Γραφική με Υπολογιστές

Εργασία #1: Πλήρωση Τριγώνων

ΕΠΙΜΕΛΕΙΑ: ΠΙΤΤΗΣ ΓΕΩΡΓΙΟΣ

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1. Περιγραφή της λειτουργίας και του τρόπου κλήσης των προγραμμάτων

Σκοπός της εργασίας είναι η υλοποίηση ενός αλγορίθμου πλήρωσης τριγώνων, με τελικό στόχο τον χρωματισμό μιας εικόνας και την αποθήκευση της σε ένα αρχείο png. Στα πλαίσια της εργασίας υλοποίησα τις ακόλουθες συναρτήσεις:

- **vector_interp** : χρησιμοποιεί την μέθοδο της γραμμικής παρεμβολής για να υπολογίσει το διάνυσμα χρώματος V(1x3 διάνυσμα) ενός σημείου p , το οποίο βρίσκεται πάνω στο ευθύγραμμο τμήμα που ορίζουν τα σημεία p_1 και p_2. Καλώ την συνάρτηση δίνοντας ως ορίσματα εισόδου τις συντεταγμένες (x_1, y_1) και (x_2, y_2) των σημείων p_1 και p_2 αντίστοιχα , τα διανύσματα χρώματος V_1 και V_2 των σημείων p_1 και p_2 αντίστοιχα καθώς και μια συντεταγμένη(coord) του σημείου p που καθορίζεται από την τιμή του ορίσματος εισόδου dim. Αν $\dim = 1$, τότε εισάγω στην συνάρτηση την τετμημένη x του σημείου p. Διαφορετικά, αν $\dim = 2$, τότε εισάγω την τεταγμένη y του σημείου p. Η συνάρτηση επιστρέφει το διάνυσμα V του σημείου p.

- **line_drawing :** εφαρμόζει τον αλγόριθμο του Bresenham για τη σχεδίαση γραμμών. Με αυτή την συνάρτηση δημιουργώ τις πλευρές κάθε τριγώνου της εικόνας καθώς και τις γραμμές σάρωσης. Λαμβάνει ως είσοδο τις συντεταγμένες (x_1, y_1) και (x_2, y_2) των σημείων vert1 και vert2 αντίστοιχα. Το vert1 είναι το αρχικό σημείο της γραμμής και το vert2 είναι το τελικό σημείο της γραμμής που θέλω να σχεδιάσω. Η συνάρτηση επιστρέφει τη λίστα points που περιέχει τις συντεταγμένες όλων των σημείων της γραμμής που σχεδίασα. Ο αλγόριθμος του Bresenham που υλοποιώ βασίζεται στις σημειώσεις του μαθήματος.

- **f_shading :** με αυτή την συνάρτηση υλοποιώ τον χρωματισμό ενός τριγώνου της ζητούμενης εικόνας, χρησιμοποιώντας την μέθοδο του flat shading. Πιο συγκεκριμένα, βάφω όλα τα pixels ενός τριγώνου με το ίδιο χρώμα. Δηλαδή, βάφω τα pixels που βρίσκονται πάνω στις πλευρές του τριγώνου καθώς και τα pixels που βρίσκονται στο εσωτερικό του τριγώνου με χρώμα ίσο με τον διανυσματικό μέσο όρο των χρωμάτων των τριών κορυφών του τριγώνου. Αν αυτή η μέθοδος εφαρμοστεί για όλα τα τρίγωνα της εικόνας , τότε πολλά γειτονικά τρίγωνα Θα διαφέρουν αρκετά ως προς το χρώμα τους. Για τον λόγο αυτόν, λοιπόν, Θα διακρίνονται εμφανή τρίγωνα διαφορετικών χρωμάτων στη τελική εικόνα. Η συνάρτηση δέχεται ως όρισμα εισόδου τον MxNx3 πίνακα img (την εικόνα δηλαδή, η οποία διαθέτει τυχόν προϋπάρχοντα τρίγωνα). Στην τρίτη διάσταση του πίνακα img βρίσκονται τα 1x3 διανύσματα χρώματος κάθε pixel της εικόνας. Επίσης, λαμβάνει ως είσοδο τον 3x2 πίνακα vertices, ο οποίος περιέχει τις 2D συντεταγμένες των κορυφών ενός τριγώνου. Ακόμη, δέχεται ως είσοδο τον 3x3 πίνακα vcolors , στον οποίο

βρίσκονται αποθηκευμένα τα διανύσματα χρώματος των κορυφών του τριγώνου. Επιστρέφει την εικόνα img χρωματισμένη.

- **g_shading :** με αυτή την συνάρτηση υλοποιώ τον χρωματισμό ενός τριγώνου της ζητούμενης εικόνας, χρησιμοποιώντας την μέθοδο του Gouraud για την πλήρωση τριγώνων. Ειδικότερα, βρίσκω το χρώμα των pixels ενός τριγώνου κάνοντας γραμμική παρεμβολή. Επομένως, η g_shading χρησιμοποιεί την συνάρτηση vector_interp για τον σκοπό αυτό. Αν αυτή η μέθοδος εφαρμοστεί για όλα τα τρίγωνα της εικόνας, τότε τα γειτονικά pixels θα έχουν πολύ παρόμοιο χρώμα και συνακόλουθα τα γειτονικά τρίγωνα δεν θα παρουσιάζουν έντονες χρωματικές διαφορές. Δηλαδή, η χρωματική μεταβολή μεταξύ των τριγώνων θα συμβαίνει με πολύ ομαλό τρόπο. Αυτός είναι ο λόγος που εξαφανίζονται τα τρίγωνα στην τελική εικόνα και προκύπτει ένα αρκετά πιο όμορφο αποτέλεσμα. Η συγκεκριμένη συνάρτηση δέχεται τα ίδια ορίσματα εισόδου με την f_shading. Επιστρέφει την εικόνα img χρωματισμένη.

- **render_img:** με αυτή την συνάρτηση πραγματοποιώ τον χρωματισμό της ζητούμενης εικόνας είτε με την μέθοδο flat shading είτε με την μέθοδο gouraud shading. Λαμβάνει ως είσοδο τον Lx2 πίνακα vertices που περιέχει τις 2D συντεταγμένες των κορυφών όλων των τριγώνων της εικόνας(L είναι το συνολικό πλήθος των κορυφών των τριγώνων της εικόνας). Επιπλέον, λαμβάνει ως είσοδο τον Kx3 πίνακα faces, ο οποίος περιέχει τις κορυφές των K τριγώνων της εικόνας(K = συνολικά τρίγωνα εικόνας). Κάθε γραμμή του πίνακα faces είναι ένα 1x3 διάνυσμα, όπου κάθε στοιχείο του είναι ένας ακέραιος αριθμός που αναφέρεται στις συντεταγμένες μιας συγκεκριμένης κορυφής τριγώνου του πίνακα vertices(π.χ. το 0 αναφέρεται στις συντεταγμένες της 1ης κορυφής του πίνακα vertices). Ακόμη, δέχεται ως είσοδο τον Lx3 πίνακα vcolors , ο οποίος περιέχει τα διανύσματα χρώματος των κορυφών όλων των τριγώνων της εικόνας. Επίσης, στα ορίσματα εισόδου συγκαταλέγονται ο Lx1 πίνακας depth που περιέχει το βάθος κάθε κορυφής τριγώνου καθώς και η μεταβλητή shading που καθορίζει την μέθοδο πλήρωσης των τριγώνων της εικόνας (flat ή gouraud shading). Αν shading = "f" ,τότε στο εσωτερικό της συνάρτησης καλείται η f_shading. Διαφορετικά αν shading = "g" τότε καλείται η συνάρτηση g_shading. Η συνάρτηση αρχικοποιεί τον καμβά διαστάσεων 512x512 με άσπρο χρώμα και υπολογίζει το βάθος του κάθε τριγώνου της εικόνας ως το κέντρο βάρους τους βάθους των κορυφών του. Στην συνέχεια, ταξινομεί κατάλληλα τα τρίγωνα, ώστε να βρίσκεται πρώτο εκείνο με το μεγαλύτερο βάθος και τελευταίο εκείνο με το μικρότερο βάθος. Αυτό συμβαίνει διότι ο χρωματισμός της εικόνας ξεκινά από τα τρίγωνα με το μεγαλύτερο βάθος και συνεχίζει με εκείνα που έχουν μικρότερο βάθος. Επιστρέφει την έγχρωμη εικόνα img διάστασης MxNx3.

2. Περιγραφή της διαδικασίας χρωματισμού των τριγώνων και παρουσίαση του αντίστοιχου ψευδοκώδικα

f_shading:

Αρχικά, βρίσκω τη μέγιστη τετμημένη(x_max) , την ελάχιστη τετμημένη(x_min), τη μέγιστη τεταγμένη(y_max) και την ελάχιστη τεταγμένη(y_min) κάθε πλευράς του τριγώνου. Από τις παραπάνω συντεταγμένες βρίσκω τις αντίστοιχες συντεταγμένες x_max, x_min, y_max, y_min του τριγώνου. Ορίζω τον μετρητή του πλήθους των ενεργών σημείων (active_points_counter) και τον αρχικοποιώ σε 0. Για κάθε τεταγμένη y του τριγώνου, δημιουργώ μια γραμμή σάρωσης. Αυτή η γραμμή σάρωσης είναι επαρκώς μεγάλη ώστε να μπορεί να σαρώσει όλα τα pixel του τριγώνου που βρίσκονται στην συγκεκριμένη τεταγμένη y. Η γραμμή σάρωσης , σαρώνει τόσο τα pixel που βρίσκονται πάνω στις πλευρές του τριγώνου και αποτελούν τα σημεία τομής των πλευρών του τριγώνου με αυτήν(ενεργά σημεία), όσο και τα pixel που βρίσκονται εντός του τριγώνου. Ορίζω την λίστα active_points στην οποία αποθηκεύω τις συντεταγμένες των ενεργών σημείων που συναντώ. Αν κάποιο σημείο που βρίσκεται σε κάποια πλευρά του τριγώνου συμπίπτει με κάποιο σημείο της γραμμής σάρωσης, τότε προσθέτω τις συντεταγμένες του συγκεκριμένου σημείου στην λίστα active_points και επαυξάνω τον μετρητή active_points_counter κατά 1. Μόλις σαρωθούν όλα τα pixels του τριγώνου που βρίσκονται στη συγκεκριμένη τεταγμένη y του τριγώνου, ψάχνω στην λίστα active_points και εντοπίζω το μέγιστο και ελάχιστο ενεργό σημείο ως προς την τετμημένη x. Έτσι, μπορώ να γνωρίζω τις συντεταγμένες όλων των σημείων του τριγώνου σε αυτό το y. Αν η γραμμή σάρωσης πέρασε από κορυφή του τριγώνου τότε θα έχω active_points_counter = 1, οπότε δεν χρωματίζω. Διαφορετικά, (δηλαδή αν active_points_counter ≠ 1) τότε χρωματίζω κάθε pixel του τριγώνου , που βρίσκεται στη συγκεκριμένη τεταγμένη y, με χρώμα ίσο με τον διανυσματικό μέσο όρο των χρωμάτων των τριών κορυφών.

Ψευδοκώδικας f_shading:

(ο συγκεκριμένος ψευδοκώδικας εκτελείται 1 φορά για κάθε τρίγωνο της εικόνας)

```
for k = 0 : 1 : 3-1
    Βρίσκουμε τα xk_max, xk_min, yk_max, yk_min της k-οστής πλευράς του τριγώνου
end
```

Βρίσκω το x_max του τριγώνου :

x_max = max_xk {x1_max, x2_max, x3_max}

Βρίσκω το x_min του τριγώνου :

x_min = min_xk {x1_min, x2_min, x3_min}

Βρίσκω το y_max του τριγώνου :

y_max = max_yk {y1_max, y2_max, y3_max}

Βρίσκω το y_min του τριγώνου :

y_min = min_yk {y1_min, y2_min, y3_min}

Αρχικοποιώ τον μετρητή ενεργών σημείων:

active_points_counter = 0

for y = y_min : 1 : y_max

Χαράζω την γραμμή σάρωσης με αρχικό σημείο το [x_min, y] και τελικό σημείο το [x_max, y]

Αρχικοποιώ την <Λίστα ενεργών σημείων> σε κενή λίστα για την γραμμή σάρωσης με τεταγμένη y

Κάθε φορά που η γραμμή σάρωσης συναντά ενεργό σημείο , το προσθέτω στη <Λίστα ενεργών σημείων>

Κάθε φορά που η γραμμή σάρωσης συναντά ενεργό σημείο , ανανεώνω τον μετρητή ενεργών σημείων :

active_points_counter = active_points_counter + 1 ;

Βρίσκω την τελική <Λίστα ενεργών σημείων>

Υπολογίζω το μέγιστο και το ελάχιστο ενεργό σημείο ως προς την τετμημένη x :

max_point, min_point

```
        if (active_points_counter είναι διάφορο του 1)
            for i = min_point[0]: 1 : max_point[0]
                Βάφω το pixel με συντεταγμένες [i, y] με χρώμα ίσο με
                τον διανυσματικό μέσο όρο των χρωμάτων των τριών κορυφών
            end
        end

        Αρχικοποιώ τον μετρητή ενεργών σημείων:
        active_points_counter = 0

end
```

g_shading:

Αρχικά, βρίσκω τη μέγιστη τετμημένη(x_max) , την ελάχιστη τετμημένη(x_min), τη μέγιστη τεταγμένη(y_max) και την ελάχιστη τεταγμένη(y_min) κάθε πλευράς του τριγώνου. Από τις παραπάνω συντεταγμένες βρίσκω τις αντίστοιχες συντεταγμένες x_max, x_min, y_max, y_min του τριγώνου. Ορίζω τον μετρητή του πλήθους των ενεργών σημείων (active_points_counter) και τον αρχικοποιώ σε 0. Για κάθε τεταγμένη y του τριγώνου, δημιουργώ μια γραμμή σάρωσης. Αυτή η γραμμή σάρωσης είναι επαρκώς μεγάλη ώστε να μπορεί να σαρώσει όλα τα pixel του τριγώνου που βρίσκονται στην συγκεκριμένη τεταγμένη y. Η γραμμή σάρωσης , σαρώνει τόσο τα pixel που βρίσκονται πάνω στις πλευρές του τριγώνου και αποτελούν τα σημεία τομής των πλευρών του τριγώνου με αυτήν(ενεργά σημεία), όσο και τα pixel που βρίσκονται εντός του τριγώνου. Ορίζω την λίστα active_points στην οποία αποθηκεύω τις συντεταγμένες των ενεργών σημείων που συναντώ. Χρωματίζω με γραμμική παρεμβολή τα σημεία των πλευρών του τριγώνου που βρίσκονται στην γραμμή σάρωσης. Δηλαδή χρωματίζω τα κοινά σημεία μεταξύ των πλευρών του τριγώνου και της γραμμής σάρωσης. Στη συνέχεια προσθέτω τις συντεταγμένες αυτών των κοινών σημείων στην λίστα active_points και επαυξάνω τον μετρητή active_points_counter κατά 1. Μόλις σαρωθούν όλα τα pixels του τριγώνου που βρίσκονται στη συγκεκριμένη τεταγμένη y του τριγώνου, ψάχνω στην λίστα active_points και εντοπίζω το μέγιστο και ελάχιστο ενεργό σημείο ως προς την τετμημένη x. Έτσι, μπορώ να γνωρίζω τις συντεταγμένες όλων των σημείων του τριγώνου σε αυτό το y. Αν η γραμμή σάρωσης πέρασε από κορυφή του τριγώνου τότε θα έχω active_points_counter = 1, οπότε δεν χρωματίζω. Διαφορετικά, (δηλαδή αν active_points_counter ≠ 1) τότε χρωματίζω κάθε εσωτερικό pixel του τριγώνου , που βρίσκεται στη συγκεκριμένη τεταγμένη y, με γραμμική παρεμβολή. Άρα, υλοποιώ γραμμική παρεμβολή ξεχωριστά για τα pixels των πλευρών του

τριγώνου και ξεχωριστά για τα εσωτερικά pixels του τριγώνου. Αν δεν χρωματιστούν πρώτα όλα τα pixels των πλευρών του τριγώνου, τότε στην συνέχεια ο χρωματισμός των εσωτερικών pixel δεν μπορεί να πραγματοποιηθεί και συνεπώς θα πάρω σαν αποτέλεσμα τον λευκό καμβά.

Ψευδοκώδικας g_shading:

(ο συγκεκριμένος ψευδοκώδικας εκτελείται 1 φορά για κάθε τρίγωνο της εικόνας)

for k = 0 : 1 : 3-1

Βρίσκουμε τα xk_max, xk_min, yk_max, yk_min της k-οστής πλευράς του τριγώνου

end

Βρίσκω το x_max του τριγώνου :

x_max = max_xk {x1_max, x2_max, x3_max}

Βρίσκω το x_min του τριγώνου :

x_min = min_xk {x1_min, x2_min, x3_min}

Βρίσκω το y_max του τριγώνου :

y_max = max_yk {y1_max, y2_max, y3_max}

Βρίσκω το y_min του τριγώνου :

y_min = min_yk {y1_min, y2_min, y3_min}

Αρχικοποιώ τον μετρητή ενεργών σημείων:

active_points_counter = 0

for y = y_min : 1 : y_max

Χαράζω την γραμμή σάρωσης scan_line με αρχικό σημείο το [x_min, y] και τελικό σημείο το [x_max, y]

Αρχικοποιώ την <Λίστα ενεργών σημείων> σε κενή λίστα για την γραμμή σάρωσης με τεταγμένη y

for j = scan_line[0] : 1 : scan_line[-1]

if (j υπάρχει σε κάποια πλευρά του τριγώνου)

Βάφω με γραμμική παρεμβολή το pixel με τις συντεταγμένες

του σημείου j

end

end

Κάθε φορά που η γραμμή σάρωσης συναντά ενεργό σημείο , το προσθέτω στη <Λίστα ενεργών σημείων>

Κάθε φορά που η γραμμή σάρωσης συναντά ενεργό σημείο , ανανεώνω τον μετρητή ενεργών σημείων :

active_points_counter = active_points_counter + 1 ;

Βρίσκω την τελική <Λίστα ενεργών σημείων>

Υπολογίζω το μέγιστο και το ελάχιστο ενεργό σημείο ως προς την τετμημένη x :

max_point, min_point

if (active_points_counter είναι διάφορο του 1)

for i = min_point[0] + 1 : 1 : max_point[0] -1

Βάφω με γραμμική παρεμβολή το pixel με συντεταγμένες [i , y]

end

end

Αρχικοποιώ τον μετρητή ενεργών σημείων:

active_points_counter = 0

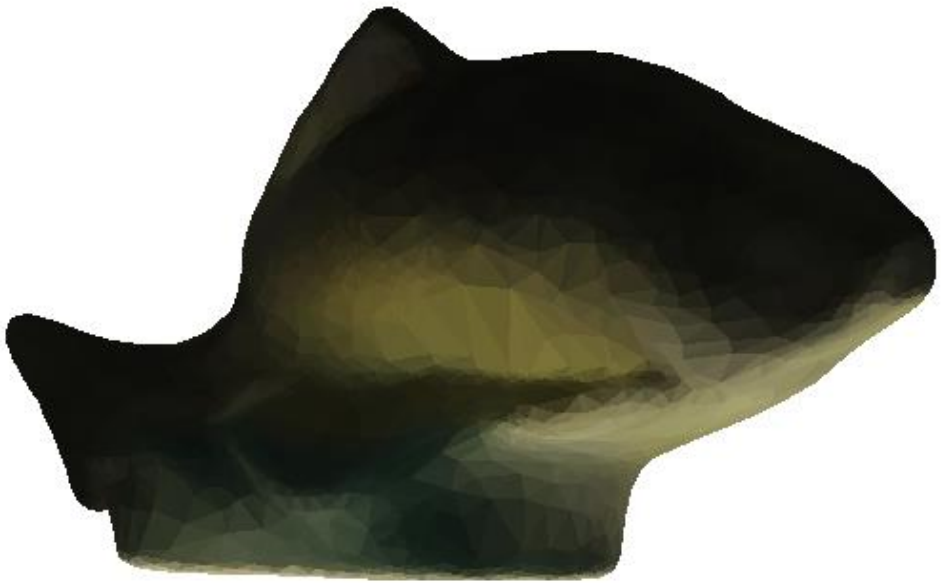
end

3. Περιγραφή των παραδοχών που χρησιμοποίησα

- Ο καμβάς έχει διαστάσεις $M = 512$, $N = 512$.
- Το background του καμβά είναι λευκό ($rgb = (1,1,1)$).
- Στην συνάρτηση vector_interp υπέθεσα ότι το σημείο p βρίσκεται μεταξύ των σημείων p_1 και p_2.
- Υπέθεσα ότι οι κορυφές όλων των τριγώνων βρίσκονται εντός του καμβά.

4. Τα ενδεικτικά αποτελέσματα που παράγονται από τα demos.

- Χρωματισμός εικόνας με την μέθοδο flat shading
  (παράγεται από το demo_f.py)



Εικόνα 1: "Flat" shading

- Χρωματισμός εικόνας με την μέθοδο gouraud shading
  (παράγεται από το demo_g.py)

Εικόνα 2: "Gouraud" shading

Κατάφερα να απεικονίσω και να αποθηκεύσω την ζητούμενη εικόνα τόσο με συναρτήσεις της βιβλιοθήκης matplotlib όσο και με συναρτήσεις της βιβλιοθήκης OpenCV. Για να τρέξουν τα demos πρέπει όλα τα αρχεία κώδικα καθώς και το hw1.npy να βρίσκονται στο ίδιο project.