ΚΥΡΤΑ ΠΟΛΥΓΩΝΑ

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ
Στα πλαίσια του μαθήματος «ΓΡΑΦΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ»

Υπεύθυνος Καθηγητής: Βασίλειος Δρακόπουλος

Στεφανία Δουλιάκα (ΑΜ 00974)

Ζώης Χουτεσιώτης (ΑΜ 00878)

Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική
Πανεπιστήμιο Θεσσαλίας

ΛΑΜΙΑ 2022-2023

Περίληψη

Στην παρούσα εργασία αναφερόμαστε στη μαθηματική θεωρία πίσω από τα κυρτά πολύγωνα, αναλύουμε μια γενικευμένη διαδικασία χάραξης αυτών από ένα δοθέντα αλγόριθμο αρχέγονου τριγώνου, και περιγράφουμε τη διαδικασία απεικόνισής τους σε υπολογιστικό περιβάλλον. Αρχικά, παρουσιάζουμε τα μαθηματικά θεμέλια που αφορούν τα κυρτά πολύγωνα, συμπεριλαμβανομένων των ιδιοτήτων τους, των γωνιών και των ακμών τους. Στη συνέχεια, εξετάζουμε τις βασικές λειτουργίες που αφορούν τα κυρτά πολύγωνα και εφαρμόζονται στη γραφική υπολογιστών, όπως ο υπολογισμός του εμβαδού και ο έλεγχος των εσωτερικών σημείων τους, καθώς και ο υπολογισμός των γραμμικών συναρτήσεων των ακμών τους για όλα τα εικονοστοιχεία του περιβάλλοντος κυτίου τους. Στο δεύτερο μέρος της εργασίας μας, επικεντρωνόμαστε στην απεικόνιση αυθαίρετων κυρτών πολυγώνων από χρήστη σε υπολογιστικό περιβάλλον χρησιμοποιώντας την OpenGL. Επιπλέον, παρουσιάζουμε παραδείγματα κώδικα και εφαρμόζουμε τους αλγορίθμους ψηφιδόξυσης και υπολογισμού του εμβαδού. Τέλος, παρουσιάζουμε τα αποτελέσματα των δοκιμών που πραγματοποιήσαμε για την αξιολόγηση της απόδοσης του προτεινόμενου συστήματος. Τα αποτελέσματα δείχνουν ότι το σύστημα παρέχει αποτελεσματική και ακριβή απεικόνιση των κυρτών πολυγώνων με χρήση της βιβλιοθήκης OpenGL.

Λέξεις - κλειδιά

Κυρτά πολύγωνα, ψηφιδόξυση, υπολογιστική γεωμετρία, υπολογιστική γραφική, OpenGL

Abstract

In this study, we refer to the the mathematical theory behind convex polygons, the general procedure for constructing them using a given primitive triangle algorithm, and describe the process of visualizing them in a computational environment with only mathematical operations. Initially, we present the mathematical foundations concerning convex polygons, including their properties, angles, and edges. We then examine the fundamental mathematical operations related to convex polygons, such as computing their area and convex hull, as well as calculating the linear functions of their edges for all pixel elements within their bounding boxes. In the second part of the study, we focus on the visualization of arbitrary convex polygons in a computational environment using OpenGL. Additionally, we provide code examples and implement algorithms for rasterization and area analysis. Finally, we present the results of the tests conducted to evaluate the performance of the proposed system. We assess the speed and accuracy of the computations in various scenarios and polygon sizes. The results demonstrate that the system provides efficient and accurate visualization of convex polygons using the OpenGL library.

Key Words

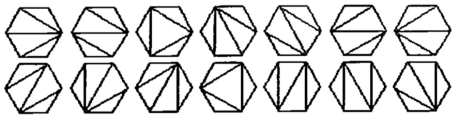
Convex polygons, rasterization, computational geometry, computer graphics, OpenGL

Περιεχόμενα

Εισαγωγή

Η υπολογιστική γεωμετρία αποτελεί κλάδο της επιστήμης των υπολογιστών που συνδυάζει τη γεωμετρία με την υπολογιστική ισχύ και τη γραφική υπολογιστών. Αποσκοπεί στην ανάπτυξη αλγορίθμων και μεθόδων για την αναπαράσταση, την ανάλυση και την επεξεργασία σχημάτων δύο και τριών διαστάσεων στον υπολογιστή. Ένας από τους τομείς που απασχολεί την υπολογιστική γεωμετρία είναι η μελέτη των πολυγώνων. Τα πολύγωνα είναι κλειστά γεωμετρικά σχήματα με ευθείες ακμές που σχηματίζουν γωνίες μεταξύ τους. Στη υπολογιστική γεωμετρία μελετούνται αλγόριθμοι για την κατασκευή, αναγνώριση και αλληλεπίδραση με πολύγωνα με ποικίλες εφαρμογές σε ευρύτερο φάσμα επιστημονικών πεδίων.

Η εργασία αυτή επιχειρεί να γενικεύσει και να μετατρέψει έναν αλγόριθμο σχεδίασης αρχέγονου τριγώνου triangle1 σε μια διαδικασία convex1 (κάθε πολύγωνο ασχέτως κοιλότητας ή κυρτότητας μπορεί να κατασκευαστεί από τρίγωνα, επομένως ττα τελευταία αναφέρονται ως αρχέγονα) (Δρακόπουλος, 2019). Η διαδικασία αυτή θα χαράσσει σε ένα υπολογιστικό περιβάλλον αλληλεπίδρασης χρήστη αυθαίρετα κυρτά πολύγωνα, υπολογίζοντας αυξητικώς τις γραμμικές συναρτήσεις των ακμών τους για όλα τα εικονοστοιχεία του περιβάλλοντος κυτίου τους. Ο χρήστης έχει τη δυνατότητα να μεταβάλλει διαδραστικώς και σε πραγματικό χρόνο το πλήθος των ακμών του κυρτού πολυγώνου. Οι υπολογισμοί σε όλη την έκταση του προγράμματος γίνονται με μαθηματικό τρόπο, δηλαδή αποφεύγεται η χρήση των προκαθορισμένων εντολών σχεδίασης της OpenGL πλην εκείνων που σχετίζονται με σημεία και γραμμές.



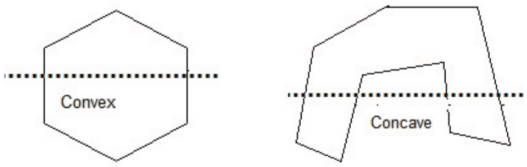
Εικόνα 1: Τριγωνοποίηση πολυγώνων - Τα τρίγωνα ως αρχέγονα πρότυπα σχεδίασης

Στόχος της εργασίας είναι η χάραξη ή αλλιώς ψηφιδόξυση (rasterization), η απεικόνιση και το γέμισμα (polygon fill) μόνο δισδιάστατων κυρτών πολυγώνων. Επομένως εξασφαλίζεται ο ορθός έλεγχος κοιλότητας με βάση το γεωμετρικό ορισμό κυρτών και κοίλων πολυγώνων. Για την επίτευξη των παραπάνω χρησιμοποιήθηκαν και δοκιμάστηκαν αλγόριθμοι χάραξης και ελέγχου εσωτερικών σημείων των

πολυγώνων, καθώς και υπολογισμού του κυρτού περιβλήματος όπως ο αλγόριθμος γεμίσματος με γραμμή σάρωσης (Scanline) και ο αλγόριθμος περιτυλίγματος (Jarvis - Wrapping).

1. Θεωρητικό Υπόβαθρο

Στη γλώσσα των μαθηματικών ένα πολύγωνο (polygon) αναφέρεται ως ένα επίπεδο σχήμα που ορίζεται από τρεις ή περισσότερες θέσεις συντεταγμένων που ονομάζονται κορυφές (vertices), και συνδέονται στη σειρά από ευθύγραμμα τμήματα που ονομάζονται ακμές (edges) ή πλευρές (sides) του πολύγωνου. Γεωμετρική απαίτηση είναι οι πλευρές των πολύγωνων να μην έχουν άλλα κοινά σημεία εκτός από τις κορυφές τους. Έτσι, ένα πολύγωνο πρέπει να έχει όλες του τις κορυφές σε ένα μόνο επίπεδο και δεν μπορούν να υπάρχουν διασταυρώσεις πλευρών (Baker, Hearn, 2010).



Εικόνα 1.1: Κυρτό και κοίλο πολύγωνο

Ως εσωτερική γωνία (interior angle) ενός πολύγωνου ορίζεται η γωνία εντός του συνόρου του, που σχηματίζεται από δύο προσκείμενες πλευρές. Αν όλες οι εσωτερικές γωνίες ενός πολυγώνου είναι μικρότερες ή ίσες με $180°$, τότε το πολύγωνο ανάφερεται ως κυρτό (convex). Εξ' ορισμού, εάν επιλέξουμε δύο οποιαδήποτε εσωτερικά σημεία ενός κυρτού πολυγώνου, το ευθύγραμμο τμήμα που ενώνει τα δύο αυτά σημεία θα βρίσκεται επίσης στο εσωτερικό του. Ενα πολύγωνο που δεν είναι κυρτό ονομάζεται κοίλο (concave), και έχει τουλάχιστον μία εσωτερική γωνία μεγαλύτερη από $180°$. Σε αυτήν την περίπτωση η προέκταση μερικών πλευρών του θα τέμνει άλλες πλευρές, και τουλάχιστον ένα ζεύγος εσωτερικών σημείων θα παράγει ένα ευθύγραμμο τμήμα που θα τέμνει το σύνορο του. Με βάση αυτά τα χαρακτηριστικά μπορεί να δημιουργηθεί ένας αλγόριθμος ελέγχου κοιλότητας.

Για να ελέγξουμε εάν ένα πολύγωνο είναι κυρτό ή κοίλο, ο έλεγχος μπορεί γενικευμένα να επιτυγχανθεί με τη δημιουργία ενός διανύσματος για κάθε πλευρά του πολυγώνου και υπολογίζοντας το εξωτερικό γινόμενο των προσκείμενων πλευρών

του. Σε μαθηματική ανάλυση τα γινόμενα αυτά των διανυσμάτων θα έχουν πρόσημο είτε θετικό είτε αρνητικό, αλλά πάντα ίδιο για ένα κυρτό πολύγωνο. Απλούστερα, εφαρμόζεται έλεγχος γραμμής, κατά τον οποίο εάν μια γραμμή περνά μέσα από το πολύγωνο θα πρέπει να τέμνει μόνο δύο πλευρές του πολυγώνου. Αντίθετα, σε ένα κοίλο πολύγωνο, η γραμμή μπορεί να τέμνει το σχήμα σε περισσότερα από δύο σημεία. Προγραμματιστικά στην εργασία μας ο έλεγχος κοιλότητας, δεδομένου ότι υπάρχει σχεδίαση διεπαφής χρήστη για εμφάνιση σχημάτων δύο διαστάσεων, έχει επιτυγχανθεί με βάση το αν ο χρήστης ορίζει τα τρία διαδοχικά σημεία του τριγώνου κατά τη φορά του ρολογιού ή αντίστροφα αυτής.

```
// If both clockwise and counterclockwise orientations are detected,
    // the polygon is concave.
    if (isClockwise && isCounterClockwise) {
        return false;
```

1.1 Χάραξη πολυγώνου

Η χάραξη πολυγώνου σε δύο διαστάσεις είναι μία διαδικασία όπου καθορίζονται τρία ή περισσότερα σημεία που αποτελούν το πολύγωνο σε ένα επίπεδο ή ένα χώρο. ’Ενα πολύγωνο αποτελείται από ευθεία τμήματα ή ακμές που συνδέουν διαδοχικά σημεία ή κορυφές. Προϋπόθεση για τη χάραξη (ψηφιδόξυση) πολυγώνων είναι να τηρηθούν βασικές μαθηματικές λειτουργίες για τον καθορισμό των γεωμετρικών ιδιοτήτων τους. Ορίζονται οι κορυφές, δηλάδη καθορίζονται συντεταγμένες - θέσεις των σημείων που αποτελούν τις άκρες και αυτές συνδέονται με ευθείες γραμμές μεταξύ τους για να σχηματιστούν ακμές. Δεδομένου ότι ένα πολύγωνο μπορεί να είναι κυρτό ή κοίλο αναλόγως της παραπάνω σχεδίασης, οφείλουμε αναλόγως τις απαιτήσεις ενός προβλήματος ή το πεδίο εφαρμογής του πολυγώνου να κάνουμε έλεγχο κυρτότητας. Τέλος υπολογίζεται η περιοχή του πολύγωνου με βάση τις συντεταγμένες των κορυφών του. Υπογραμμίζεται ότι το πιο απλό πολύγωνο (μάλιστα κυρτό) είναι το τρίγωνο, γι’ αυτό και στην εργασία μας η χάραξη των πολυγώνων γίνεται πάντα με γνώμονα τη χάραξη τριγώνων.

1.1.1 Εφαρμογές αλγορίθμων

Η χάραξη πολυγώνων είναι μια εφαρμογή της υπολογιστικής γεωμετρίας με ποικίλες εφαρμογές που ασχολείται με τον υπολογισμό των εξωτερικών ή εσωτερικών σημείων,

καθώς και με τον υπολογισμό της κυρτότητας και της μορφολογίας του πολυγώνου. Διατίθεται πληθώρα αλγορίθμων χάραξης, οι οποίοι διαφέρουν σε πολυπλοκότητα και εμφανίζουν διαφορετικά αποτελέσματα, υπερτερώντας ή υστερώντας ο καθένας αναλόγως τη χρησιμότητα και τις απαιτήσεις του συστήματος.

Μεταξύ των βασικών αλγορίθμων χάραξης πολυγώνων είναι ο αλγόριθμος Jarvis (ή αλλιώς γνωστός ως αλγόριθμος περιτυλίγματος και ο αλγόριθμος Scanline (ή αλλιώς γνωστός ως αλγόριθμος γεμίσματος με γραμμή σάρωσης). Παρότι εφαρμόσαμε και κάναμε δοκιμές και με τους δύο αλγορίθμους, στο πρόγραμμά μας αξιοποιήσαμε τον αλγόριθμο γεμίσματος με γραμμή σάρωσης, καθώς έδειξε να είναι πιο αποδοτικός και πιο ακριβής με βάση τις απαιτήσεις της εργασίας.

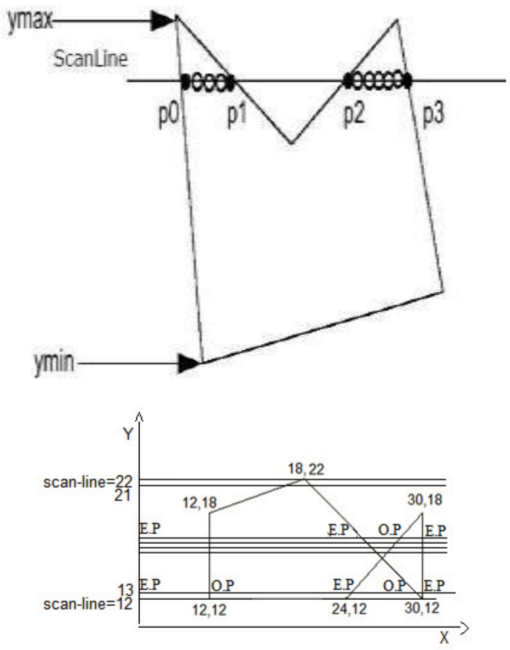


Figure 1.2: Παραδείγματα εφαρμογής του αλγορίθμου γεμίσματος με γραμμή σάρωσης

Αλγόριθμος Scanline

Ο συγκεκριμένος αλγόριθμος χρησιμοποιείται για τον υπολογισμό των σημείων που βρίσκονται στο εσωτερικό ενός πολυγώνου και την απεικόνισή τους. Ο αλγόριθμος ακολουθέι μία ευθεία γραμμή (scanline) που διατρέχει οριζόντια το χώρο του πολυγώνου και γεμίζει τον εσωτερικό χώρο του με χρώμα ή διαφορετικά γραφικά στοιχεία (στην περίπτωσή μας χρώμα). Αρχικά υπολογίζεται το ορθογώνιο περίβλημα

του πολυγώνου (bounding box). Αυτό ορίζεται από τις ελάχιστες και τις μέγιστες τιμές των x και y συντεταγμένων των κορυφών του πολυγώνου. Για κάθε γραμμή (σάρωση) μεταξύ ελάχιστης και μέγιστης τιμής y του περιβλήματος ο αλγόριθμος ξεκινάει από την αριστερή άκρη της γραμμής, ελέγχει κάθε σημείο της γραμμής και τη σχέση του με τα σημεία του πολυγώνου και αν ένα σημείο της γραμμής βρίσκεται εντός του πολυγώνου, τότε το σημείο ανήκει στο μέρος του γεμίσματος. Αντίθετα, αν ένα σημείο βρίσκεται εκτός του πολυγώνου, τότε δεν ανήκει στο χωρίο που θα γεμίσει με χρώμα. Χαρακτηριστικό του αλγορίθμου είναι πως από τη στιγμή που εισέρχεται ένα σημείο στο πολύγωνο, κάθε επόμενο σημείο της γραμμής που είναι μέρος του πολυγώνου θα πρέπει να βρίσκεται επίσης μέσα στο πολύγωνο. Μόνο όταν η γραμμή εξέρχεται από το πολύγωνο, δεν θα πρέπει να περιλαμβάνεται πλέον στο γέμισμα, κάτι που εξασφαλίζεται με τον έλεγχο κατάστασης κάθε σημείου και με ταυτόχρονο έλεγχο κυρτότητας.

Η μνήμη του συστήματος είναι καθοριστική για τον παραπάνω αλγόριθμο, καθώς ο Scanline χρειάζεται αρκετή μνήμη για να αποθηκεύσει τα δεδομένα των πολυγώνων και την εικόνα που θα δημιουργηθεί. Είναι σημαντικό να αναφερθούμε στο γεγονός πως εφαρμόζονται κατάλληλες μαθηματικές λειτουργίες τόσο για τη σάρωση κάθε γραμμής, αλλά και για τον καθορισμό εμφάνισης χρωμάτων. Ταυτόχρονα, η απόδοση του αλγορίθμου μπορεί να επηρεαστεί από την ταχύτητα πρόσβασης και εγγραφής στη μνήμη του συστήματος, αφού η αποτελεσματική διαχείριση των δεδομένων απαιτεί συχνές αναγνώσεις και εγγραφές, ειδικά για πιο σύνθετες εφαρμογές. Αναμφίβολλα, όπως κάθε αλγόριθμος, έχει πλεονεκτήματα και αδυναμίες, και η επιλογή εξαρτάται από το εκάστοτε πρόβλημα.

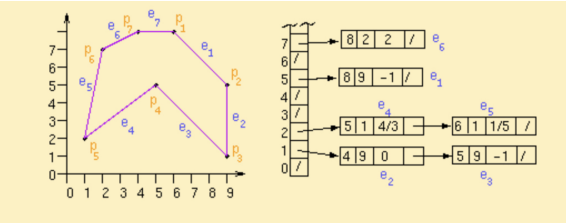


Figure 1.3: Υπολογισμός και απεικόνιση σημείων πολυγώνου

2. Υλοποίηση

Η μετάφραση των παραπάνω θεωρητικών εννοιών και αλγορίθμων σε λειτουργικά προγράμματα είναι βασικός στόχος της επιστήμης των υπολογιστών. Στην παρούσα εργασία, μετά την απόκτηση του θεωρητικού υποβάθρου σχετικά με τη χάραξη πολυγώνων και την κατανόηση των απαιτήσεων για τη μέγιστη λειτουργικότητα των αλγορίθμων ψηφιδόξυσης, επικεντρωνόμαστε στην υλοποίηση του προγράμματος χάραξης και απεικόνισης κυρτών πολυγώνων με τη χρήση της OpenGL. Στο πλαίσιο αυτό, εξετάζουμε και εξηγούμε το δοθέντα κώδικα για χάραξη τριγώνων triangle1 και έπειτα ακολουθεί η ερμηνεία της γενίκευσης σε convex1 για πολύγωνα. Σε δεύτερο στάδιο, παρουσιάζουμε βασικές εντολές που χρησιμοποιήθηκαν για την καλύτερη απόδοση του προγράμματος και αναλύουμε την ενσωμάτωση του αλγορίθμου Scanline στον κώδικά μας για την απεικόνιση των κυρτών πολυγώνων.

2.1 Η διαδικασία triangle1

Παρακάτω αναγράφεται η διαδικασία triangle1, η οποία είναι σε μορφή ψευδοκώδικα και περιγράφει έναν αλγόριθμο χάραξης τριγώνου.

```
void triangle1(Vertex v0, Vertex v1, Vertex v2, Colour c) {
    Line l0, l1, l2;
    float e0, e1, e2, e0t, e1t, e2t;
    // Υπολογισμός των συντελεστών ευθείας (a, b, c) από τις κορυφές
    mkline(v0, v1, &l0); mkline(v1, v2, &l1); mkline(v2, v0, &l2);
    // Υπολογισμός του περιβάλλοντος κυτίου του τριγώνου
    int bb_xmin = std::min(v0.x, std::min(v1.x, v2.x));
    int bb_xmax = std::max(v0.x, std::max(v1.x, v2.x));
    int bb_ymin = std::min(v0.y, std::min(v1.y, v2.y));
    int bb_ymax = std::max(v0.y, std::max(v1.y, v2.y));
    // Εκτίμηση των γραμμικών συναρτήσεων στο σημείο (bb_xmin, bb_ymin)
    e0 = l0.a * bb_xmin + l0.b * bb_ymin + l0.c;
    e1 = l1.a * bb_xmin + l1.b * bb_ymin + l1.c;
    e2 = l2.a * bb_xmin + l2.b * bb_ymin + l2.c;
```

```
    for (int y = bb_ymin; y <= bb_ymax; y++) {
        e0t = e0; e1t = e1; e2t = e2;
        for (int x = bb_xmin; x <= bb_xmax; x++) {
            if (sign(e0) == sign(e1) == sign(e2)) {
                setPixel(x, y, c);
            }
            e0 = e0 + l0.a;
            e1 = e1 + l1.a;
            e2 = e2 + l2.a;
        }
        e0 = e0t + l0.b;
        e1 = e1t + l1.b;
        e2 = e2t + l2.b;
    }
}
```

Η συνάρτηση παίρνει τρεις κορυφές ενός τριγώνου $(v0, v1, v2)$ και ένα χρώμα c ως είσοδο. Η αναπαράσταση των ευθειών που σχηματίζονται από τις τρεις ακμές του τριγώνου γίνονται με βάση τη δομή δεδομένων line. Υπολογίζεται το περιβάλλον κυτίο bounding box του τριγώνου, δηλαδή το ελάχιστο και το μέγιστο x και y που καλύπτονται από το τρίγωνο. Ο κώδικας εκτελεί μια επανάληψη για κάθε γραμμή y από το ελάχιστο ως το μέγιστο y του περιβάλλοντος κυτίου. Σε αυτήν την επανάληψη, υπολογίζονται οι γραμμικές συναρτήσεις $(e0, e1, e2)$ για το σημείο (bb_xmin, y), δηλαδή για κάθε στήλη x από το ελάχιστο ως το μέγιστο x του περιβάλλοντος κυτίου.

Στην εσωτερική επανάληψη για κάθε στήλη x από το ελάχιστο ως το μέγιστο x του περιβάλλοντος κυτίιου, γίνεται έλεγχος για το αν οι τιμές $(e0, e1, e2)$ έχουν το ίδιο πρόσημο. Αυτό υποδηλώνει ότι το σημείο (x, y) είναι εντός του τριγώνου. Σε αυτήν την περίπτωση, η συνάρτηση $setPixel(x, y)$ καλείται για να χρωματίσει το σημείο (x, y) με το καθορισμένο χρώμα c.

2.2 Η διαδικασία convex1

Η διαδικασία χάραξης τριγώνων μπορεί αποτελεσματικά να γενικευθεί σε μια διαδικασία χάραξης κυρτών πολυγώνων. Παρακάτω παρουσιάζεται συνοπτικά η διαδικασία convex1. Αρχικά ο κώδικας της συνάρτησης ελέγχει τον αριθμό των κορυφών του πολυγώνου και την κυρτότητα. Αν ο αριθμός είναι μικρότερος από τρεις

ή το πολύγωνο δεν είναι κυρτό, η συνάρτηση τερματίζει δίχως να γίνει χάραξη. Στη συνέχεια, η συνάρτηση υπολογίζει τις ευθείες που αντιπροσωπεύουν τις πλευρές του πολυγώνου, καθώς και μια αρχική εκτίμηση για κάθε κορυφή του πολυγώνου. Αυτό επιτυγχάνεται με τη συνάρτηση mkline, η οποία υπολογίζει τους συντελεστές της ευθείας που διέρχεται από δύο κορυφές. Δημιουργείται ένα διάνυσμα lines για τις πλευρές. Οι αρχικές εκτιμήσεις περιέχονται στο διάνυσμα es. 'Επειτα, υπολογίζεται το περίβλημα του πολυγώνου, δηλαδή οι ελάχιστες και μέγιστες τιμές x και y ανάμεσα στις κορυφές. Αυτό το βήμα είναι απαραίτητο για τον περιορισμό της επανάληψης του αλγορίθμου της Scanline στην περιοχή που περιβάλλει το πολύγωνο.

Με την εφαρμογή του αλγορίθμου, επιλέγεται κάθε γραμμή από το ελάχιστο y ως το μέγιστο y του περιβλήματος, ενώ ταυτόχρονα γίνεται έλεγχος της διέλευσης των γραμμών. Για να καθοριστεί εάν η κάθε γραμμή διασχίζει το πολύγωνο, δηλαδή για το εάν ένα σημείο βρίσκεται εντός ή εκτός του πολυγώνου, ο αλγόριθμος χρησιμοποιεί τη μέθοδο περιστροφής (winding number), καταμετρώντας τον αριθμό των περιστροφών που απαιτούνται για να διασχίσει το σημείο κατά μήκος του πολυγώνου. Σε κάθε γραμμή, ο κώδικας ελέγχει αν η scanline τέμνει τις πλευρές του πολυγώνου και υπολογίζει τον αριθμό των τεμνόμενων πλευρών. Αν το αποτέλεσμα είναι περιττό, τότε το σημείο ανήκει στο εσωτερικό του πολυγώνου και γίνεται η χάραξη του εικονοστοιχείου με το καθορισμένο χρώμα.

Με αυτόν τον τρόπο η παραγόμενη από το αρχέγονο τρίγωνο και τη συνάρτηση triangle1, convex1 συνάρτηση, υλοποιεί τη χάραξη (ψηφιδόξυση) ενός κυρτού πολυγώνου με τη χρήση του αλγορίθμου Scanline.

2.3 Υλοποίηση σε OpenGL

Η εργασία υλοποιήθηκε σε προγραμματιστικό περιβάλλον με τη χρήση C++/OpenGL και χρησιμοποιήθηκε η βιβλιοθήκη GLUT για τη δημιουργία γραφικού παραθύρου και την αλληλεπίδραση με το χρήστη.

Από τα στιγμιότυπα είναι εύκολη η κατανόηση του προγράμματος. Όπως φαίνεται, το γέμισμα των κυρτών πολυγώνων γίνεται επιτυχώς με μπλε χρώμα για κάθε εσωτερικό εικονοστοιχείο. Τα μπλε πολύγωνα που αποτυπώνονται αποτελούν στιγμιότυπα από το ίδιο εκτελέσιμο, με το πολύγωνο στη Figure 2.3 να αποτελεί το τελευταίο εικονιζόμενο πριν ο χρήστης διακόψει το πρόγραμμα. Στην εικόνα της Figure 2.4 δεν υπάρχει γέμισμα, καθώς αποτελεί στιγμιότυπο από δοκιμή της υλοποίησης με χρήση του αλγορίθμου Jarvis, όπου μας ενδιέφερε ο υπολογισμός του περιβάλλοντος κυτίου (convex hull). Όμως σε αυτήν την περίπτωση μπορούσαμε επιτυχώς να δίνουμε

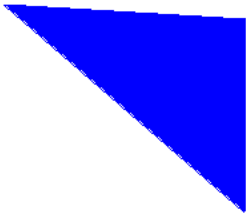


Figure 2.1: Στιγμιότυπο του εκτελέσιμου προγράμματος

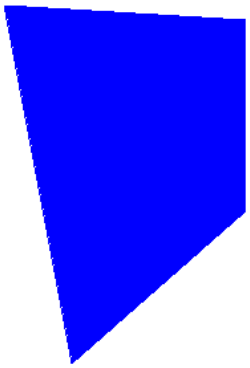


Figure 2.2: Στιγμιότυπο του εκτελέσιμου προγράμματος

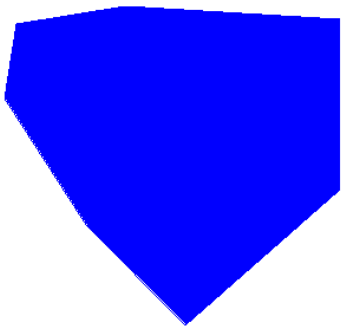


Figure 2.3: Στιγμιότυπο του εκτελέσιμου προγράμματος

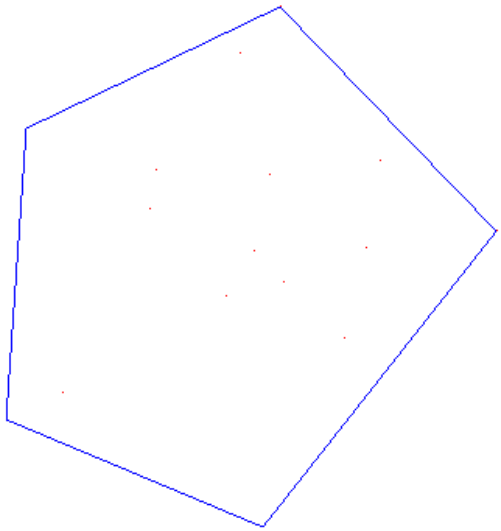


Figure 2.4: Στιγμιότυπο του εκτελέσιμου προγράμματος

στο χρήστη τη δυνατότητα να αντιληφθεί τη μορφολογία των πολυγώνων, αφού αν έδινε σημείο που σχημάτιζε εσωτερική γωνία μεγαλύτερη των $180°$, το πρόγραμμα απέτρεπε το σχηματισμό μη κυρτού πολυγώνου, αλλά εμφάνιζε το εσωτερικό σημείο στην οθόνη με κόκκινο χρώμα. Ένας συνδυασμός των δεδομένων και των μεθόδων που χρησιμοποιήσαμε θα μπορούσε να μας δώσει τη δυνατότητα να επεξεργαστούμε τα πολύγωνα και να βελτιστοποιήσουμε προγράμματα με στόχο την ευρύτερη ανάπτυξη σε ένα φάσμα πεδίων που αξιοποιούν το σχεδιασμό των 2D και 3D πολυγώνων.

Αναφορικά με τη δομή και τη σύνταξη, ο εκτελέσιμος κώδικας περιλαμβάνει τις εξής δομές δεδομένων:

- **Vertex,** ώστε να αναπαραστήσει ένα σημείο στο επίπεδο με συντεταγμένες x και y
- **Line** για να αναπαραστήσει μια γραμμή με τις παραμέτρους a, b, c και της εξίσωσης $ax + by + c = 0$
- **Colour** για να αναπαραστήσει ένα χρώμα με τις συνιστώσες r, g, b.

Οι κύριες συναρτήσεις που ορίστηκαν στον κώδικα αναγράφονται στον παρακάτω πίνακα.

Συναρτήσεις	**Ερμηνεία**
setPixel	Ορισμός ενός pixel για (x, y) με καθορισμένο χρώμα
drawLine	Σχεδίαση γραμμής από $(x1, y1$ σε $(x2, y2)$ με το καθορισμένο χρώμα
drawPolygon	Σχεδίαση πολυγώνου
isConvex	'Ελεγχος κυρτότητας
convex1	Υλοποίηση scanline και γέμισμα
display	Απεικόνιση των γραφικών.
reshape	Ενημέρωση παραμέτρων του παραθύρου
glViewPort	Προσαρμογή περιοχής απεικόνισης
keyboard	Καθορισμός προγράμματος βάση χρήστη

Table 2.1: Πίνακας Συναρτήσεων

2.3.1 Εκτελέσιμος Κώδικας

Ο κώδικας υποθέτει ότι το πολύγωνο ορίζεται από το χρήστη με τη σειρά των κλικ του ποντικιού πάνω στην οθόνη. Αυτό σημαίνει ότι το σημείο που ορίζεται κάθε φορά ενώνεται με ευθεία γραμμή με το τελευταίο σημείο. Λόγω του πεπερασμένου αριθμού των ακμών του πολυγώνου, η σχεδίασή του ολοκληρώνεται όταν δεν υπάρχει άλλο ενδεχόμενο σημείο εισαγωγής από το χρήστη που θα καθιστά το πολύγωνο κυρτό. Εναλλακτικά, η σχεδίαση ολοκληρώνεται όταν ο χρήστης πατήσει το πλήκτρο ESCAPE (ESC). Σε περίπτωση που ο χρήστης ορίσει με το ποντίκι σημεία που καθορίζουν το πολύγωνο μη κυρτό, ο έλεγχος κυρτότητας εξασφαλίζει την αποτελεσματικότητα του προγράμματος καθώς δεν θα σχεδιαστεί, και φυσικά δεν θα γεμίσει τα εσωτερικά εικονοστοιχεία του πολυγώνου.

Παρακάτω επισημαίνονται σημαντικά κομμάτια του κώδικα που εξασφαλίζουν την ομαλή και αποτελεσματική υλοποίηση σε OpenGL/C++

1. **"std::vector<Vertex> vertices"**: Είναι ένα παγκόσμιο διάνυσμα που αποθηκεύει τις κορυφές του πολυγώνου που σχεδιάζει ο χρήστης.

2. **"std::vector<Vertex> convexVertices"**: Είναι ένα διάνυσμα που χρησιμοποιείται για να αποθηκεύσει τις κορυφές του κυρτού πολυγώνου.

3. "**void setPixel(float x, float y, const Colour& colour)"**: Αυτή η συνάρτηση ορίζει το χρώμα ενός pixel στη θέση (x, y) χρησιμοποιώντας τη δομή *Colour* που έχει οριστεί.

4. **"void drawLine(float x1, float y1, float x2, float y2, const Colour& colour)"**: Αυτή η συνάρτηση σχεδιάζει μια γραμμή μεταξύ δύο σημείων $(x1, y1)$ και $(x2, y2)$ χρησιμοποιώντας τη δομή *Colour* που έχει οριστεί.

5. **"void drawPolygon(const std::vector<Vertex>&polygon, const Colour& colour)"**: Αυτή η συνάρτηση σχεδιάζει ένα πολύγωνο συνδέοντας τις κορυφές που αποθηκεύονται στο διάνυσμα polygon χρησιμοποιώντας τη δομή *Colour* που έχει οριστεί.

6. **"bool isConvex(const std::vector<Vertex>& polygon)"**: Αυτή η συνάρτηση ελέγχει εάν ένα πολύγωνο, που αναπαρίσταται από το διάνυσμα polygon είναι κυρτό ή όχι. Πραγματοποιεί υπολογισμούς διανυσματικού γινομένου για να προσδιορίσει τον προσανατολισμό των ακμών του πολυγώνου.

7. **"void convex1(const std::vector<Vertex>& polygon, const Colour& c)"**: Αυτή η συνάρτηση γεμίζει ένα κυρτό πολύγωνο, που αναπαρίσταται από το διάνυσμα polygon, χρησιμοποιώντας τον αλγόριθμο *Scanline*. Υπολογίζει τις γραμμές και τις αρχικές εκτιμήσεις για κάθε κορυφή, καθορίζει το περίγραμμα που περικλείει το πολύγωνο και γεμίζει τα εσωτερικά pixels.

8. **"void display()"**: Αυτή η συνάρτηση εμφανίζει το γραφικό παράθυρο και καλείται από τη βιβλιοθήκη OpenGL για να ανανεώνει την εικόνα στην οθόνη.

Συμπεράσματα

Η δυνατότητα απεικόνισης και αλληλεπίδρασης με γραφικά στοιχεία επιτρέπει στο χρήστη να έχει έναν οπτικό τρόπο αναπαράστασης και επεξεργασίας γεωμετρικών δομών. Σε συνολικό βαθμό ο κώδικας παρέχει μια ικανοποιητική και ευέλικτη βάση για την ανάπτυξη περαιτέρω λειτουργικοτήτων και εξέλιξη του προγράμματος. Προσφέρει ευελιξία και δυνατότητες για επέκταση και προσαρμογή. Με την υπάρχουσα υλοποίηση, μπορεί να γίνει αναπαραγωγή και αποθήκευση των κυρτών πολυγώνων, καθώς και προσθήκη επιπλέον λειτουργιών όπως η επεξεργασία και η μετακίνηση των πολυγώνων. Συνοψίζοντας, η εργασία αυτή εστίασε στην υλοποίηση ενός κώδικα για τη χάραξη αυθαίρετων κυρτών πολυγώνων με χρήση περιβάλλοντος γραφικών. Ο κώδικας επιτρέπει τη δημιουργία πολυγώνων με την επιλογή κορυφαίων σημείων μέσω του ποντικιού και προσφέρει την απεικόνιση των πολυγώνων στο γραφικό παράθυρο επιτυχώς. Η αλληλεπίδραση με το ποντίκι είναι ομαλή.

Επίλογος

Μέσω αυτής της εργασίας αποκτήθηκε η κατανόηση της σχεδίασης πολυγώνων και αρχέγονων σχημάτων δύο διαστάσεων σε γραφικό περιβάλλον. Συνολικά αυτή η εργασία μας έδωσε την ευκαιρία να εξοικειωθούμε με την ανάπτυξη εφαρμογών υπολογιστικής γεωμετρίας. Με αυτήν τη βάση μπορούμε να προχωρήσουμε σε περαιτέρω αναπτύξεις και να εφαρμόσουμε τον κώδικα σε πραγματικά προβλήματα χαραγμένων αυθαίρετων κυρτών πολυγώνων και γεωμετρικών εφαρμογών γενικότερα, θέτοντας στόχους βελτιστοποίησης.

Bibliography

[1] Δρακόπουλος, Β. (2017). *Γραφική Υπολογιστών* [Πανεπιστημιακές Σημειώσεις]. Πανεπιστήμιο Θεσσαλίας, Λαμία

[2] Δρακόπουλος, Β. (2017). *Εισαγωγή στην OpenGL* [Πανεπιστημιακές Σημειώσεις]. Πανεπιστήμιο Θεσσαλίας, Λαμία

[3] Καμωνά, Λ. (2008). *Βέλτιστες Τριγωνοποιήσεις Κυρτών Πολυγώνων*. [Μεταπτυχιακή Εργασία Εξειδίκευσης]. Αποθετήριο Πανεπιστημίου Ιωαννίνων, Ιωάννινα.

[4] Hearn, D., & Baker, M. P. (2010). *Γραφικά Υπολογιστών με OpenGL, εκδόσεις TZIOΛΑ*.

[5] Al Rawi (2014). *Implementation of an efficient Scanline-Algorithm*.

[6] Neider, J., & Davis, T. (1997). *Opengl Programming Guide: The Official Guide to Learning Opengl, Version 1.1, Second Edition*. Addison-Wesley Publishing Company.https://www.researchgate.net/publication/292183487_Scan-Line_Methods_for_Parallel_Rendering

[7] Chen, J. (1996). *Computational Geometry: Methods and Applications*. Computer Science Department, Texas A&M University.