ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

Kaloumenos Stavros

RESEARCH WORK OF THE COURSE

«COMPUTER GRAPHICS»

Coordinator and teacher

Vasilios Dracopoulos

Lamia, May 2025

SUMMARY

The purpose of the project was to develop a program in C++ and OpenGL for the creation of arbitrary convex polygons, with the possibility of dynamic modification of the number of sides by the user in real time. The program calculates the polygons with mathematical calculations, without using predefined commands OpenGL, except for points and lines. The program generates random points and calculates the convex contour of these points using the method ”monotone-chain”. A basic calculation is the calculation of the ”cross prod- uct” of two vectors, to check the direction of rotation and ensure the correct order of points. The use of the library was initially planned GLUI, but due to inability to install AntTweakBar was chosen for management of the polygon parameters by the user.

ΠΕΡΙΛΗΨΗ

Σκοπός του έργου ήταν η ανάπτυξη ενός προγράμματος σε C++ και OpenGL για τη δημιουργία αυθαίρετων κυρτών πολυγώνων, με τη δυνατότητα δυναμικής τροποποίησης του αριθμού των πλευρών από τον χρήστη σε πραγματικό χρόνο. Το πρόγραμμα υπολογίζει τα πολύγωνα με μαθηματικούς υπολογισμούς, χωρίς χρήση προκαθορισμένων εντολών OpenGL, με εξαίρεση σημείων και γραμμών. Το πρόγραμμα δημιουργεί τυχαία σημεία και υπολογίζει το κυρτό περίγραμμα αυτών των σημείων χρησιμοποιώντας τη μέθοδο ”monotone-chain”. Ένας βασικός υπολογισμός είναι ο υπολογισμος του ”cross product” δύο διανυσμάτων, για να ελεγχθεί η κατεύθυνση της περιστροφής και να διασφαλιστεί η σωστή σειρά των σημείων. Αρχικά προγραμματίστηκε η χρήση της βιβλιοθήκης GLUI, αλλά λόγω αδυναμίας εγκατάστασης επιλέχθηκε η AntTweakBar για τη διαχείριση των παραμέτρων του πολυγώνου από το χρήστη.

TABLE OF CONTENTS

1 INTRODUCTION

The purpose of this work is to design and develop an interactive program in C++ and OpenGL, which supports the creation and filling of arbitrary convex polygons with the ability to dynamically change the number of sides in real time. A requirement of the application is that geometric calculations be performed mathematically, without the use of OpenGL's ready-made drawing functions, with the exception of use for points and lines.

The work focuses on the implementation of the convex1 procedure, as a generalization of the existing triangle1 procedure, which is limited to triangles. The goal is to be able to fill any convex polygon, following the rasterization technique by calculating the linear functions of the edges for each horizontal line of the surrounding box.

2 THEORETICAL BACKGROUND

2.1 COMPUTATIONAL GEOMETRY

A branch of computer science that studies algorithms related to geometry, it developed primarily due to the need for efficient processing of geometric data, particularly in the context of computer graphics and computer-aided design.

2.2 CONVEX HULL

A convex hull is the smallest possible polygon that encloses all the points of a given set of points in a two-dimensional (2D) space. In the case of three-dimensional (3D) space, the convex hull would take the form of a convex polyhedron.

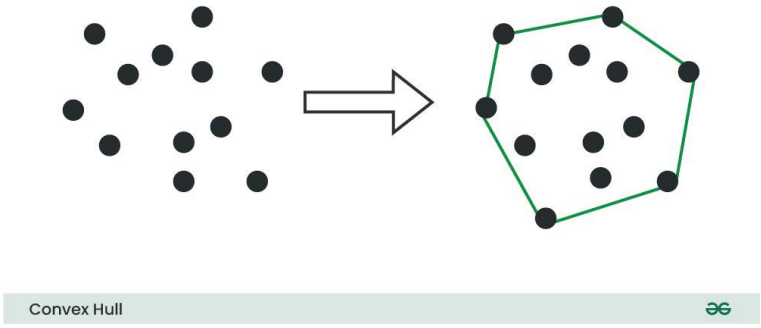

Figure 1: 2D convex polygon

2.3 GRAHAN SCAN ALGORITHM

This is a simple but efficient algorithm that calculates the convex hull of a set of points with a running time of $O(n \log n)$. The algorithm iteratively adds points to the convex hull until all points are added. It first finds the point with the smallest y coordinate, sorts the remaining points by polar angle, and adds iteratively, checking whether the last two points form a right turn and removing the last point if so.

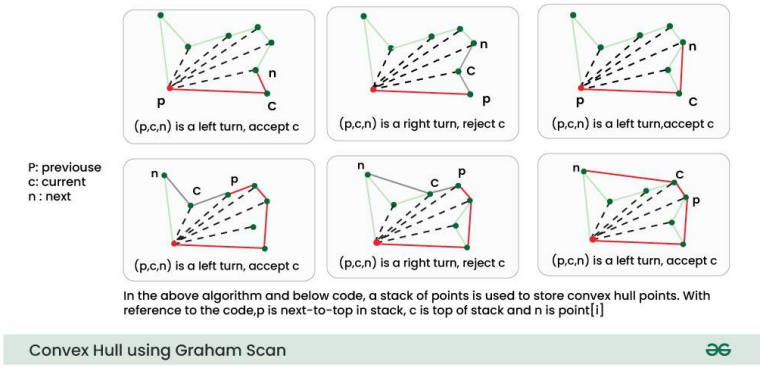

Figure 2: Accept or reject points

2.4 ANDREW'S MONOTONE CHAIN ΑΛΓΟΡΙΘΜΟΣ

A variant of the Graham Scan algorithm. Both algorithms have the same execution time, but Monotone Chain is easier to implement, as it does not require the calculation of polar angles and reduces precision errors. In essence, it is a more efficient version of Graham Scan.

2.5 CROSS PRODUCT

The vector derivative **a** × **b** of vectors **a** and **b** results in the vector **C** which is defined in three-dimensional space (3D), is perpendicular to both **a** and **b** with a direction given by the right-hand rule and a magnitude equal to the area of the parallelogram formed by the vectors.

The vector derivative $\mathbf{a} \times \mathbf{b}$ is defined as follows:

$$\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\| \sin(\theta)\, \mathbf{n}$$

where:

- θ is the angle between the vectors **a** and **b**,
- $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ are the sizes of the vectors **a** and **b**,
- **n** is the unit vector perpendicular to the plane formed by the vectors **a** and **b**.

If the vectors are parallel, the vector derivative will be the zero vector **0**, because $\sin(\theta)$ will be equal to zero.

2.6 SCANLINE ALGORITHM

A 3D graphics method that processes an image line by line to calculate which surfaces are visible in the final view. That is, it examines line by line from top to bottom, finding which pieces will be visible, making it fast and light on the current computer's memory consumption.

3 SYSTEM DESIGN

Visual Studio Code was chosen as the integrated development environment for this work due to its familiarity and ease of use. The OpenGL, FreeGlut and AntTweakBar libraries were then downloaded and placed in the appropriate MinGW (Minimalist GNU for Windows) folders as the computer's operating system was Windows 10. Initially, the GLUI library was going to be used, but due to the inability to properly install it, the AntTweakBar library was preferred, specifically for easy GUI creation and Open Source..

After research and reflection, the following list of requirements to be implemented was created:

1 : Window setup and 2D coordinate system definition.

2 : Creating a user interface (GUI).

3 : Creating a convex polygon with n sides.

4 : Polygon drawing.

4 IMPLEMENTATION

4.1 Window setup and 2D coordinate system definition

A simple problem that was solved using the classical functions of FreeGlut (glutInit, glutInitWindowPosition, ... etc).

4.2 Creating a user interface (GUI)

SetupAntTweakBar was created, which first initializes AntTweakBar to work with OpenGL. Then it is informed about the size of the window and creates a toolbar named ”settings” to which it adds a slider that controls the number of sides of the polygon (3 to 1000). Furthermore, if the user changes the size of the window, AntTweakBar adjusts accordingly via the **glutReshapeFunc**.

4.3 Creating a convex polygon with n sides.

The most difficult part of the task was ensuring the convexity of the polygon with the number of its sides being exactly equal to **n**, something that was finally achieved with the **generateConvexPolygon** function. The function initially checks whether $\mathbf{n} \geq \mathbf{3}$, ensuring that the polygon will have at least 3 points, and continues generating unique random crowd points **n**.

Then, within **generateConvexPolygon**, **computeHull** is called, which uses the **Andrew’s Monotone Chain Algorithm** to find the **convex hull** of **n** points, divides the construction into two phases: lower and upper, and checks which points create convexity, if not, the last point is removed.

In the case where the generated **convex hull** is composed of less than **n** points, then the sides of the hull are subdivided by linear interpolation and intermediate points are calculated for each side until there are exactly **n** sides. The newly added vertices are placed on the line connecting the two, so the polygon is still convex.

Finally, the **centroid** is calculated and using **atan2**, the points are sorted around it in a counterclockwise **(CCW)** direction. This ensures a consistent and correct arrangement of the points, avoiding problems such as internal corners and gaps.

4.4 Polygon drawing

The polygon is drawn using the **fillPolygon** function which uses the **ScanLine algorithm**, first the smallest and largest y values of the polygon are calculated to determine the scanline width. Then an outline table (ET) is created in order to store the following values after checking for each edge:

- **yMin**: Smallest y value of the edge
- **yMax**: Smallest x value of the edge
- **xAtYmin**: The value x when y = yMin
- **invSlope (Δx/Δy)**: The slope (to calculate change of x with respect to y)

Then, a new active edge table (AET) is constructed, containing the edges that cross the current scan line. These edges are sorted by x value, then the intersections between them with the current scan line are calculated, and inner shading is performed between their pairs, which is repeated for all scan lines.

5 Results

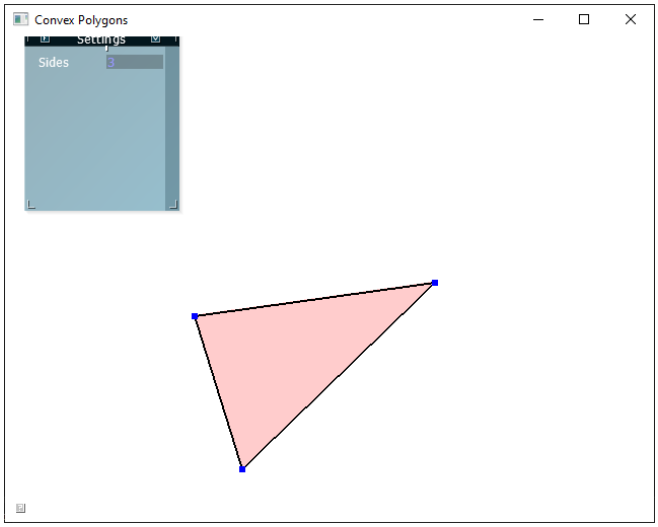

Figure 3: Convex polygon with 3 sides

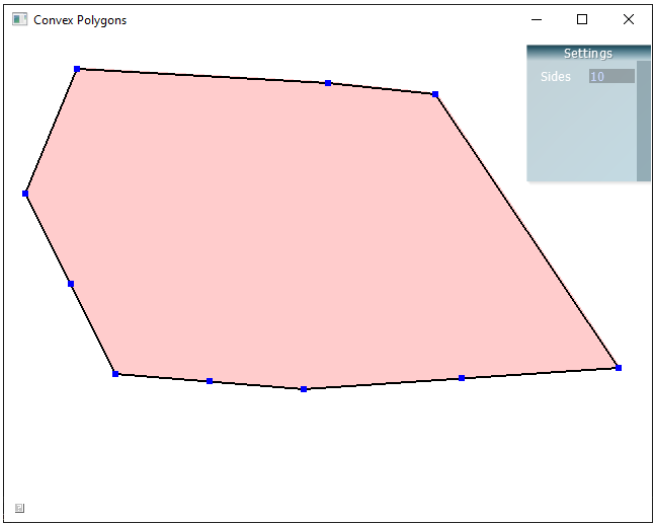

Figure 5: Convex polygon with 10 sides

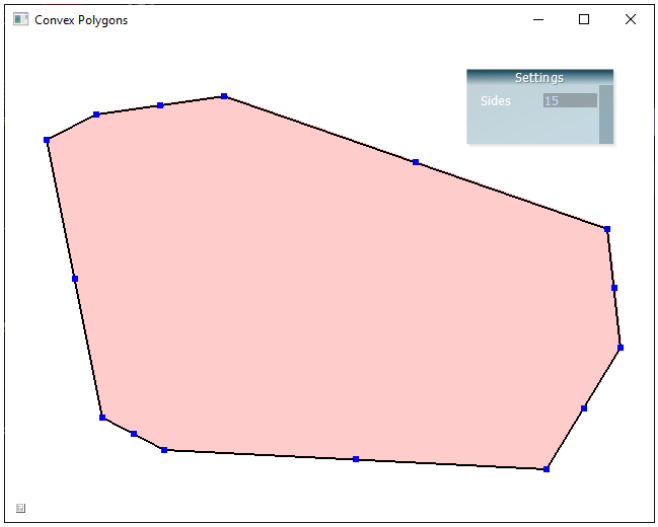

Figure 4: Convex polygon with 15 sides

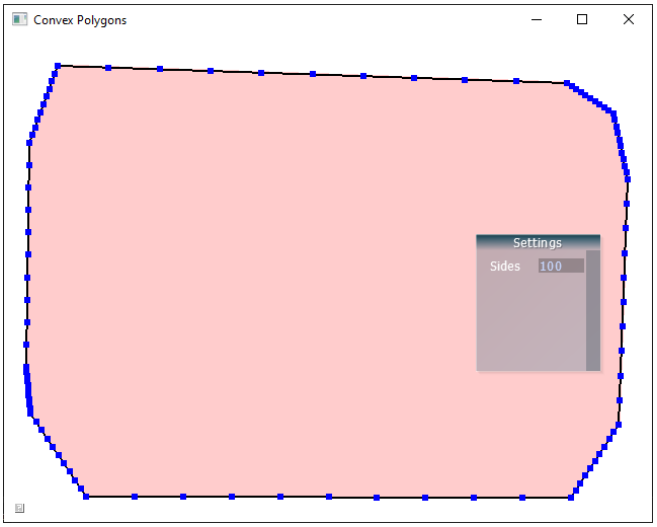

Figure 6: Convex polygon with 100 sides

6 EPILOGUE

In conclusion, it can be noted that the requested project plan for the ”convex polygons” assignment of the Computer Graphics course during the academic year 2024-2025 was implemented with complete success. It is noteworthy, however, that the efficiency of the program would be significantly improved if the computing power of the graphics card (GPU) were used with the Triangle Tessellation method, making the program faster as it would use hardware designed for graphics functions, and of course less ”heavy” since the load of pixel-level calculations would be released from the processor (CPU).

7 REFERENCES

Wikipedia

1 Computational geometry: https://el.wikipedia.org/wiki/Τ _

2 Cross Product: https://en.wikipedia.org/wiki/Cross_product

3 Scanline Algorithm: https://en.wikipedia.org/wiki/Scanline_rendering

4 Tessellation: https://en.wikipedia.org/wiki/Tessellation_(computer_graphics)

GeeksForGeeks

1 Convex Hull: https://www.geeksforgeeks.org/convex-hull-algorithm/

2 Graham Scan Algorithm: https://www.geeksforgeeks.org/convex-hull-using-graham-scan/

3 Andrew’s Monotone Chain Algorithm: https://www.geeksforgeeks.org/convex-hull-monotone-chain-algorithm/

4 Cross Product: https://en.wikipedia.org/wiki/Cross_product