Εργασία 1: Πλήρωση Τριγώνων

Πλευρίδη Βασιλική Βαρβάρα (ΑΕΜ:10454)

Απρίλιος 2024

1 Εισαγωγή

Στόχος της παρούσας εργασίας είναι η υλοποίηση αλγορίθμων που στοχεύουν στην πλήρωση και μετέπειτα στον κατάλληλο χρωματισμό τριγώνων. Μέσα από τις ζητούμενες συναρτήσεις (vector_interp, f_shading, g_shading,render_img) και με την βοήθεια των δύο scripts demo_g και demo_f γίνεται η παρουσίαση του ζητούμενου αντικειμένου, του οποίου τα δεδομένα εμπεριέχονται στο δοσμένο αρχείο hw1.py.

2 Συναρτήσεις και σχολιασμός υλοποίησης

Παρακάτω σχολιάζονται αναλυτικά οι ζητούμενες συναρτήσεις και πως αυτές υλοποιήθηκαν σε γλώσσα προγραματισμού python.

2.1 Συνάρτηση vector_interp

Η συγκεκριμένη συνάρτηση ουσιαστικά είναι μια υλοποίηση γραμμικής παρεμβολής. Η γραμμική παρεμβολή είναι μια μέθοδος που χρησιμοποιείται για να εκτιμήσει μια τιμή μεταξύ δύο γνωστών τιμών, χρησιμοποιώντας μια ευθεία γραμμή που συνδέει αυτά τα δύο σημεία. Στην περίπτωση αυτή, η γραμμική παρεμβολή εφαρμόζεται σε διανυσματικές τιμές (V1,V2) σε έναν πολυδιάστατο χώρο, υπολογίζοντας ένα ενδιάμεσο σημείο με την βοήθεια της συντεταγμένης coord καθορισμένη από την παράμετρο dim για το αν είναι τεταγμένη (1) ή τετμήμενη (2) του σημείου αυτού, μεταξύ δύο γνωστών σημείων (p1,p2) και αντιστοιχίζοντας την τιμή (έξοδος V) του διανύσματος σε αυτό το ενδιάμεσο σημείο.

2.2 Συνάρτηση f_shading

Η συνάρτηση αυτή αποτελεί έναν αλγόριθμο πλήρωσης τριγώνου με τον χρωματισμό του να καθορίζεται από τον διανυσματικό μέσο όρο των χρωμάτων των τριών κορυφών του. Παρακάτω παρατίθενται ένας ψευδοκώδικας για την ¨σκιαγράφιση' της λογικής της κύριας συνάρτησης,οι πρόσθετες συναρτήσεις και κλάσεις που χρησιμοποιήθηκαν για την απλοποίησή της, οι παραδοχές που χρησιμοποιήθηκαν καθώς και κάποια γενικότερα σχολία για την υλοποίηση του κώδικα:

```
1: function SHADING(image, vertices, vcolors)
2:     for k=0:1:2 do
3:         Βρίσκουμε τα y_min,y_max,x_min,x_max, slopes της κάθε ακμής k
4:     end for
5:
6:     ymin = min(y_min)
7:     ymax = max(y_max)
8:     Αρχικοποίηση του y σε y=ymin
9:     Δημιουργία αντικειμένων κλάσης για κάθε ακμή του τριγώνου
10:                                          ▷ Εύρεση των πρώτων ενεργών ακμών:
11:    for k=0:1:2 do
12:        if y_min[k]=y then
13:            Προσθήκη του k στις ενεργές ακμές
14:        else if  y_max[k]=y then
15:            Αφαίρεση του k από τις ενεργές ακμές
16:        end if
17:                                        ▷ Εύρεση των πρώτων ενεργών σημείων:
18:        for Όλα τα αντικείμενα των ακμών do
19:            if ανήκουν στις ενεργές ακμές then
20:                if  slope[k]<0 then
21:                    x=xmax του αντικειμένου
22:                end if
23:                if slope[k]>0 then
24:                    x=xmin του αντικειμένου
25:                end if
26:            end if
27:        end for
28:
29:        for y=ymin:1:ymax do
30:            Προσθήκη των τιμών x των σε έναν νέο πίνακα x_values
31:            Ταξινόμηση του πίνακα x_values
32:
33:            for i=1:2:2 do
34:                for x=x_values[1]:1:x_values[2] do
35:                    drawpixel(x,y)
36:                end for
37:            end for
38:            Κάλεσμα της active_boundary_points για ανανέωση των ενεργών
   σημείων και ακμών
39:        end for
40:
```

Ο παραπάνω ψευδοκώδικας απευθύνεται και στην περίπτωση του f_shading αλλά και σε αυτήν του g_shading.Η βοηθητική συνάρτηση active_boundary_points είναι επίσης κοινή. Για την υλοποίηση του shading, χρησιμοποιήθηκε η συνάρτηση slope,για τον υπολογισμό κλίσης της κάθε πλευράς.

Μία γενική δομή της λειτουργίας της συνάρτησης active_boundary_points είναι η εξης: Χωρίζεται σε 3 διαφορετικές εκδοχές, στην ύπαρξη νέας ενεργής ακμής και στην προσθήκη των νέων ενεργών σημείων, στην αφαίρεση κάποιας ακμής και σε όλες τις άλλες ακμές που δεν καλύπτονται από τις παραπάνω περιπτώσεις συνοδευομένη από την ανανέωση των ενεργών σημείων τους.

Τέλος δημιουργήθηκε η κλάση x_values_edges για την ευκολότερη διαχείρηση των δεδομένων με στόχο εν τέλει την απλοποίηση της υλοποίησης του κώδικα. Κάθε αντικειμένο αυτής της κλάσης, έχει τις εξής μέλη:

- **edge**: Η κωδικοποίηση με αριθμό (0,1,2) της πρώτης κορυφής της κάθε ακμής
- **slope**: Η κλίση της ακμής
- **xValue**: Η τρέχουσα τιμή x της ακμής
- **statement**: Ένδειξη που υποδεικνύει εάν η ακμή είναι ενεργή (1) ή όχι (0)
- **ymin**: Η ελάχιστη τιμή y που η ακμή τέμνει
- **ymax**: Η μέγιστη τιμή y που η ακμή τέμνει
- **xmin**: Η ελάχιστη τιμή x που η ακμή τέμνει
- **xmax**: Η μέγιστη τιμή x που η ακμή τέμνει

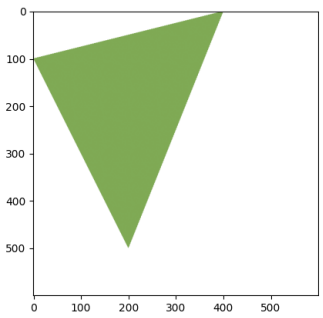
Η παραπάνω κλάση βοήθησε πολύ την απλοποίηση του κώδικα και ειδικά η τιμή της statement, η οποία διευκόλυνε την επιλογή των σωστών ενεργών σημείων. Επίσης, να σημειωθεί ότι επειδή επιλέχθηκε σαν παραδοχή η τιμή της edge να είναι η πρώτη κορυφή της ακμής, σε κάποια σημεία του κώδικα χρησιμοποιείται το $l = (k + 1) \% 3$ ώστε ανάλογα με το ανάλογο ζητούμενο, να επιλέγεται η άλλη κορυφή της ακμής. Ένα φανερό παράδειγμα αυτής της λογικής του κώδικα, βρίσκεται μέσα στην συνάρτηση active_boundary_points , στην περίπτωση νέας ενεργής ακμής, όπου υπάρχει ο απαιτούμενος έλεγχος και ο οποίος καθορίζει αν θα επιλαγει η τετμημένη της πρώτης (k) ή της δεύτερης κορυφής (l).

Οι **παραδοχές** που γίνονται στην υλοποίηση αυτή είναι ότι για τον χρωματισμό των pixel, θεωρήθηκε οτι τα pixel που συμπίπτουν με οριακά σημεία , δεν χρωματίζονται. Επίσης, οι οριζόντιες πλευρές των τριγώνων, εξαιρούνται από τις ενεργές ακμές.Γι΄ αυτό άλλωστε στον ορισμό των ενεργών ακμών, οι ακμές που η μεγιστη και η ελάχιστη τεταγμένη ταυτίζονται (δηλαδή είναι οριζόντιες), αφαιρούνται από την λίστα.

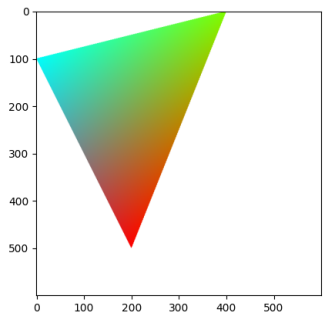
2.3 Συνάρτηση g_shading

Η συνάρτηση αυτή είναι σχεδόν ίδια με την f_shading με μόνη διαφορά να είναι αυτή του χρωματισμού των τριγώνων. Σε αυτή την συνάρτηση λοιπόν, χρησιμοποιήθηκε η βοηθητική συνάρτηση pixel_color, η οποία ουσιαστικά καλεί δύο φορές την συνάρτηση vector_interp,όπως ζητείται και από την εκφώνηση. Η μια φορά είναι για την εύρεση των χρωμάτων των συνευθεικών σημείων με του ζητούμενου σημείου κατά τον άξονα y και μία για τον εύρεση του χρώματος του τελικού σημείου. Να σημειωθεί ότι, σε περίπτωση που στο πρώτο στάδιο, τα σημεία p1 και p2 βρίσκονται στην ίδια ευθεία y, τότε, προς αποφυγή της διαιρέσεως με το 0 εσωτερικά της συνάρτησης vector_interp,χρησιμοποιούνται οι τετμημένες των σημείων αυτών (dim=1).

Για να συγκριθούν οι δύο διαφορετικές εκδοχές απόδοσης χρώματος, ορίστηκε ένα τρίγωνο με τυχαίες κορυφές [[0,100],[400,0],[200,500]],οι οποίες έχουν χρώμα [[0, 1, 1], [0.5, 1, 0],[1, 0, 0]] αντίστοιχα. Η διαφορά στον χρωματισμό, ανάλογα με την συνάρτηση που θα καλεστεί, φαίνεται παρακάτω:



Μέθοδος flat shading



Μέθοδος gouraud shading

2.4 Συνάρτηση render_img

Στόχος αυτής της τελευταίας συνάρτησης είναι η δυνατότητα σχεδιασμού και χρήσης κατάλληλων χρωμάτων πολύπλοκων εικόνων έχοντας ως είσοδο τα απαραίτητα στοιχεία για τα επιμέρους τρίγωνα από τα οποία αποτελείται. Πρώτο στάδιο λοιπόν του αλγορίθμου είναι η δημιουργία μιας εικόνας με τις κατάλληλες απαιτήσεις, στην οποία θα προστεθούν σταδιακά και τα τρίγωνα. Στην συνέχεια,γίνεται ο υπολογισμός του βάθους για κάθε τρίγωνο με βάση το κέντρου βάρους των κορυφών του κάθε τριγώνου και στην συνέχεια η κατά φθίνουσα ταξινόμηση τους με κριτήριο το παραπάνω. Το τελικό στάδιο ειναι η σχεδίαση και ο χρωματισμός των τριγώνων, ανάλογα βέβαια με το ποιά συνάρτηση χρωματισμού θα επιλεχθεί.Να σημειωθεί ότι υποχρεωτικά θα πρέπει πρώτα να γίνει η ταξινόμηση των τριγώνων, με την βοήθεια του πίνακα faces, γιατί είναι αυτός που έχει αναφορά και στους υπόλοιπους πίνακες vertices και vcolors.

3 Scripts και αποτελέσματα

Για την επαλίθευση της σωστής λειτουργίας των κωδίκων, δημιουργήθηκαν τα αρχεία demo_f.py και demo_g.py, στα οποία αφού γίνει η φόρτωση των απαιτούμενων δεδομένων από το αρχείο hw1.npy, καλούν την συνάρτηση render_img με shading=f και g αντίστοιχα και τέλος αποθηκεύουν την ζητούμενη εικόνα. Τα τελικά αποτελέσματα παρουσιάζονται παρακάτω:

Μέθοδος flat shading

Μέθοδος gouraud shading