Convex Grabbing Game Implementation

1 Overview

My final project is an implementation of a convex hull game discussed by Matsumoto et al. 2020[1]. I used Graham's Scan algorithm for convex hull creation, and I programmed the game using JavaScript, HTML, and CSS, which makes it easily accessible via a web interface. The interface supports convex hull creation both automatically and manually, and allows the user to plot and rearrange points on a grid to adjust the convex hull in real time. Additionally, the user can scroll over each point to view its associated "weight," used for the convex hull game. The main feature of the website is the game itself, which requires two players and is described in detail in section 2. In this write-up, I will first summarize the game and discuss the main points of the work of Matsumoto et al., then describe my implementation in depth. The game can be found at the link below:

`https://www.coltonwolk.com/convexhull/index.html`

2 The Game

2.a Instructions

The game is nicknamed the "convex grabbing game," based off of the "graph grabbing game"[2]. The game has the following rules:

- Two players take turns removing a point off of the convex hull.
- Each point has an associated weight, or score, which is added to the player's total upon being removed from the game board.

[1]Matsumoto, N., Nakamigawa, T. & Sakuma, T. Convex Grabbing Game of the Point Set on the Plane. Graphs and Combinatorics 36, 51–62 (2020). https://doi.org/10.1007/s00373-019-02117-z

[2]Winkler, P.M.: Mathematical puzzles: a connoisseur's collection. A K Peters, Natick, MA (2003)

- The goal is to maximize the total score, and whichever player has a higher score (more than half of the total weight) once all weights have been removed from the board wins.
- Points may only be removed from the convex hull, and the hull is reformed after each move. Thus, neither player can greedily select maximum weight points to win in all cases.

It has been shown that the first player will always win under a specific set of circumstances, but a guarantee for an arbitrarily structured board with arbitrary point scores is not yet known. As mentioned, the game is based on the "graph grabbing game" where two players collect non-cut vertices instead of convex hull points, and similarly try to maximize their scores.

2.b Board Layouts

There are a few basic layouts of the graph which make it easy to determine which player will win. If, for example, all points are on the convex hull, then it is clear that the first player can repeatedly take the point with the maximum score and then win the game. It is also clear that, if a point on the hull has a score greater than half of total weight on the board, then the first player will win. Although a general case is not known, there are other particular settings which guarantee a win for one of the two players, Alice and Bob (who take turns in that order), which Matsumoto et al. states as the paper's main theorem:

Theorem 1. *Let P be an odd-point set on the plane in general position. If P has at most two inner points, then Alice wins the game on P.*

This theorem is proved by induction on the number of points. I did not use this theorem in my implementation, but it is useful to know as a strategy while playing the game.

3 My Work

3.a Graham's Scan Implementation

In total my program has 4 documents: an HTML file, a CSS file, and two JavaScript files. The JavaScript files contain an implementation of Graham's

Scan as well as most of the demo/game code. I based the website style and structure loosely off of a previous website I created, but heavily customized it for this project. I've included my pseudocode of Graham's Scan convex hull algorithm below:

Algorithm 1: Graham's Scan

```
Data: A set S containing n points
Result: Points on the Convex Hull
if n ≤ 3 then
    return points;
end
c ← centroid of three arbitrary points;
l ← leftmost point of S;
j ← // first Jarvis March edge;
for each point p ∈ S do
    find slope of p with respect to centroid;
    find slope of p with respect to l;
    j ← max slope of lp so far;
end
sort S with respect to slope with centroid;
stack T.push(l, j);
p ← j
while p ≠ l do
    p ← next point in sorted order;
    while p and top two points in T don't form a left-hand turn do
        T.pop;
    end
    T.push(p);
end
return T;
```

My implementation style follows closely to the lecture notes from class[3], although the actual implementation is much longer and requires some intermediate steps. Most of my time in this part was spent testing and debugging the program.

[3] http://www.cs.tufts.edu/comp/163/notes05/CH1_handout.pdf

4 Learning Process and Future Work

My project has gone through many stages. After deciding that I wanted to program a game, I initially thought about implementing Fortune's algorithm combined with a bubble-shooter game style, such that any shot hitting the wavefront would set back the sweep line or delete the corresponding "growing" parabola, but this proved to require heavy modifications to Fortune's algorithm and thus became far too complex. I then looked at research papers online for inspiration and found a paper relating to the graph/gold grabbing game[4], then came across the convex hull application. Overall, this has been an educational experience not only in programming a convex hull algorithm, but also in learning about combinatorial and graph-related games and applicable mathematical principles behind them. For future possibilities and extensions of this project, I've come up with a few additional ideas:

- Visualize the steps of Graham's scan, or program additional convex hull algorithms and illustrate the steps in creating the hull.
- Create specific point and weight arrangements to allow for selection of particular game boards showcased in the paper.
- Program a "CPU" player to automatically calculate the optimal moves based on the paper, and have it compete against a real person.
- Use machine learning to train the "CPU" on both specific and random arrangements of points and weights.
- Further investigate the open problems posed by Matsumoto et al.
- Generalize this game to 3D convex hulls, Voronoi diagrams, Delaunay triangulations, or other geometric data structures and representations.

[4]Seacrest, D.E., Seacrest, T.: Grabbing the gold. Discrete Math. 312, 1804–1806 (2012)