begining, pointC);  
 LiveEdges <- new Edge (pointC, cur\_edge.end);  
 DeadEdges <- LiveEdges.remove(cur\_edge);  
}

Функція findPoint(edge) знаходить для напрямленого ребра edge точку справа від нього, найкращу у розумінні того, що трикутник з вхідного ребра та точки буде утворений так що в описаному навколо нього колі не буде знаходитись жодної точки множини S, причому ця властивість буде збережена і після завершення роботи алгоритму.



З малюнку видно, які частини треба шукати мінімальними або максимальними, відповідно до опису нижче.   
findPoint(edge){  
 foreach point in S{  
 if (point is right from edge){  
 center <- обрахувати центр описаного кола навколо трикутника  
 (edge.beginning, edge.end, point);  
Якщо (center is right from edge) обраховуємо відстань від середини edge вліво до кола та зберігаємо точку зі знайденою максимальною відстанню pointMax.  
Якщо (center is left from edge) обраховуємо відстань від середини edge вправо до кола та зберігаємо точку зі знайденою мынымальною відстанню pointMin.  
  
 }  
 }  
Якщо знайдена принаймні одна точка center для якої знаходиться зліва від edge (тобто pointMin != null), повернути pointMin, інакше повернути pointMax.  
}  
  
Таким чином буде побудована множина метвих ребер DeadEdges, яка і складає тріангуляцію. Також треба зауважити, що необхідно розглядати крайні випадки, коли  
у множині від початку відсутні не колінеарні точки (тобто точки які не лежать на одній прямій). Тріангуляція для випадку, коли всі точки розташовані на одній прямій не визначена, і не буде побудована.  
  
**Складність**Для кожного ребра (кількісно O(N)) алгоритм переглядаючи усі інші точки заданої множини шукає найкращу точку, яка б утворила трикутник з даним ребром, тому у складність виконання алгоритму O(N2).