Αναφορά

Γραφική με Υπολογιστές 2021
Εργασία #1: Πλήρωση Τριγώνων
Τσούσης Παναγιώτης
ΑΕΜ: 9590

Ενδεικτικά αποτελέσματα

Γενικές πληροφορίες:

Για την παραγωγή μιας εικόνας png, τόσο το flat rendering όσο και το gouraud rendering χρειάζονται λιγότερο από 10 δευτερόλεπτα, με επεξεργαστή Intel Core i7-7700 2.80GHz. Το πρόγραμμα έτρεξε σε Windows 10 και η έκδοση του Matlab στο οποίο αναπτύχθηκε είναι R2018a.

Οι 2 εικόνες παράγονται με την απλή εκτέλεση των 2 demo.

Flat rendering:(*demo_flat.m*)

Gouraud rendering:(*demo_gouraud.m*)

Παραδοχές

Οι παραδοχές που έγιναν είναι οι εξής:

- Η αρχή των αξόνων για την τοποθέτηση των pixels είναι η (1,1) και βρίσκεται πάνω αριστερά.
- Οι συντεταγμένες ενός pixel αναφέρονται στο κέντρο ενός τετραγώνου, με μέγεθος πλευράς 1.
- Αν δύο κορυφές εκφυλίζονται σε μια, το χρώμα που θα έχει η κορυφή είναι ο μέσος όρος των δύο.

Περιγραφή Συναρτήσεων

Στο παραδοτέο υπάρχουν οι εξής 6 συναρτήσεις Matlab:

- *vector_interp*(*p*1, *p*2, *a*, *V*1, *V*2, *dim*)
- *paint_triangle_flat*(*img, vertices_2d, vertex_colors*)
- *paint_triangle_gouraud*(*img, vertices_2d, vertex_color*)
- *render*(*vertices_2d, faces, vertex_colors, depth, renderer*)
- *bresenham*(*vertices_2d*)
- *sort_faces_by_depth(faces, depth)*

καθώς και 2 scripts επίδειξης:

- *demo_flat.m*
- *demo_gouraud.m*

Όλες οι συναρτήσεις είναι καλά σχολιασμένες στα αγγλικά, σε περίπτωση που χρειαστεί περαιτέρω διερεύνηση ο κώδικας.

vector_interp(p1, p2, a, V1, V2, dim) :

• *p*1 και *p*2 οι δισδιάστατες συντεταγμένες δύο κορυφών ενός τριγώνου.
• *V*1 και *V*2 οι τρισδιάστατες τιμές που αντιστοιχούν στις κορυφές *p*1 και *p*2.
• *a* το σημείο στο οποίο θα εφαρμοστεί η παρεμβολή.
• *dim* ∈ *{*1, 2*}* είναι η κατεύθυνση κατά την οποία θα πραγματοποιηθεί η γραμμική παρεμβολή (δηλαδή οριζοντίως ή καθέτως).
• *value* είναι η τιμή που προκύπτει από γραμμική παρεμβολή των *V*1 και *V*2 κατά τη κατεύθυνση *dim*.

Περιγραφή λειτουργίας:

Αν τα p1 και p2 είναι ταυτόσημα, το χρώμα που επιστρέφει η συνάρτηση υπολογίζεται ως value = 0.5*V1 + 0.5*V2

Διαφορετικά, το χρώμα είναι V1*per - V2*(1-per) , όπου per είναι το ποσοστό |a-p1| / |p2-p1|.

bresenham(vertices_2d) :

- vertices_2d : πίνακας 2x2 με τις συντεταγμένες 2 σημείων

Περιγραφή λειτουργίας:

Εφαρμόζει τον αλγόριθμο του bresenham ακριβώς όπως περιγράφεται στις σημειώσεις, με τη διαφορά ότι κάνει τους κατάλληλους μετασχηματισμούς (x = -x, y=x κλπ.) πριν την εφαρμογή (σχολιασμένα στον κώδικα) ώστε να δουλεύει σε κάθε περίπτωση κλίσης ευθείας.

Επιστρέφει ένα πίνακα Nx2 με όλα τα pixels που ανήκουν στον πίνακα.

sort_faces_by_depth(faces, depth) :

- faces : ο πίνακας Kx3 που περιέχει τις κορυφές των K τριγώνων που πρόκειται να χρωματιστούν.
- depth : ο πίνακας Lx1 που δηλώνει το βάθος κάθε κορυφής.

Περιγραφή λειτουργίας:

Η συνάρτηση αυτή επιστρέφει τον πίνακα faces, αφού πρώτα τον κάνει sort με βάση τον πίνακα depth.

Αρχικά, δημιουργεί έναν πίνακα (faces_depth) στον οποίο αποθηκεύται το βάθος κάθε τριγώνου ως ο μέσος όρος του βάθους των κορυφών του. Στην συνέχεια, γίνεται sorting του faces_depth από το μεγαλύτερο στο μικρότερο, κρατώντας σε ένα δεύτερο πίνακα I τους αλλαγμένους δείκτες του. Τέλος, αλλάζει τη θέση των στοιχείων του πίνακα faces με βάση τους δείκτες του πίνακα I και τον επιστρέφει.

paint_triangle_*flat*(img, vertices_2d, vertex_colors)

- img: εικόνα (πίνακας διάστασης $M \times N \times 3$) με τυχόν προϋπάρχοντα τρίγωνα.
- vertices_2d*:* ακέραιος πίνακας διάστασης 3×2 που σε κάθε γραμμή περιέχει τις δισδιάστατες συντεταγμένες μιας κορυφής του τριγώνου.
- vertex_colors*:* πίνακας διάστασης 3×3 που σε κάθε γραμμή περιέχει το χρώμα μιας κορυφής του τριγώνου σε μορφή RGB (με τιμές στο διάστημα [0, 1]).

Περιγραφή λειτουργίας:

Η συνάρτηση αυτή δέχεται ως όρισμα μια εικόνα img και τις κορυφές ενός τριγώνου (vertices_2d) μαζί με τα χρώματά τους (vertex_colors) και παράγει την εικόνα img, αλλά με το τρίγωνο εισόδου χρωματισμένο μονοχρωμικά. Το χρώμα με το οποίο χρωματίζεται το τρίγωνο είναι ο μέσος όρος που προκύπτει από τα χρώματα των κορυφών.

Ψευδοκώδικας:

```
//Find the flat color

color = 1/3*sum(vertex_colors);

//Find pixels that belong to the sides

side_pixels = <Apply bresenham to find all the pixels that belong to any
               side>;

//Paint every pixel that belongs to a side

for i=1:length(side_pixels)
     img(<side_pixels(i) coordinates>) = color;
end


//Find min and max of y for scanlines

ymin = min(vertices_2d(:,2));
ymax = max(vertices_2d(:,2));

//Paint pixels that don't belong in scanline

for i=ymin:ymax
     //Find the x of every pixel with y = i of scanline
        rows = <Find every pixel in side_pixels with y=i>
     //Keep only the x value and not the y
        scanline = <Keep only the x value of rows>
     //Sort the scanline array
        scanline = <sort scanline>
     //From lowest to highest value of x for the current scanline,
paint every pixel not in scanline
        n = length(cur_scan_x);
        for j = scanline(1):scanline(n)
            if <j not in scanline>
                img(< y=i , x=j >)=color;
            end
        end
end

    result = img;
```

paint_triangle_*gouraud*(img, vertices_2d, vertex_colors)

- img: εικόνα (πίνακας διάστασης $M \times N \times 3$) με τυχόν προϋπάρχοντα τρίγωνα.
- vertices_2d*:* ακέραιος πίνακας διάστασης 3×2 που σε κάθε γραμμή περιέχει τις δισδιάστατες συντεταγμένες μιας κορυφής του τριγώνου.
- vertex_colors*:* πίνακας διάστασης 3×3 που σε κάθε γραμμή περιέχει το χρώμα μιας κορυφής του τριγώνου σε μορφή RGB (με τιμές στο διάστημα [0, 1]).

Περιγραφή λειτουργίας:

Η συνάρτηση αυτή δέχεται ως όρισμα μια εικόνα img και τις κορυφές ενός τριγώνου (vertices_2d) μαζί με τα χρώματά τους (vertex_colors) και παράγει την εικόνα img, αλλά με το τρίγωνο εισόδου χρωματισμένο. Το χρώμα με το οποίο χρωματίζεται το κάθε pixel του τριγώνου προκύπει με γραμμική παρεμβολή είτε για την πλευρά ενός τριγώνου ή για το scanline.

Ψευδοκώδικας:

```
//Find all pixels that belong to the sides
side1 = <bresenham for p1 p2>;
side2 = <bresenham for p2 p3>;
side3 = <bresenham for p1 p3>;

//Get every color for every vertex
color1 = vertex_colors(1,:);
color2 = vertex_colors(2,:);
color3 = vertex_colors(3,:);

//Store all the pixels that belong to the sides in one array
side_pixels = [side1;side2;side3];


//'Fix' vertex colors if 3 points are the same.
if <p1 == p2 == p3>
    color_fix = (0.33)*color1 +(0.33)*color2+(0.33)*color3;
    color1 = color_fix;
    color2 = color_fix;
    color3 = color_fix;
else
    //'Fix' vertex colors if 2 points are the same.
    if <p1 == p2>
        color_fix = 0.5*color1 + 0.5*color2;
        color1=color_fix;
```

```
            color2=color_fix;
        end

        if <p2 == p3>
            color_fix = 0.5*color2 + 0.5*color3;
            color2=color_fix;
            color3=color_fix;
        end

        if <p1 == p3>
            color_fix = 0.5*color1 + 0.5*color3;
            color1=color_fix;
            color3=color_fix;
        end
    end

    //'Paint' the pixels of the sides before we start. In this case,
with linear interpolation

    for <every side>

         <find right dim>

           for <every pixel of the side>
               pixel_color = <linear interpolation for the pixel>
               img(<pixel coordinates>) = pixel_color
           end
    end


    //Find scanline boundaries

    ymin = min(vertices_2d(:,2));
    ymax = max(vertices_2d(:,2));

   //Scan every line and paint

    for i=ymin:ymax
        //~~~Find every x coordinate of the side pixels that belong
to the scanline~~~
        scanline = <every x in y=i>
        //Sort so that highest values are last
        scanline = <sort scanline>;
        //~~~Find the inner active points of the scanline~~~
        n = length(scanline);
```

```
        //Flag that we are in the triangle
        in = 0;
        //Flag that we have null filling space (1 or 2 pixels or
horizontal line)
        empty = 1;
        //Inner point of active line 1
        start_x = -1;
        //Inner point of active line 2
        end_x = -1;
        //Check every x in range of the scan
        for j = cur_scan_x(1):cur_scan_x(n)
            //If current x is not in scanline
            %Find start
            if <current x not in scan> && <in==0, we are not in>
                //save the x of the inner active point 1
                start_x = j-1;
                //Flag that we are inside the triangle
                in = 1;
                //Flag that filling space is not null
                empty = 0;
            end
            %Find end
            if <current x is in scan> && (in == 1, we are in)
                //Save the x of the inner active point 2
                end_x = j;
                //Flag that we are not in the triangle
                in = 0;
            end

        end

        //~~~Color the filling space~~~

        if empty == 0 %If the filling space is not null
            for k = (start_x+1):(end_x-1)
                color1 = img(<x=start_x , y = i>);
                color2 = img(<x=end_x , y = i>);
                color = <interpolate for k between start_x and end_x>
                img(<x=k , y=i>)= color;
            end
        end
    end

    result = img;
```

render(vertices_2d, faces, vertex_colors, depth, renderer)

- Img: έγχρωμη εικόνα διάστασης M×N×3. Η εικόνα θα περιέχει K χρωματισμένα τρίγωνα τα οποία σχηματίζουν την προβολή ενός 3D αντικειμένου στις 2 διαστάσεις.
- vertices_2d : πίνακας με τις κορυφές των τριγώνων της εικόνας. Ο πίνακας vertices_2d είναι διάστασης L×2 και περιέχει τις συντεταγμένες ενός πλήθους L κορυφών. Για απλούστευση υποθέστε ότι όλες βρίσκονται εντός του καμβά.
- faces: πίνακας που περιέχει τις κορυφές των K τριγώνων. Ο πίνακας είναι διάστασης K × 3. Η i-στη γραμμή του πίνακα δηλώνει τις τρεις κορυφές που σχηματίζουν το τρίγωνο (με αναφορά σε κορυφές του πίνακα vertices και αρίθμηση που ξεκινά από το 1).
- vertex_colors : πίνακας με τα χρώματα των κορυφών. Ο πίνακας C είναι διάστασης L×3. Η i-στη γραμμή του πίνακα δηλώνει τις χρωματικές συνιστώσες της αντίστοιχης κορυφής.
- depth: πίνακας που δηλώνει το βάθος της κάθε κορυφής πριν την προβολή του αντικειμένου στις 2 διαστάσεις. Ο πίνακας depth είναι διάστασης L × 1
- renderer: είναι μια μεταβλητή ελέγχου (‘flat’ ή ‘gouraud’) που καθορίζει τη συνάρτηση χρωματισμού που θα χρησιμοποιηθεί.
- M και N είναι το ύψος και το πλάτος του καμβά αντίστοιχα.

Περιγραφή λειτουργίας:

Η συνάρτηση αυτή δέχεται ως όρισμα πλέον όλους τους πίνακες που δίνονται ως δεδομένα και επιπλέον την επιλογή renderer που μπορεί να είναι ‘flat’ ή ‘gouraud’. Ανάλογα με την επιλογή, καλεί επαναληπτικά για όλα τα τρίγωνα τη συνάρτηση paint_triangle_flat ή paint_triangle_gouraud και παράγει ως έξοδο ένα πίνακα img, που έχει όλα τα τρίγωνα χρωματισμένα. Αν στον img εφαρμόσουμε τη συνάρτηση imwrite, παράγει μια εικόνα png του δωσμένου αντικειμένου.

ΠΡΟΓΡΑΜΜΑ demo_flat

Περιγραφή λειτουργίας:

Το πρόγραμμα αυτό, όταν τρέξει στο Matlab μαζί με τις συναρτήσεις του, παράγει μια εικόνα png του ρακούν στα αρχεία, με flat rendering.

ΠΡΟΓΡΑΜΜΑ demo_gouraud

Περιγραφή λειτουργίας:

Το πρόγραμμα αυτό, όταν τρέξει στο Matlab μαζί με τις συναρτήσεις του, παράγει μια εικόνα png του ρακούν στα αρχεία, με gouraud rendering.

Διάγραμμα Κλήσεων