Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Ηλεκτρονικής και Υπολογιστών

ΓΡΑΦΙΚΗ ΜΕ ΥΠΟΛΟΓΙΣΤΕΣ

Εργασία 1η : Πλήρωση Τριγώνων

Εαρινό εξάμηνο 2023/24

Δεϊρμεντζόγλου Ιωάννης
Τομέας Ηλεκτρονικής και Υπολογιστών
A.E.M.: 10015
Email: deirmentz@ece.auth.gr

ΕΙΣΑΓΩΓΗ

Ζητούμενο στην παρούσα εργασία ήταν η δημιουργία ενός αλγορίθμου με κατάλληλη προσαρμογή ώστε να χειρίζεται την ειδική περίπτωση των τριγώνων (μετατροπή σάρωσης γραμμών, σκίαση με μεθόδους flat ή gouraud, χρωματισμό, ερμηνεία σε καμβά) με τον οποίο επιτυγχάνεται η πλήρωση κλειστών πολυγώνων και συγκεκριμένα τριγώνων με τελικό στόχο την απεικόνιση μιας εικόνας.

Βοηθητικές συναρτήσεις

- **bresenhamAlgorithm (vertexPoint1, vertexPoint2, axis)**
  Η συναρτηση αυτή υλοποιεί τον αλγόριθμο γραμμικής σχεδίασης του bresenham για 2 κορυφές του τρίγωνου. Δέχεται ως ορίσματα τις συντεταγμένες των κορυφών και τον άξονα κατά τον οποίο υπολογίζεται η ευθεία. Ανάλογα, με τον άξονα που επιλέγεται υπολογίζονται το αρχικό και το τελικό σημείο της ευθείας και ακολουθείται η διαδικασία που περιγράφεται στις σημειώσεις με τον υπολογισμό των deltaX και deltaY, χρησιμοποιεί την μεταβλητή f για να παρακολουθήσει την πορεία της γραμμής, διατρέχει τα pixels κατά μήκος της γραμμής, ενημερώνει το x, y βάσει της f κλπ. Ο αλγόριθμος επιστρέφει έναν Mx2 πίνακα με τις συντεταγμένες των pixels που ανήκουν στην γραμμή.

  Ψευδοκώδικας για τον άξονα x :

```
Αν axis = x
   deltaX = 2 * (x1 - x0) και deltaY = 2 * |y1 - y0|
   f = -deltaY + deltaX / 2.
   Τρέχον σημείο (x, y) = (x0, y0).

   Για κάθε τιμή x από x0+1 μέχρι x1(δεν συμπεριλαμβάνεται το x1):
      Εάν f  < 0 τότε
          Εάν y0 < y1 τότε
              y = y + 1 (μετακίνηση προς τα πάνω)
          Εάν y0 > y1 τότε
              y = y - 1 (μετακίνηση προς τα κάτω)
          f = f + deltaX
       f = f - deltaY
       Πρόσθεσε το τρέχον σημείο (x, y) στον πίνακα edgePixels.
```

- **fillLine (vertexPoint1, vertexPoint2)**
  Η συναρτηση αυτή ουσιαστικά ‘’σχεδιάζει’’ το ευθύγραμμο τμήμα μεταξύ 2 κορυφών. Τα vertexPoint1, vertexPoint2 αποτελούν τις συντεταγμένες 2 κορυφών του τρίγωνου και επιστρέφονται οι συντεταγμένες των pixels που ανήκουν στη γραμμή που συνδέει αυτές τις δύο κορυφές. Αρχικά, ελέγχεται αν οι 2 κορυφές έχουν κάποια ιδιά συντεταγμένη (παράλληλες είτε ως προς τον άξονα x ή τον άξονα y). Εάν συμβαίνει κάτι τέτοιο είναι προφανές ότι

επιλέγονται όλα τα pixels αναμεσά στις 2 κορυφές με την ίδια κοινή συντεταγμένη. Δηλαδή πχ. για την περιπτωση με το ίδιο x το εύρος είναι : **[min(vertexPoint1[1],vertexPoint2[1]), max(vertexPoint1[1], vertexPoint2[1])] .**
Αν οι δύο κορυφές δεν είναι ούτε στην ίδια γραμμή ούτε στην ίδια στήλη, τότε εκτελείται ο αλγόριθμος του bresenham αφού πρώτα υπολογιστεί η κλίση της ευθείας και επιλεχθεί με βάσει αυτή ο άξονας x ή y. Παρατίθεται και ο αντίστοιχος ψευδοκώδικας :

```
# Υπολογισμός κλίσης
slope = |vertexPoint1[0] - vertexPoint2[0]| / |vertexPoint2[1] - vertexPoint1[1]|
if |slope| < 1:
    # Bresenham στον άξονα y
    edgePoints = bresenhamAlgorithm(vertexPoint1, vertexPoint2, axis=1)
else:
    # Bresenham στον άξονα x
    edgePoints = bresenhamAlgorithm(vertexPoint1, vertexPoint2, axis=0)
```

Συναρτηση Γραμμικής Παρεμβολής

- **vector_interp(p1, p2, V1, V2, coord, dim)**

  Η συνάρτηση πραγματοποιεί γραμμική παρεμβολή ανάμεσα σε δύο ορισμένα σημεία, για τη τιμή του διανύσματος σε μια ορισμένη θέση, που δίνεται από ένα ζεύγος συντεταγμένων. Τα διανύσματα V1,V2 ουσιαστικά είναι 1x3 και αποτελούν τις τιμές χρώματος για τα αντίστοιχα σημεία. Ανάλογα με την επιλεγμένη διάσταση, υπολογίζει τις κλίσεις v1Coeff, v2Coeff με τη χρήση γραμμικής παρεμβολής και ακολουθείται η λογική του κυρτού γραμμικού συνδυασμού. Τέλος, επιστρέφει την εκτιμώμενη τιμή του διανύσματος στο σημείο παρεμβολής.

  Ακολουθεί ο ψευδοκώδικας :

```
vector_interp(p1, p2, V1, V2, coord, dim)

    Εάν dim == 1 τότε
        x = coord
        Αν x > max(x1, x2) τότε
            Εκτός του εύρους [x1, x2]
        Αν |x2 - x1| < 1e-3 τότε
            Αν |x - x1| < |x - x2| τότε
                Επιστροφή V1
            Διαφορετικά
                Επιστροφή V2
        v1Coeff = |x2 - x| / |x2 - x1|
        v2Coeff = |x - x1| / |x2 - x1|
        V = V1 * v1Coeff + V2 * v2Coeff
```

```
        Επιστροφή V

    Διαφορετικά εάν dim == 2 τότε
        y = coord
        Αν y > max(y1, y2) τότε
            Εκτός του εύρους [x1, x2]

        Αν  |y2 - y1| < 1e-3 τότε
            Αν |y - y1| < |y - y2| τότε
                Επιστροφή V1
            Διαφορετικά
                Επιστροφή V2
        v1Coeff = |y2 - y| / |y2 - y1|
        v2Coeff = |y - y1| / |y2 - y1|
        V = V1 * v1Coeff + V2 * v2Coeff
        Επιστροφή V
```

Συναρτηση Flat Shading

- ***updated_img = f_shading (img, vertices, vcolors)***

  Η συνάρτηση δέχεται ως εισόδους μια εικόνα img, έναν πίνακα με τις συντεταγμένες των κορυφών του τριγώνου vertices και τον πίνακα με τα χρώματα των κορυφών vcolors. Αρχικά η συναρτηση υπολογίζει τα ελάχιστα και μέγιστα άκρα τόσο για την τετμημένη όσο και για την τεταγμένη (xk,yk,min,max), ενώ παράλληλα εκτελούνται έλεγχοι για οριζόντιες και κάθετες πλευρές στο τρίγωνο. Στη συνέχεια, υπολογίζει τις συντεταγμένες των pixel που ανήκουν σε κάθε πλευρά του τριγώνου χρησιμοποιώντας τη συνάρτηση fillLine. Έπειτα, υπολογίζει το μέσο χρώμα του τριγώνου και χρωματίζει τις περιοχές μεταξύ των πλευρών με αυτό το μέσο διανυσματικό χρώμα.

  Έπειτα, υλοποιεί τον αλγόριθμο σάρωσης γραμμής (scanline) για την πλήρωση του τριγώνου με το μέσο χρώμα των κορυφών αφού υπολογιστούν η κατωτάτη και ανώτερη γραμμή σάρωσης. Ακολουθεί ο υπολογισμός των αρχικών ενεργών γραμμών και των αρχικών ενεργών σημείων του τριγώνου. Στη συνέχεια, υλοποιείται μια διαδικασία κατά την οποία ξεκινά η σάρωση από τις γραμμές σάρωσης. Η σάρωση ξεκινάει από τις ελάχιστες τιμές των τεταγμένων και καταλήγει στις μέγιστες. Έτσι, υπολογίζονται τα διαστήματα πλήρωσης που άκρα τους είναι τα αντίστοιχα ενεργά οριακά σημεία εκείνη την στιγμή. Τελικά, χρωματίζονται αυτά τα διαστήματα από το ένα οριακό σημείο ως το άλλο.

  Βήματα

  Α) επαναληπτικά για κάθε πλευρά του τριγώνου υπολογίζει τις μέγιστες τεταγμένες και τετμημένες αλλά και την κλίση της(Bresenham)

  Β) τοποθετεί τις γραμμές σάρωσης από τη μικρότερη στη μεγαλύτερη

Γ) υπολογίζει τα διαστήματα πλήρωσης
Δ) επαναληπτικά σαρώνοντας την λίστα από τη μικρότερη τεταγμένη μέχρι τη μεγαλύτερη κάνει τα παρακάτω βήματα: βρίσκει τα ενεργά οριακά σημεία, τα ταξινομεί, θέτει την ελάχιστη τετμημένη ως το πρώτο ενεργό οριακό σημείο, θέτει την μέγιστη τετμημένη ως το δεύτερο ενεργό οριακό σημείο και τώρα ξεκινάει επαναληπτικά σαρώνοντας την λίστα από την μικρότερη τετμημένη μέχρι τη μεγαλύτερη να χρωματίζει τα εσωτερικά σημεία του διαστήματος πλήρωσης.

Επιστρέφει την ανανεωμένη εικόνα ίδιας διάστασης με την img με τα ήδη προ υπάρχοντα και χρωματισμένα τρίγωνα αλλά και που περιέχει τις τιμές RGB για όλα τα μέρη του τριγώνου.

Ακολουθεί ο ψευδοκώδικας :

```
f_shading
 Για κάθε k από 1 έως 3 (αριθμός κορυφών)
     Αν k είναι 3 τότε
        x_k_min[k]  =  min(κορυφές[k][0],  κορυφές[k-
2][0])
        x_k_max[k]  =  max(κορυφές[k][0],  κορυφές[k-
2][0])
        y_k_min[k]  =  min(κορυφές[k][1],  κορυφές[k-
2][1])
        y_k_max[k] = max(κορυφές[k][1], κορυφές[k -
2][1])
      Αλλιώς
        x_k_min[k] = min(κορυφές[k][0], κορυφές[k +
1][0])
        x_k_max[k] = max(κορυφές[k][0], κορυφές[k +
1][0])
        y_k_min[k] = min(κορυφές[k][1], κορυφές[k +
1][1])
        y_k_max[k] = max(κορυφές[k][1], κορυφές[k +
1][1])

  y_min = min(y_k_min)
  y_max = max(y_k_max)

  edge1 = fillLine(vertex Point1, vertex Point2)
  edge2 = fillLine(vertex Point2, vertex Point3)
  edge3 = fillLine(vertex Point3, vertex Point1)

  Έλεγχος για οριζόντιες γραμμές στην κορυφή και τη
βάση

  Ταξινόμηση ακμών edge1, edge2, edge3 ανάλογα με τις
συντεταγμένες x
```

```
  Υπολογισμός μέσου χρώματος meanColor = (v1Color +
v2Color + v3Color) / 3

  Γέμισμα ακμών edge1, edge2, edge3 με το μέσο χρώμα
meanColor

    Υπολογισμός bottomScanline
    Υπολογισμός topScanline

     Αλγόριθμος Σάρωσης Γραμμής (Scanline Algorithm)
       Για κάθε scanline από bottomScanline έως
topScanline :
          Αντιστοίχιση σημείων activePoints στην
τρέχουσα scanline
          Αν το activePoints έχει μήκος 1 τότε
             Συνέχισε στο επόμενο βήμα
          Διαφορετικά
             Υπολογισμός min_col_active = ελάχιστο x
σημείου στο activePoints
             Υπολογισμός max_col_active = μέγιστο x
σημείου στο activePoints

       Γέμισμα των σημείων με το μέσο χρώμα meanColor
ανάμεσα σε min_col_active και max_col_active

        Επιστροφή εικόνας
```

Συναρτηση Gouraud Shading

- ***updated_img = g_shading (img, vertices, vcolors)***

  Η συναρτηση αυτή είναι παρόμοια με την προηγουμένη. Η διαφορά έγκειται στον τρόπο υπολογισμού του χρώματος του κάθε pixel. Στην συγκεκριμένη περιπτωση αξιοποιείται η συναρτηση vector_interp και πραγματοποιεί δύο φορές γραμμική παρεμβολή, ώστε να βρει με τι χρώμα θα χρωματίσει τα εσωτερικά των τριγώνων, ακολουθώντας τον ίδιο αλγόριθμο για να βρει τα τρίγωνα. Αρχικά υπολογίζεται το χρώμα των προβολών του σημείου κατά τον άξονα y στις ακμές του τριγώνου, και στη συνέχεια το χρώμα που προκύπτει είναι η γραμμική παρεμβολή των χρωμάτων αυτών των σημείων-προβολών.

Συνάρτηση χρωματισμού αντικειμένου

- ***img = render_img (faces, vertices, vcolors, depth, shading)***

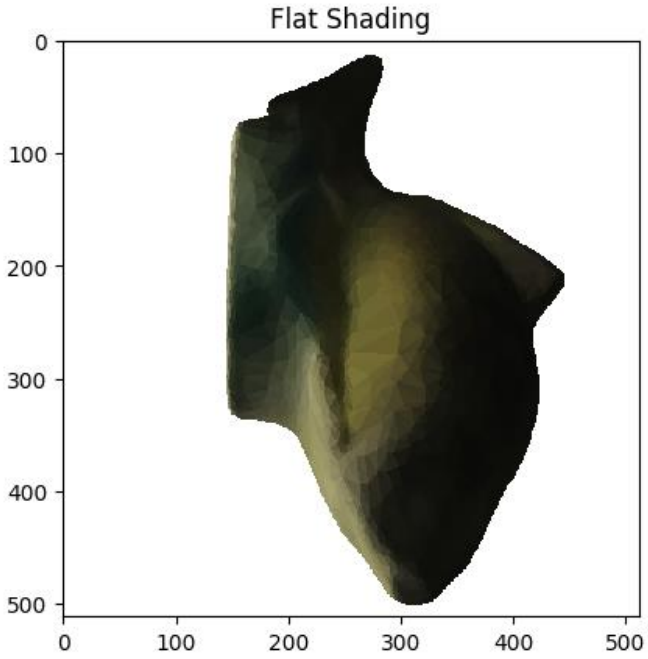
  Η συνάρτηση δέχεται ως εισόδους τον πίνακα με τις τριάδες των δεικτών που αναπαριστούν τα τρίγωνα faces, τον πίνακα με τις συντεταγμένες των κορυφών vertices, έναν πίνακα με τα χρώματα των κορυφών vcolors, έναν πίνακα με τα βάθη depth, και τον τρόπο σκίασης shading. Κατατάσσει αρχικά τα faces με βάση το βάθος από τη πιο βαθιά κορυφή στην λιγότερο βαθιά και στη συνέχεια καλεί την αντίστοιχη συνάρτηση χρωματισμού ανάλογα με το όρισμα που έχει δοθεί. Πιο συγκεκριμένα η συναρτηση, αρχικοποιεί τον καμβά σε άσπρο φόντο 512x512. Στην συνέχεια, επαναληπτικά για κάθε τρίγωνο του πίνακα faces υπολογίζει το βάθος κάθε κορυφής ως τον μέσο όρο των βαθών των 3 κορυφών και τριγώνου συνολικά και ταξινομεί τα τρίγωνα με σειρά τέτοια ώστε πρώτο να είναι αυτό με το μικρότερο βάθος και τελευταίο αυτό με το μεγαλύτερο. Επαναληπτικά για κάθε τρίγωνο διαλέγει τις κορυφές κάθε τριγώνου και τις συντεταγμένες αυτών από την RGB λίστα και καλεί είτε την συνάρτηση f_shading() είτε την g_shading() ανάλογα με την μεταβλητή shading ('f' ή 'g'). Η συναρτηση επιστρέφει την ανανεωμένη εικόνα με τα χρωματισμένα τρίγωνα img.

  Ακολουθεί ο ψευδοκώδικας :

```
Για κάθε τρίγωνο in faces:
   Υπολόγισε το βάθος του τριγώνου
   => newDepth = (depth[triangle[0]] + depth[triangle[1]] + depth[triangle[2]]) / 3

   Add το τρίγωνο, το χρώμα του και το βάθος του σε αντίστοιχες λίστες
   Stack τις λίστες των βαθών, των τριγώνων και των χρωμάτων των τριγώνων σε μία λίστα
   Sort με βάση το βάθος του τριγώνου με φθίνουσα σειρά

Για κάθε τρίγωνο in faces:
    Αν η μεταβλητή shading είναι 'f':
        img = f_shading (img, triangle, triangleColors [faces. index(triangle)])
    Αλλιώς, αν η μεταβλητή shading είναι 'g':
       img = g_shading (img, triangle, triangleColors [faces. index(triangle)])

Επιστροφή img
```

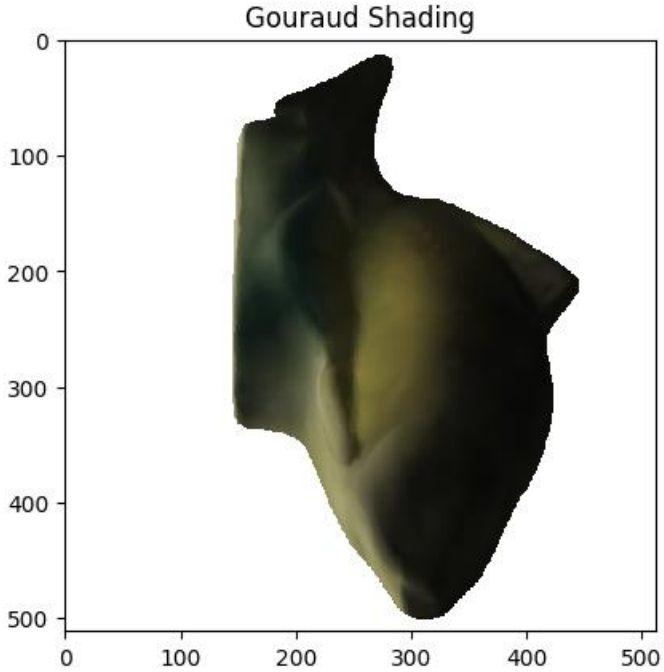
ΑΠΟΤΕΛΕΣΜΑΤΑ DEMOS

Flat Shading



Παρατηρείται ότι υπάρχει ενιαίος χρωματισμός των σημείων του κάθε τριγώνου με τον μέσο όρο χρώματος κορυφών και τα τρίγωνα είναι σχετικά διακριτά με το μάτι.

Gouraud Shading



Παρατηρείται ότι υπάρχει ομαλή συνέχεια στον χρωματισμό σημείων παρεμβολικά και τα τρίγωνα δεν είναι διακριτά.