Звіт з лабораторної роботи №6  
на тему «Метод типу розподіляй та володарюй (для мінімуму точок – Джарвіса) пошуку опуклої оболонки»  
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, яка містить N точок. Необхідно побудувати їх опуклу оболонку (повний опис границі).  
  
**Розв’язання.**Припустимо на площині задана множина S із N точок. Нехай S = S1 + S2 (так що S1 та S2 не перетинаються), S1 та S2 отримані довільним поділом множини S.  
Нехай було побудовано оболонки CH(S1) та CH(S2). Тоді CH(S1+S2) = CH(CH(S2) ˅CH(S2)), тобто CH(S) = CH(CH(S2) ˅CH(S2)).  
Отже, фактично задача полягає у розробці процедури злиття двох оболонок (з припущенням, що деякий алгоритм для знаходження оболонки вже побудований, до того ж цей алгоритм має повертати упорядковану за деяким критерієм множину точок).  
 ***Алгоритм divideAndRule(S)***:  
1. Якщо |S| <= k (k – невелике число, обране довільно або ж з деяких міркувань ефективності), то побудувати оболонку одним з прямих методів (brute force) та повернути її як упорядковану за полярним кутом відносно деякої внутрішньої точки множину точок.  
Інакше кажучи, необхідно знати «за» чи «проти» годинникової стрілки упорядковані точки опуклого многокутника, який визначає опуклу оболонку.  
В нашому випадку застосуємо алгоритм Джарвіса, що повертає множину точок упорядковану проти годинникової стрілки.  
Якщо |S| > k, то перейти до кроку 2.

2. Розбити множину S на дві підмножини приблизно однакової потужності так, що S= S1+S2.   
3. Злити опуклі оболонки множин S1 та S2 та поверну їх.   
Тобто return (mergeConvexHull(divideAndRule(S1), divideAndRule(S2)).

**Заваження.**Напрям упорядкованості точок треба вибрати (за чи проти годинникової стрілки) заздалегідь перед реалізацією алгоритму.

Необхідно розглянути окремо як мінімум три алгоритми: Jarvis(S), merge(S1, S2), grehem().  
***Алгоритм Jarvis(S):***1. Знайти найнижчу точку down = (x0,y0) = miny(S).  
2. Знайти точку серед усіх інших крім down, таку що вектор з початком в down та кінцем у ній утворює найменший кут з вектором з початком в down = (x0,y0) та кінцем в фіктивній точці (x∞***,***y0). Тобто знайдемо точку p1, вектор з кынцем у якій та початком в down утворює найменший кут в полярній системі координат з початком в down.  
Якщо кілька «мінімальних» точок, то обрати найвіддаленішу, а інші з таких позначити як використані.   
3. Циклічно розглядаючи кожну з залишених та не використаних точок починаючи з p1 знаходити для поточної точки piточку pij, так що кут між вектором з кінцем у поточній точці та початком у попередньо розглядуваній та вектором з початком у поточній і кінцем у pij утворювали найменший кут. Якщо кілька «мінімальних» точок – обрати найвіддаленішу, інші з них позначити як використані.

***Алгоритм*** ***злиття Шеймоса*** ***merge(P1, P2):***Обмеження: P1 та P2 – опуклі многокутники, задані упорядкованими множинами їх точок.  
1. Знайти будь-яку внутрішню точку p многокутника P1. p – центроїд будь-яких трьох точок P1.   
2. Визначити, чи є p внутрішньою для P2 (лежить по один бік від ребер опуклого многокутника P2). Якщо так – перейти до кроку 3, інакше – до кроку 4.  
3. Оскільки P1 та P2 упорядковані за одним критерієм, то злити вершини P1 та P2 у єдиний упорядкований список (лінійно). Перейти до кроку 5.  
4. Вибрати довільну точку p20 многокутника P2, знайти найлівішу та найправішу точки pl та pr многокутникаP2 відносно вектора pp20.  
Вилучити зі списків P1 та P2 точки:  
- з P1 точки, що лежать одночасно правіше вектора ppl та лівіше вектора ppr;  
- з P2 точки, що лежать лівіше вектора prpl.   
Злити вершини списків P1 та P2 у єдиний упорядкований за кутом відносно точки p список.  
5. Застосувати алгоритм Грехема до отриманого списку.

Алгоритм Грехема являє собою по суті вироджений випадок алгоритму Джарвіса, де замість перебору точок для кожної вершини можна рухатись уздовж оболочки лінійно та обирати точки виходячи з умови: якщо кут між попереднім ребром-вектором та поточною точкою більший за наступну точку у порядку проходу (упорядкованості списку за кутом), то обрати, як наступну точку що входить до оболонки ту саму «наступну» точку, а поточку виключити, проробити таку послідовність дій для кожної точки.  
Таким чином за лінійний час можна виключити з упорядкованого списку точки, що не належать опуклій оболонці.

**Особливості реалізації**  
Мова реалізації: Java.