THE DETERMINATION OF ORIENTATION FOR SIMPLE POLYGONS FOR USAGE WITHIN GRAPHICAL DESIGN

ANTHONY NGUYEN

1. Introduction

Explain the purpose of our algorithm and why it's important/relevant.

Before we can go in-depth into our algorithm, we must define a polygon, specifically a simple polygon.

- A polygon encloses a region, which has a measurable area.
- The edges of a polygon only meet at their respective vertices and do not intersect anywhere else.
- Exactly two edges meet at a given vertex.
- The number of edges always equal the number of vertices.

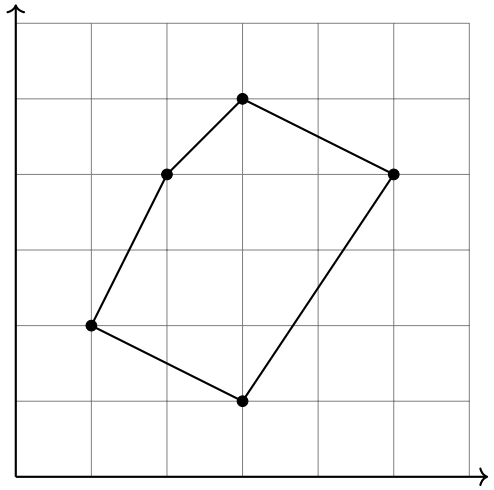


Figure 1. Example of Simple Polygon

For the polygon above, these vertices are provided without any form of ordering. When we do assign ordering to these points, we see two diverging possibilities: Clockwise (CW) and Counterclockwise (CCW).

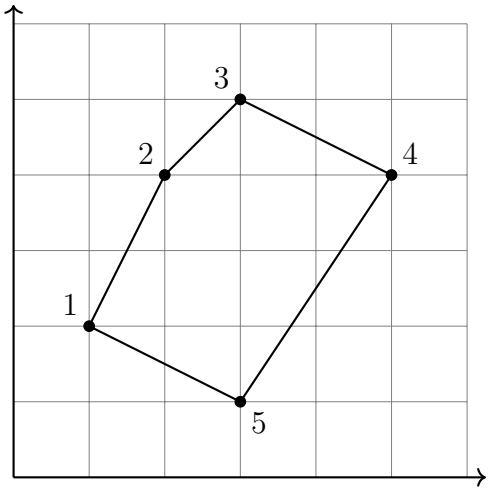


FIGURE 2. Simple Polygon with CW Orientation

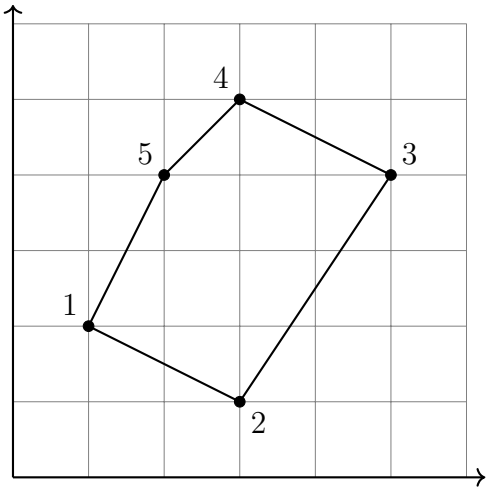


FIGURE 3. Simple Polygon with CCW Orientation

At first glance, we can easily identify the orientation of an ordered polygon if the polygon is arranged properly. However, how would we automate this process, while still maintaining speed and portability?

2. Algorithm Analysis

By utilizing the bottom-most vertex and the cross product of its adjacent extending vectors, we can determine the orientation of a given simple polygon.

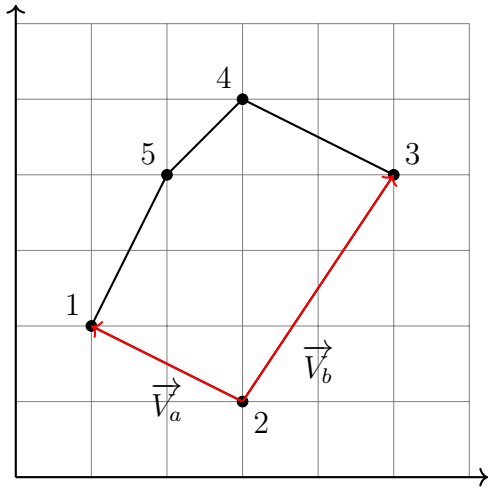


Figure 4. CCW Polygon, with Emphasized Extending Vectors

2.1. **Algorithm Breakdown.** The algorithm is as follows:

- Determine the vertex/vertices at the global minimum.
  - In the case of multiple vertices, select the vertex with the lowest x-value
- Using the selected vertex, determine the vectors going toward the adjacent vertices ($\overrightarrow{V_a}$ to the previous vertex, $\overrightarrow{V_b}$ to the next vertex)
- Determine the sign of $\overrightarrow{V_a} x \overrightarrow{V_b}$
  - [Case A] If positive, the polygon is clockwise.
  - [Case B] If negative, the polygon is counter-clockwise.
  - [Case C] If zero, the orientation is indeterminable (to be discussed later).

Case A:

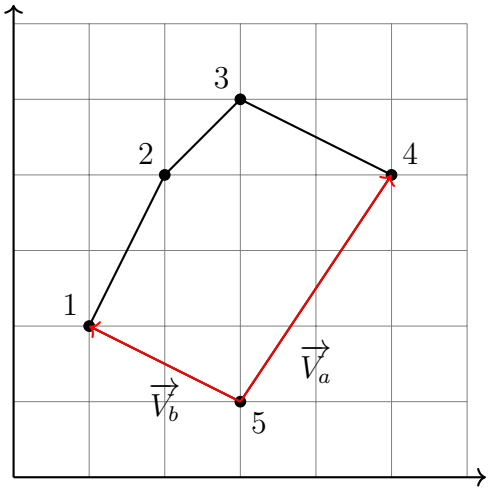


FIGURE 5. Simple Polygon with CW Orientation

$$\begin{aligned}\overrightarrow{V_a} \times \overrightarrow{V_b} &= \begin{vmatrix} X & Y & Z \\ 2 & 3 & 0 \\ -1 & 1 & 0 \end{vmatrix} \\ &= (0,\ 0,\ 5) \\ &\to 5\ \widehat{z}\ \text{[CLOCKWISE]}\end{aligned}$$

Case B:

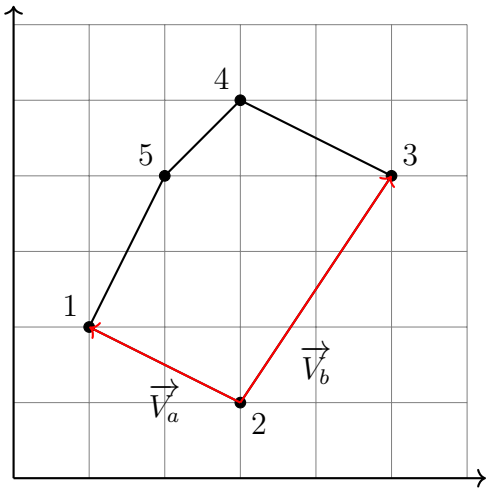
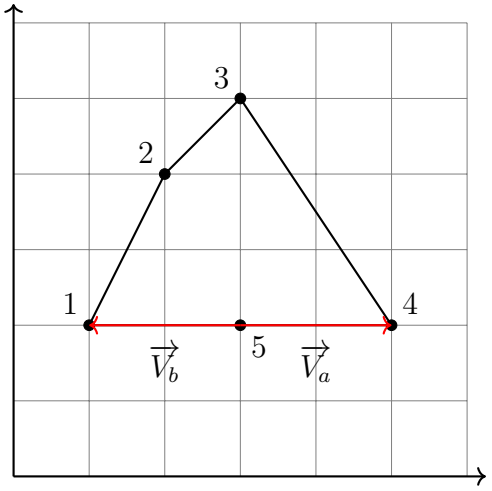


FIGURE 6. Simple Polygon with CCW Orientation

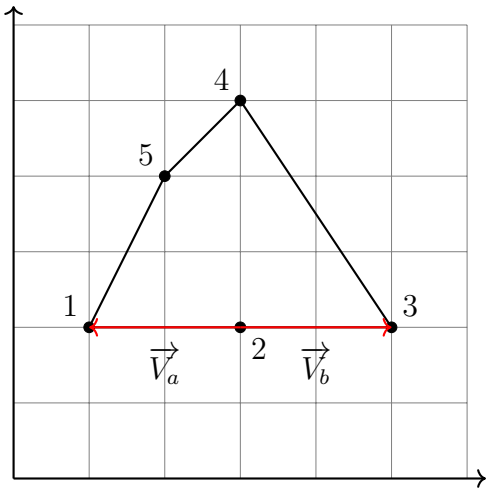
$$\begin{aligned}\overrightarrow{V_a} \times \overrightarrow{V_b} &= \begin{vmatrix} X & Y & Z \\ -1 & 1 & 0 \\ 2 & 3 & 0 \end{vmatrix} \\ &= (0,\ 0,\ -5) \\ &\to -5\ \widehat{z}\ \text{[COUNTER-CLOCKWISE]}\end{aligned}$$

Case C:



$$\begin{aligned}\overrightarrow{V_a} \times \overrightarrow{V_b} &= \begin{vmatrix} X & Y & Z \\ 2 & 0 & 0 \\ -1 & 0 & 0 \end{vmatrix} \\ &= (0,\ 0,\ 0) \\ &\to 0\ \hat{z}\ [\text{UNKNOWN}]\end{aligned}$$

FIGURE 7. Simple Polygon with CW Orientation



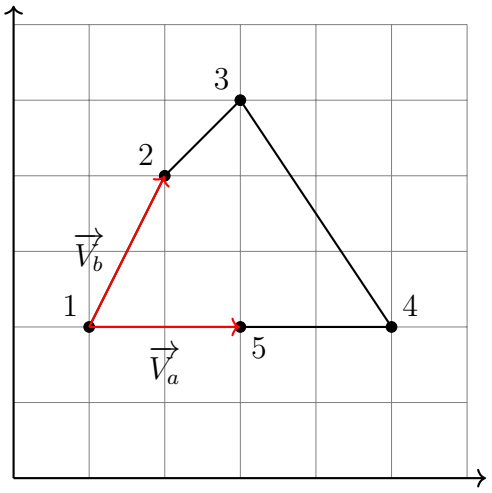
$$\begin{aligned}\overrightarrow{V_a} \times \overrightarrow{V_b} &= \begin{vmatrix} X & Y & Z \\ -1 & 0 & 0 \\ 2 & 0 & 0 \end{vmatrix} \\ &= (0,\ 0,\ 0) \\ &\to 0\ \hat{z}\ [\text{UNKNOWN}]\end{aligned}$$

FIGURE 8. Simple Polygon with CCW Orientation

Because both polygons (with differing orientations) give us Z values with varying signs, we are unable to algorithmically identify these polygons' orientation in its current state.

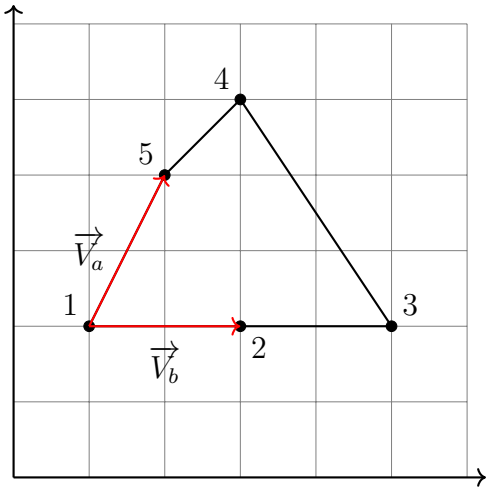
In order to prevent this edge case from causing errors, we decide to select a vertex (within our previous subset of vertices) with the lowest x-value. By doing this, we ensure that one of its adjacent vertices will have a differing y-value and will not cause an error of collinearity.

Case C (MODIFIED):



$$\overrightarrow{V_a} \times \overrightarrow{V_b} = \begin{vmatrix} X & Y & Z \\ 2 & 0 & 0 \\ 1 & 2 & 0 \end{vmatrix}$$
$$= (0,\ 0,\ 4)$$
$$\rightarrow 4\,\hat{z}\ \text{[CLOCKWISE]}$$

FIGURE 9. Simple Polygon with CW Orientation



$$\overrightarrow{V_a} \times \overrightarrow{V_b} = \begin{vmatrix} X & Y & Z \\ 1 & 2 & 0 \\ 2 & 0 & 0 \end{vmatrix}$$
$$= (0,\ 0,\ -4)$$
$$\rightarrow -4\,\hat{z}\ \text{[COUNTER-CLOCKWISE]}$$

FIGURE 10. Simple Polygon with CCW Orientation

3. Mathematical Analysis

In order to determine the orientation of the entire simple polygon, we sample the vertex with the lowest y-value. From the sampled vertex, we calculate the sign of the cross-product of its preceding vector and its succeeding vector.

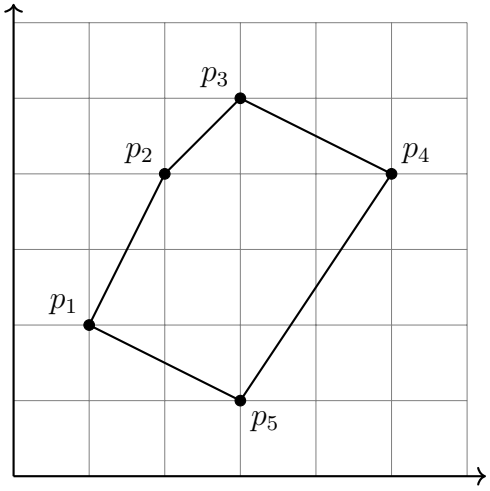


Figure 11. Simple Polygon with CW Orientation

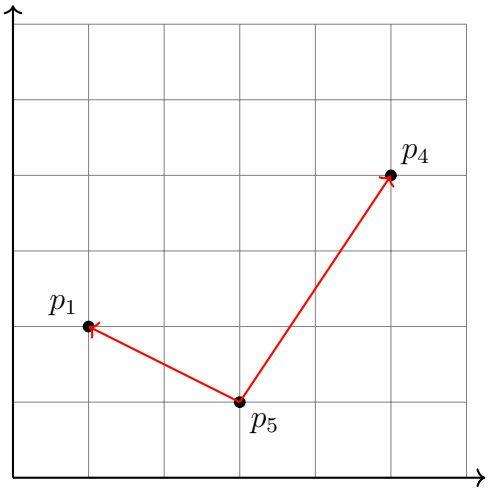


Figure 12. Sampled Vertex with its Extending Vectors

In regards to vertex p_5, all other vertices either must have an equal or higher y-value because p_5 is the bottom-most vertex by definition. As a result, we know that all other vertices will have an equal or greater y-value than our chosen vertex. As a result, the polygon will be in the same orientation as the cycle entering p_5, because this particular cycle cannot "loop around" the chosen vertex to change the orientation.

4. Comparison to Pre-existing Algorithms

Shoelace

They're both O(n). This algorithm takes less operations (only one cross-product necessary, compared to the n cross-products required for an n-sided polygon).

5. Testing

https://www.youtube.com/watch?v=3N3Bl5AA5QU

applications with imgur

u2net vertex extraction making clockwise/counterclockwise using algorithm

applications with tkinter

bentley-ottman determining orientation in real-time

applications through random polygon generation

Princeton algorithm chaikin for smootthing "Smoothness" factor

random walk monte-carlo simulation