Convex Hull Using Graham's Scan

Ethan Fox and Afamdi Achufusi

Introduction

Our goal for this project was to successfully implement the Graham's Scan method for computing the convex hull of a set of points. Our code needed to handle degeneracies in the input data, such as collinear points. Additionally, we created two new initializers for point configurations as test cases, and we incorporated test cases from other classmates.

1. Handling Degeneracies

For this project, we initially had coded a more simple version of Graham's Scan that could not handle collinear points. We realized quickly that there were some degenerate inputs for this algorithm including:

- **Collinear Points:** Just multiple points lying on the same line.
- **Duplicate Points:** Identical points in the dataset.
- **Points with the Same Polar Angle:** Points that form the same angle with the reference point.

Handling Collinear Points

Collinear points can cause issues during the sorting phase and while constructing the convex hull. To handle them:

1. **Sorting Phase:** When sorting points by their polar angle with respect to the starter point (lowest point), if multiple points have the same angle, we sort them by their distance from the starter point.
2. **Removing Duplicates:** After sorting, we remove points that have the same angle and are further from the starter point. This is done to avoid considering unnecessary collinear points that do not contribute to the convex hull.
3. **Stack Maintenance:** During the hull construction, if we see three consecutive points that are collinear, we ensure that the middle point is removed from the hull.

2. Custom Initializers

We created two new initializers to make test cases with tricky shapes to test on:

2.1 Ellipse

The ellipse initializer generates points along the perimeter of an ellipse centered in the window.

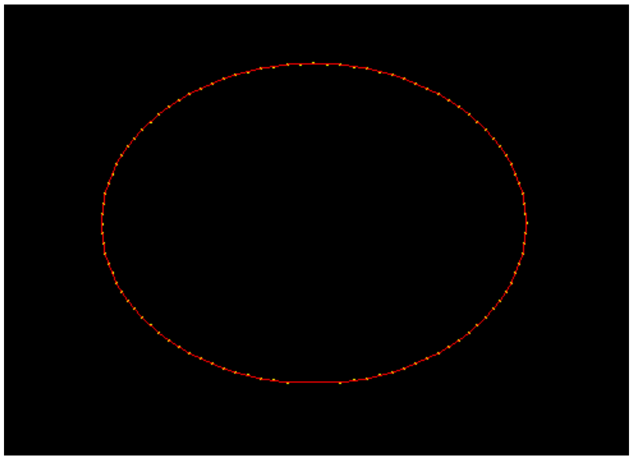

Figure 1: Points in the shape of an Ellipse

2.2 Crescent

The crescent initializer creates a crescent-shaped set of points by combining two arcs of different radii.

3. Other Initializers Used

In addition to the custom initializers, we also used plenty of predefined initializers, some from the project's original code and some from classmates on Slack.

4. Convex Hulls Computed

Using a value of $n = 100$, we computed the convex hulls for each point configuration that we could test on.

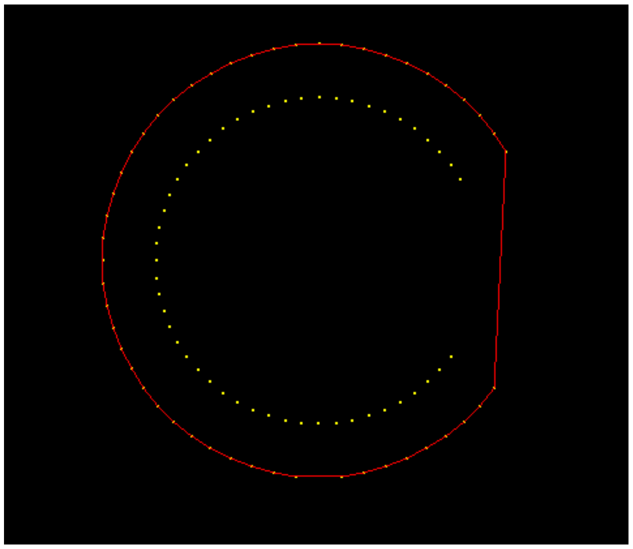

Figure 2: Points forming a Crescent

5. Algorithm Performance and Issues

For all of the cases we were able to test or code on, it performed as expected and computed the right convex hull. It is definitely possible that there could be some other degenerate input we did not discover that could give our code problems, but we did not find any.

No significant issues were encountered that prevented the code from working in any case that we were able to find at least.

7. Time Spent

- **Thinking:** Overall we thought about our implementation of this algorithm throughout the project and we definitely did not get it right the first time. We spent plenty of time thinking about how to solve degenerate cases once we discovered them.
- **Programming:** The bulk of the time spent was on programming the Graham's Scan algorithm to work on all of the cases we had and thinking of ways to handle edge cases. Lots of time was spent on this.
- **Testing:** We tested throughout because we had access to plenty of test cases from Slack, so we did not end up spending a ton of time on that.
- **Documenting:** Creating this report on Overleaf always takes longer than anticipated but it is not too long. We commented the code as we went so that was not a lot of time on documentation either.

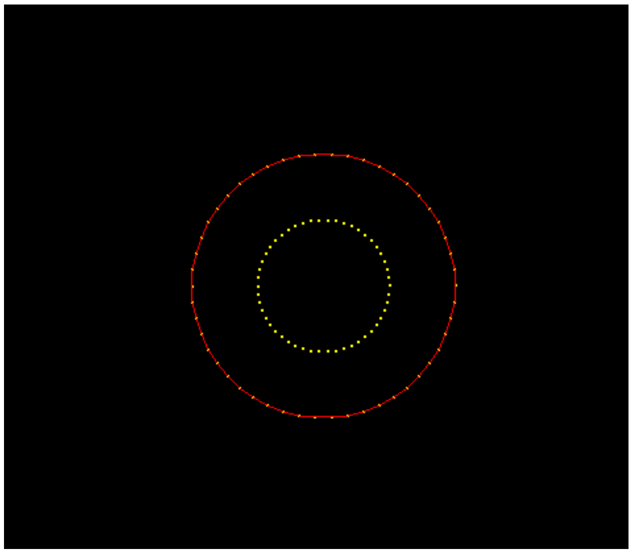

Figure 3: Points forming a Circle

8. Reflection

This project was a good programming experience that definitely helped build confidence with implementing spatial algorithms. We found it challenging at times, usually when trying to fix bugs that were causing issues with handling degeneracies such as collinear points. One thing we wish we did differently was create a more robust algorithm from the get-go for Graham's Scan that attempted to account for collinear points. Thinking harder before starting our initial algorithm probably could have saved us some time later. Something else we learned was that visualization is very helpful when you are having issues. Being able to see the points in space helped when a bug with the convex hull was happening.

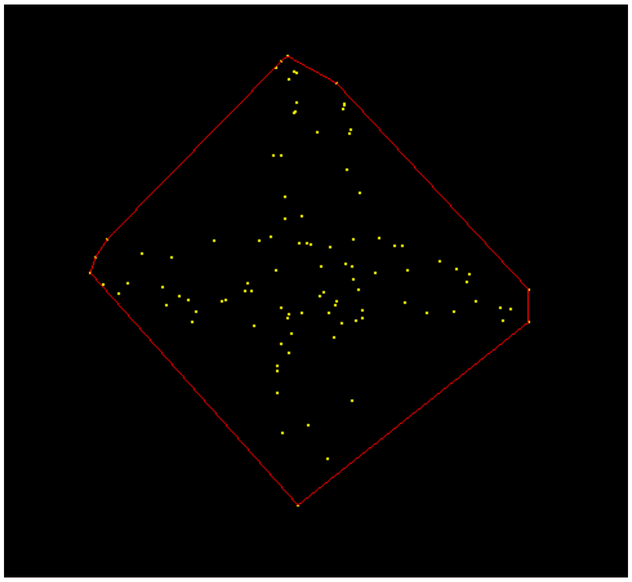

Figure 4: Points forming a Cross

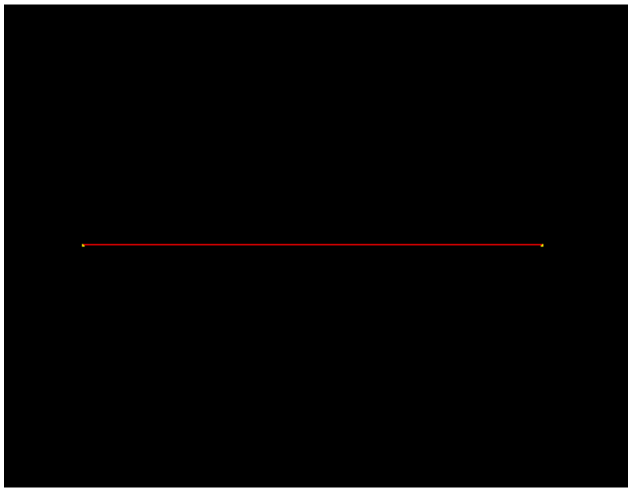

Figure 5: Points forming a Horizontal Line

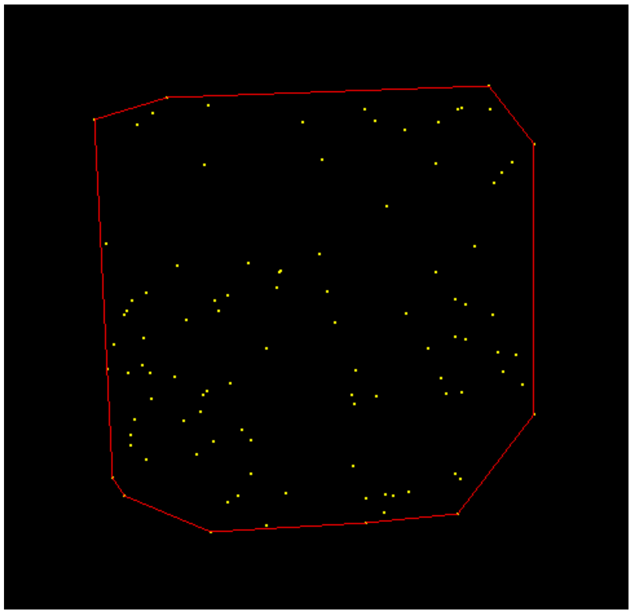

Figure 6: Randomly Distributed Points

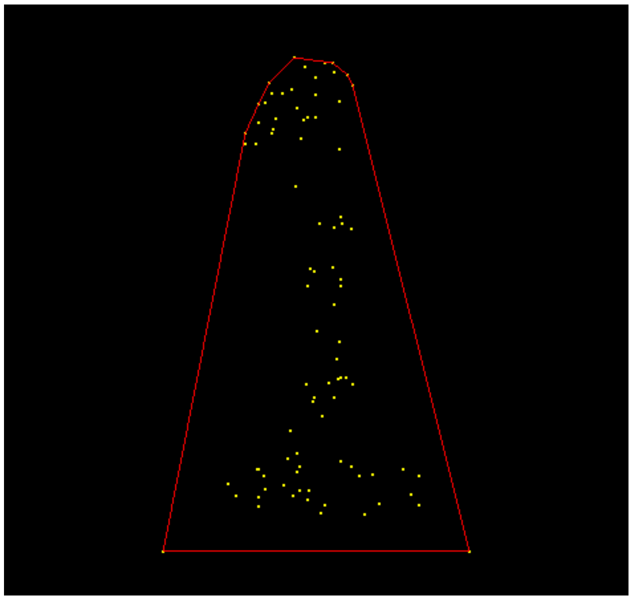

Figure 7: Points forming a 1

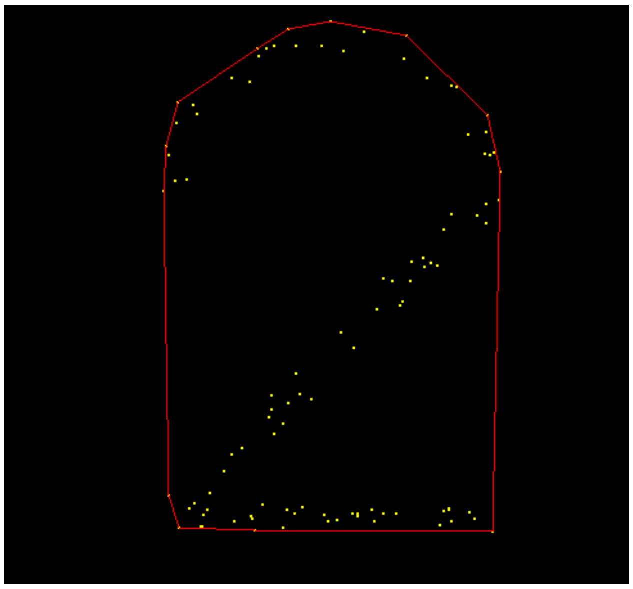

Figure 8: Points forming a 2

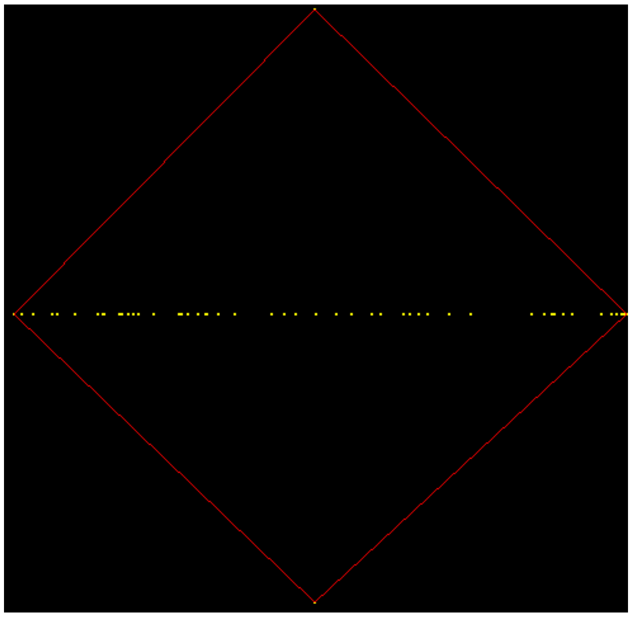

Figure 9: Points forming a thin cross

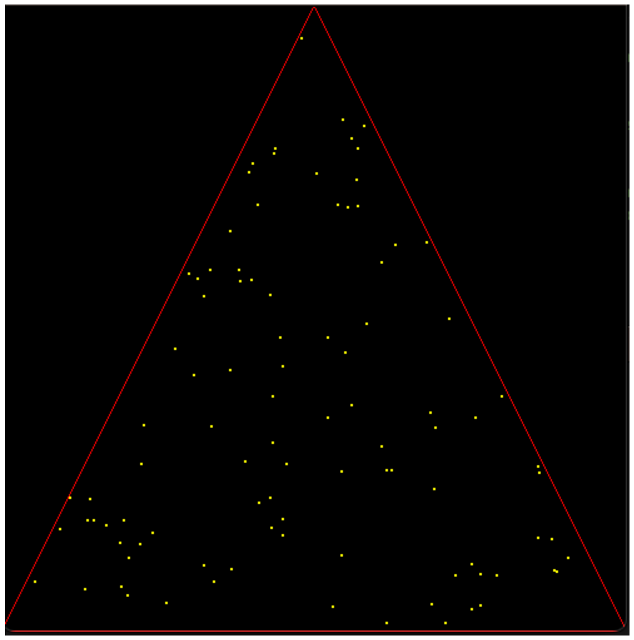
Figure 10: Points forming a triangle

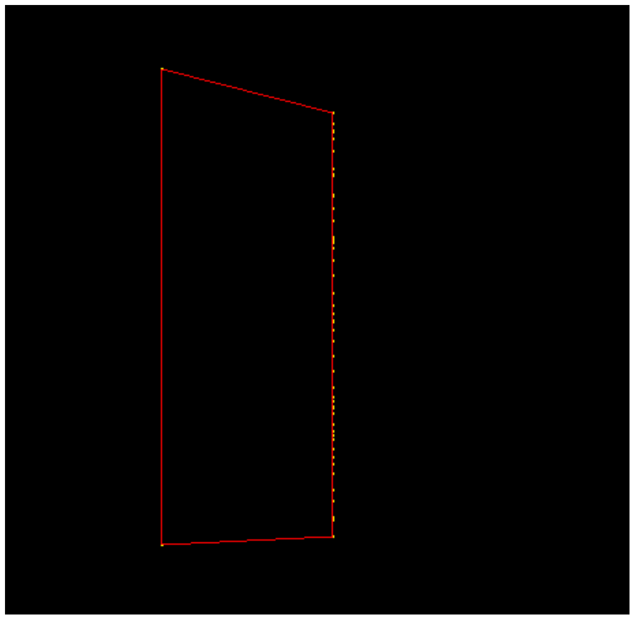
Figure 11: Points forming two vertical lines

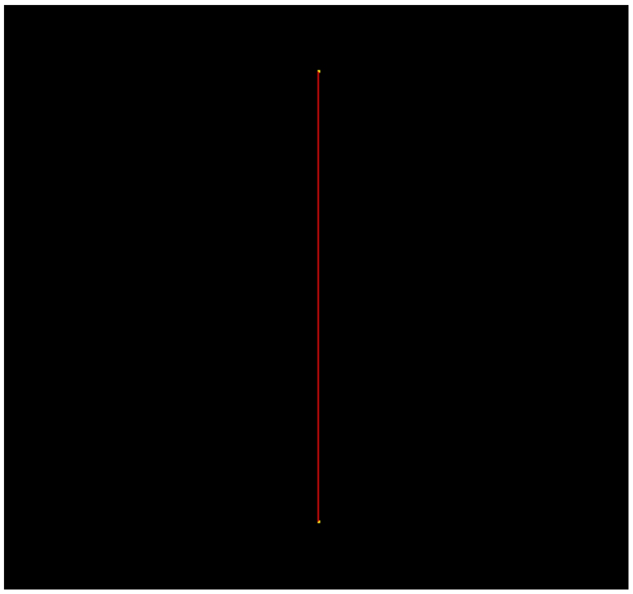

Figure 12: Points forming a vertical line