Convex Hull Report

Matthew Tibbitts, Beckett Maestas, Jacob Ly

Professor Jonathan Schrader

CSC212

University of Rhode Island

Introduction

This Project aims to investigate the convex hull. Therefore, this project will comprehend what a convex hull is and how it works. The project's main goal was to demonstrate the Graham Scan algorithm's operation and its relationship to the Convex hull. The smallest convex set contained by a form is known as a convex hull. When a group of lines linking two points is replaced by a single line, the result is a convex set. Numerous techniques are used to create the convex hull, but at their core each one compares two sets of points, eliminates extended connections between them, and replaces them with convex sets. We can make the shorter distance by substituting these extended connections, and when we combine them, we can produce the smallest form for a group of points.

Method

The Graham Scan algorithm is the algorithm that we will be using for this project. The initial step of this procedure is to locate the lowest Y position. After that, we will calculate the angle between the lowest point that we just found and the other point in the set. Finding the correct angle between the lowest and a point in the set via an arc tangent or inverse tangent will make results in a more accurate, therefore, we will use arc tangent to determine the polar angle and after that, arrange the points according to the polar angles that were generated. The lowest and first point in the sequence will be put into a stack once everything has been arranged. We will use the stack to contain all the convex hull which will also be used to compare those point in the set. Identifying if the subsequent line moves relative to the ongoing line counterclockwise will help us establish whether a point is on the convex hull or not. We decide whether the point is clockwise or counterclockwise using the cross product formula of the desired point. If the cross product result in negative, it is counterclockwise motion or turn left, else if the cross is zero, the points are colinear, and if the cross product is positive, then the point is clockwise motion or turn right. Therefore, if the point is counterclockwise or turn left, we will push it onto the stack that we mentioned earlier. Then, we proceed to checking onto the next ordered point by using all point that are all related to the point on the stack. If a point or the subsequent value rotates clockwise with respect to the line, we repeatedly remove the top point from the stack until the subsequent value rotates counterclockwise. Then we proceed to the next ordered point that is in relation to the new counterclockwise point. We will repeat this process until, we finish checking all the set point that we've been given, then finally, it will result in a convex hull is formed. We use SFML for visualization in our project therefore, SFML installation is a must. Once the Convex hull is formed, it will then send to SFML for visualization.

Implementation

The beginning of implementation for this project, we will need to get SFML downloaded in order to get it to work. First, we open the file that would be input consist of set points which would be used to generate convex hull. In the next step, we intialize a vector that will be is filled with points from a text file once the data structures have been initialized. Then the class HullGenreator is called in the main function which navigate us the function that would generate the output display onto the SFML application. This function take the vector of set point that we got from the text file as a parameter, then we have an while loop contains a statement that determine the window should be close or not. After that we have statement that compare the size of the vector of the node point to the count and graham scan function is called which will used to generate convex hull and display it onto SFML application.

When it called the graham scan function within the Hullgenerator.cpp, it initialized a variable called “anchorindex” that will be set to a value that we obtain from calling the determinelowest function. The determinelowest function is an O(n) function which will return the index at which points reside in the vector has the lowest Y coordinate. Once, we found the point with lowest Y index or “arnchorindex”, we proceed to creating a new vector that will hold the sorted polar angle value by calling a function, “sortByPolarAngle”, which would take the anchorindex we found earlier as a parameter and as the desire point to calculate the polar angle. In the sortByPolarAngle function, it will calculate the polar angle, sorts them and colors the points for the display on the grid. After we obtain the sorted vector of point based on angles, we then create a stack called “hull”, and push the zero, first, second element of the sorted point by angle onto the stack. A for loop is called and the subsequent element is added to the stack after establishing a for loop. After that, the cross product is calculated using a while loop inside of the for loop. The while loop will repeatedly run to check the computed cross product (using the isLeftTurn function to calculate the cross product and determine the motion of the point). This the while loop acts as though the convex hull's curves are being eliminated because it will pop out or remove the point that are turning right or counterclockwise.. By removing values that conflict with the vector's next ordered point, this creates the shortest distance to the next point. Following the while loop, the for loop will go through each value in the sorted list of points. Then once it goes through everything, the convex hull is obtained and will be colored for the display on SFML.

Contribution

After getting our project topic, everyone make an effort to study about the Graham's Scan algorithm and how it works through multiple online resources such as YouTube's videos, google search, and online blogs or articles. We then discussed as a group on Discord (instant messaging social platform) about what we are going to do to make convex hull using Graham's Scan algorithm interesting. Beckett Suggested that we should implement a bagplotter that basically a visualization tool which is formed by a convex hull of a given percentage of points in a data set and then another convex hull around the outliers in the data set.

GitHub Post:

Matthew creates the Github post and initialize the README file. Jacob and Beckett later join to help edit the file which include the introduction, psuedocode, the tutorial to use the code, and so on.

Code Implementation:

Beckett creates the first version of the program which include the main.cpp, Hullgenerator.cpp, and Hullgenerator.h. Then Matthew and Jacob, help debug for the error as well as figure out how to implement the write and read file function in order to read the file and output the dot file.

Graham Scan's And Other Algorithm:

Beckett mainly implement the Graham's Scan and most of other algorithms including the SFML implementation for the display output, and it has little to no error. Therefore, after checking to make sure that there is no error in the code and adding comments, Jacob and Matthew, proceed to do other tasks for the project. Matthew implemented the read and write file function as well as the node.h and node.cpp.

Final Report:

Jacob initalizes the final reports of the project while the other are testing to make sure the program are working properly. Then the other proofread, edit, and add information accordingly to the report and agree for the final submission on gradescope.