



Γραφική με Υπολογιστές
-Εργασία 1-
Πλήρωση Τριγώνων

Αλέξανδρος Πετρίδης

Τελευταία ενημέρωση: 26 Μαρτίου 2021

Περιεχόμενα

1	Συνάρτηση Γραμμικής Παρεμβολής	3
2	Συναρτήσεις Πλήρωσης Τριγώνων	3
3	Συνάρτηση χρωματισμού αντικειμένου	4
4	Αποτέλεσμα χρωματισμένου αντικειμένου <i>flat</i>	5
5	Αποτέλεσμα χρωματισμένου αντικειμένου <i>gouraud</i>	6

1 Συνάρτηση Γραμμικής Παρεμβολής

Η συνάρτηση της γραμμικής παρεμβολής δέχεται σαν ορίσματα τα χρώματα και τις συντεταγμένες δύο σημείων αλλά και την κατεύθυνση με την οποία θέλουμε να κάνουμε γραμμική παρεμβολή. Επιστρέφει τον πίνακα χρωμάτων του σημείου που έχουμε ζητήσει.

2 Συναρτήσεις Πλήρωσης Τριγώνων

Οι συναρτήσεις πλήρωσης τριγώνων χρωματίζουν το δοσμένο τρίγωνο ανάλογα με τα χρώματα των κορυφών του. Για την συνάρτηση *flat* το χρώμα υπολογίζεται για όλα τα σημεία του τριγώνου από την αρχή και είναι το ίδιο, ενώ για την *gourad* κάθε φορά πριν βαφτεί οποιοδήποτε σημείο υπολογίζουμε το χρώμα του, το οποίο διαφέρει από σημείο σε σημείο και είναι ανάλογο των χρωμάτων των κορυφών του.

Αλγόριθμος Πλήρωσης Τριγώνων

```

1      %Pseudocode here
2      for i=1:3
3          find ykmin, ykmax, xkmin, xkmax, dx, dy, slope and sign for each side.
4      end
5      find ymin, ymax
6      find active_edges, active_points
7          % paint first yscan
8      for x = active_points(1):active_points(2)
9          drawpixel(x,y)
10     end
11     find next active_points
12     for y=ymin+1:ymax-1
13         for x = active_points(1):active_points(2)
14             drawpixel(x,y)
15         end
16         find active_edges
17         retrospectively find active_points
18     end
19     % paint last yscan
20     for active_points(1):active_points(2)
21         drawpixel(x,y)
22     end

```

Αλγόριθμος εύρεσης Ενεργών πλευρών

Η λογική πίσω από τον αλγόριθμο εύρεσης ενεργών πλευρών είναι πως όταν το y από το σκανάρισμα μου θα συναντήσει σημείο το οποίο θα είναι ταυτόχρονα μέσα στο τρίγωνο αλλά και μέσα στους πίνακες $ykmin, ykmax$ που περιέχει τα ελάχιστα και τα μέγιστα y από όλες τις πλευρές, τότε θα αλλάζει την πλευρά που σχετίζεται με το σημείο αυτό και θα βάλει την επόμενη.

```

1      %Pseudocode here
2      if y in ykmin && y in ykmax
3          find which point has this y (pn)
4          if pn in active_sides
5              active_side(active_side == pn) = mod(pn+1,3) + 1
6              % with (mod(pn+1,3) + 1) we have for 1 -> 3, 2 -> 1, 3 -> 2
7              if active_side(1) == pn
8                  active_side(1) = mod(pn+1,3) + 1
9              else
10                 active_side(2) = mod(pn+1,3) + 1
11             end
12         else
13             active_side(active_side == mod(pn+1,3) + 1) = pn
14             if active_side(1) == mod(pn+1,3) + 1
15                 active_side(1) = pn
16             else
17                 active_side(2) = pn
18             end
19         end
20     end

```

Αναδρομικός αλγόριθμος εύρεσης Ενεργών σημείων

Στον αναδρομικό αλγόριθμο εύρεσης ενεργών σημείων θα βρίσκω κάθε επόμενο x από το προηγούμενό του με την βοήθεια του αλγορίθμου του *Bresenham*. Έχοντας ήδη βρεί το *sign* της κάθε μίας κλήσης. Το *sign* δίνει -1 για αρνητικές τιμές 0 για μηδενικές και +1 για θετικές και μας βοηθάει να διαλέξουμε τι θα προσθέσουμε κάθε φορά στο προηγούμενο x για να βγάλουμε το επόμενο. Έχω ακόμα εκφράσει τα δύο ενεργά σημεία που θα έχουμε κάθε φορά σαν *structs* τα οποία έχουν τις τιμές x και *sideFrom*.

```

1      %Pseudocode here
2      for 2 active_points and for their sides
3          if sign(side) == 0
4              stay same x
5          else if dx(side) != 0
6              make sum of current side sum(side) += dx(side)
7              while sum(side) >= dy(side)
8                  if dy(side) != 0
9                      x += sign(side)
10                     sum(side) -= dy(side)
11             end
12         end
13     end

```

3 Συνάρτηση χρωματισμού αντικειμένου

Η συνάρτηση χρωματισμού αντικειμένου δημιουργεί την προβολή ενός αντικειμένου το οποίο αποδομείται από πολλά μικρά τρίγωνα τα οποία χρωματίζονται μέσα από τις προηγούμενες συναρτήσεις. Η σειρά του χρωματισμού εξαρτάται από τις τιμές του βάθους των τριγώνων που απαρτίζουν τα σχήμα. Ακόμα η συνάρτηση σου δίνει την επιλογή χρωματισμού *flat* ή *gouraud*.

```

1      %Pseudocode here
2      Create image with sides MxN ,M = 1200 & N = 1200
3      for i = 1:K % K is number of triangles
4          triangle_depth = mean(depth(tops of triangle))
5      end
6      sort triangles by depth
7      if render == 'Flat'
8          for k = 1:K
9              Create Image with paint_triangle_flat for all triangles
10         end
11     else if render == 'Gouraud'
12         for k = 1:K
13             Create Image with paint_triangle_gouraud for all triangles
14         end
15     end

```

4 Αποτέλεσμα χρωματισμένου αντικειμένου *flat*

Παρακάτω βλέπουμε το αποτέλεσμα της συνάρτησης `demo_flat` η οποία καλείται χωρίς εξωτερικά ορίσματα, φορτώνοντας τα δεδομένα εισόδου που δίνονται και χρησιμοποιώντας την συνάρτηση χρωματισμού αντικειμένου πραγματοποιεί τον χρωματισμό των τριγώνων με *flat* τρόπο. Τέλος αποθηκεύει την τελική εικόνα στο αρχείο `demo_flat.png`. Αναμενόμενος χρόνος εκτέλεσης περίπου 9 λεπτά.



5 Αποτέλεσμα χρωματισμένου αντικειμένου *gouraud*

Παρακάτω βλέπουμε το αποτέλεσμα της συνάρτησης `demo_gouraud` η οποία καλείται χωρίς εξωτερικά ορίσματα, φορτώνοντας τα δεδομένα εισόδου που δίνονται και χρησιμοποιώντας την συνάρτηση χρωματισμού αντικειμένου πραγματοποιεί τον χρωματισμό των τριγώνων με *gouraud* τρόπο. Τέλος αποθηκεύει την τελική εικόνα στο αρχείο `demo_gouraud.png`. Αναμενόμενος χρόνος εκτέλεσης περίπου 9 λεπτά.

