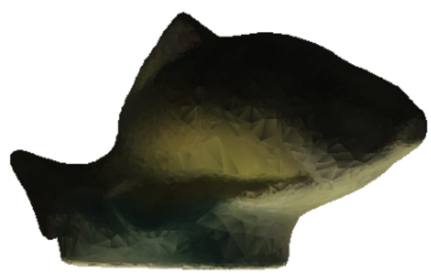
1η εργασία

Πλήρωση Τριγώνων



Δημήτριος Σουρλαντζής

Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Author Note

Αυτή η αναφορά αφορά την πρώτη εργασία για το μάθημα Γραφική με Υπολογιστές που διδάσκεται στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, του Αριστοτέλειου Πανεπιστημίου Θεσσαλονίκης. Το παρόν έγγραφο δεν αποτελεί κάποια επιστημονική έρευνα, αλλά την υλοποίηση διαφόρων αλγορίθμων που ζητήθηκαν για την εργασία. Για οποιαδήποτε απορία στείλτε μήνυμα στην ηλεκτρονική διεύθυνση sourland@ece.auth.gr

Contents

1 Εισαγωγή

Σε αυτήν την εργασία πρέπει να υλοποιηθεί ο αλγόριθμος πλήρωσης τριγώνων με χρήση γραμμών σάρωσης. Στην παρούσα αναφορά παρουσιάζονται δύο διαφορετικές υλοποιήσεις, μία με την χρήση του κλασικού αλγορίθμου γραμμών σάρωσης, και μία πιο βελτιωμένη που χρησιμοποιεί τις γραμμές σάρωσης σε συνδυασμό με τα κέντρα βάρους των τριγώνων.

2 Η Πλήρωση Γενικών Πολυγώνων

Η πλήρωση πολυγώνων είναι από τα τελευταία βήματα που γίνονται στο pipeline του συστήματος γραφικών. Πριν αρχίσει ο χρωματισμός, έχει ολοκληρωθεί ήδη η μοντελοποίηση του κόσμου και η προβολή τρισδιάστατων στοιχείων του κόσμου πάνω στον καμβά δύο διαστάσεων. Η παραλλαγή του αλγορίθμου πλήρωσης πολυγώνων σε αλγόριθμο πλήρωσης τριγώνων είναι πολύ συνηθισμένη επιλογή, καθώς τα τρίγωνα είναι πιο εύκολο να χρωματιστούν. Επιπλέον, διαμερίζοντας κάθε πολύγονο σε επιμέρους τρίγωνα, μπορούμε με τεχνικές παραλληλοποίησης να χρωματίσουμε πολύγωνα πολύ γρήγορα. Είναι σημαντικό να σημειωθεί πως η παραλληλοποίηση είναι προς το παρόν εκτός των πλαισίων του μαθήματος *Γραφική με Υπολογιστές* οπότε δεν έγινε κάποια απόπειρα παραλληλοποίησης

3 Ο κλασικός Αλγόριθμος Scanline

3.1 Οι εξαιρέσεις στον Αλγόριθμο Scanline

Πριν περάσουμε στον ψευδοκώδικα του αλγορίθμου, αξίζουν να σημειωθούν τα είδη των τριγώνων που αντιμετωπίζουμε.

- **Τρίγωνα στα οποία οι κορυφές τους συγχωνεύονται στο ίδιο σημείο**: Σε αυτήν την περίπτωση απλά χρωματίζουμε τον καμβά μας στο σημείο των κορυφών. Το χρώμα που βάφουμε είναι ο μέσος όρος του χρώματος των κορυφών και στην λειτουργεία *flat* και στην λειτουργία *Gouraud*, αφού στην δεύτερη περίπτωση αφού μιλάμε για το ίδιο σημείο στον καμβά, δεν έχει νόημα να εφαρμόσουμε γραμμική παρεμβολή.

- **Τρίγωνα στα οποία δύο στις τρεις κορυφές τους συγχωνεύονται στο ίδιο σημείο**: Σε αυτήν την περίπτωση έχουμε να χρωματίσουμε ένα ευθύγραμμο τμήμα. Μπορούμε να εφαρμόσουμε τον αλγόριθμο του Bresenham, αλλά παρατηρήθηκε στην τελική εικόνα πως το τρίγωνο είναι αόρατο, οπότε μπορούμε να παραλείψουμε τον χρωματισμό τέτοιων τριγώνων.

- **Τρίγωνα με οριζόντια ακμή στην αρχή**: Χρωματίζουμε την οριζόντια ακμή μόνη της και αρχίζουμε την διαδικασία scanline από το επόμενο y.

- **Τρίγωνα με οριζόντια ακμή στο τέλος**: Η οριζόντια ακμή θα χρωματιστεί αυτόματα από τον αλγόριθμο αφού θα είναι το τελευταίο βήμα του αλγορίθμου και τα ενεργά pixels θα υπολογιστούν απο το προηγούμενο βήμα.

- **Κανονικό τρίγωνο**: Σε αυτήν την περίπτωση ο αλγόριθμος δεν θα συναντήσει κανένα εμπόδιο

3.2 Υλοποίηση του αλγορίθμου Scanline

Η βασική λογική του αλγορίθμου είναι η εξής:

- Εξετάζουμε αν το τρίγωνο ανήκει σε κάποιες από τις εξαιρέσεις 1 ή 2 που αναφέραμε στην ενότητα 3.1 για να δούμε αν αξίζει να χρωματιστεί.
- Αν γίνεται να χρωματιστεί, βρίσκουμε τα ολικά μέγιστα και ελάχιστα των x και y ανάμεσα στις κορυφές του τριγώνου ε.ω. να το φράξουμε σε ένα τετράπλευρο όπως στο σχήμα 1.

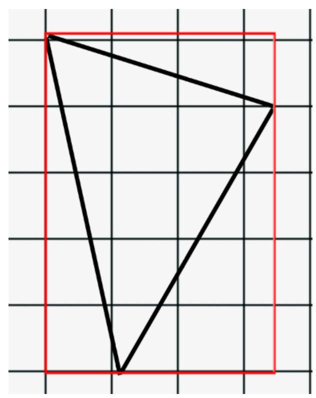


Figure 1: Κάθε τρίγωνο περιβάλλεται από ένα παραλληλόγραμμο

- Βρίσκουμε τις ενεργές ακμές και κορυφές του τριγώνου: Αν το τρίγωνο ξεκινάει με οριζόντια πλευρά, τότε οι ενεργές ακμές είναι οι κορυφές που σχηματίζουν την οριζόντια πλευρά και οι ενεργές πλευρές είναι οι υπόλοιπες 2. Χρωματίζεται η οριζόντια ακμή μόνη της και μετά ο αλγόριθμος συνεχίζει κανονικά. Σε κάθε άλλη περίπτωση, οι ενεργές πλευρές είναι αυτές που έχουν κορυφή στο y_{min} και οι ενεργές κορυφές x_1, x_2 είναι, εξίσου, η κοινή κορυφή των 2 ενεργών ακμών.
- Από εκεί και πέρα:
  - Τρέχουμε σε επανάληψη για κάποιο y από y_{min} εώς y_{max}.

– Ενημερώνουμε τα ενεργά σημεία με βάση τις κλήσεις των ενεργών ακμων.

– Ζωγραφίζουμε τα pixels στο ύψος y απο x_1 εώς x_2

– Ελέγχουμε μήπως πρέπει να αλλάξουν οι ενεργές πλευρές και δράτουμε αναλόγως.

Ο αλγόριθμος σε αναλυτικά βήματα είναι:

Algorithm 1 Χρωματισμός Τριγώνου

```
procedure COLOR TRIANGLE(Image, 2DVertices, VerticeColors, ShadeMode)
    Δημιουργία της κλάσης Ακμή
    ΑΡΧΙΚΟΠΟΙΗΣΗ ΑΚΜΗΣ(Όνομα, Αρχή, Τέλος, Χρώμα, Κλίση, Ενεργή)
    if Any Slope = NaN then
        Stop
    end if
    Βρές μέγιστα και ελάχιστα x,y
    Υπολόγισε τις ενεργές ακμές
    if Horizontal = True then
        x1 ← edge.start
        x2 ← edge.end
        Χρωμάτισε Οριζόντια Ακμή
    else
        Βρες ενεργή κορυφή x_1
        x_2 ← x_1
        Χρωμάτισε την κορυφή
    end if
    for y_min ≤ y ≤ y_max do
        Ενημέρωση x_1, x_2

        Color_A ← ΓΡΑΜΜΙΚΗ ΠΑΡΕΜΒΟΛΗ ΧΡΩΜΑΤΟΣ(Edge1_{y_min}, Edge1_{y_max}, y, Edge_1Colors)
        Color_B ← ΓΡΑΜΜΙΚΗ ΠΑΡΕΜΒΟΛΗ ΧΡΩΜΑΤΟΣ(Edge2_{y_min}, Edge2_{y_max}, y, Edge_2Colors)
        for x_1 ≤ x ≤ x_2 do
            if thenShadeMode = flat
                Color ← mean(VerticeColors)
            else
                Color ← ΓΡΑΜΜΙΚΗ ΠΑΡΕΜΒΟΛΗ ΧΡΩΜΑΤΟΣ(x_1, x_2, x, Color_A, Color_B)
                ΧΡΩΜΑΤΙΣΜΟΣ PIXEL(x,y, Color)
            end if
        end for
        Ενημέρωση ενεργών ακμών
    end for
```

Η γραμμική παρεμβολή χρώματος, για τον χρωματισμό με μέθοδο Gouraud γίνεται ως εξής:

Algorithm 2 Χρωματισμός Τριγώνου

procedure Γραμμικη παρεμβολη$(x_1, x_2, x, Color_1.Color_2)$
  $t \leftarrow \frac{x-x_1}{x_2-x_1}$
  $FinalColor \leftarrow |Color_1 + t \times (Color_2 - Color_1)|$
  Επιστροφή FinalColor

Τελικά το αντικείμενο που χρωματίζεται είναι το παρακάτω:

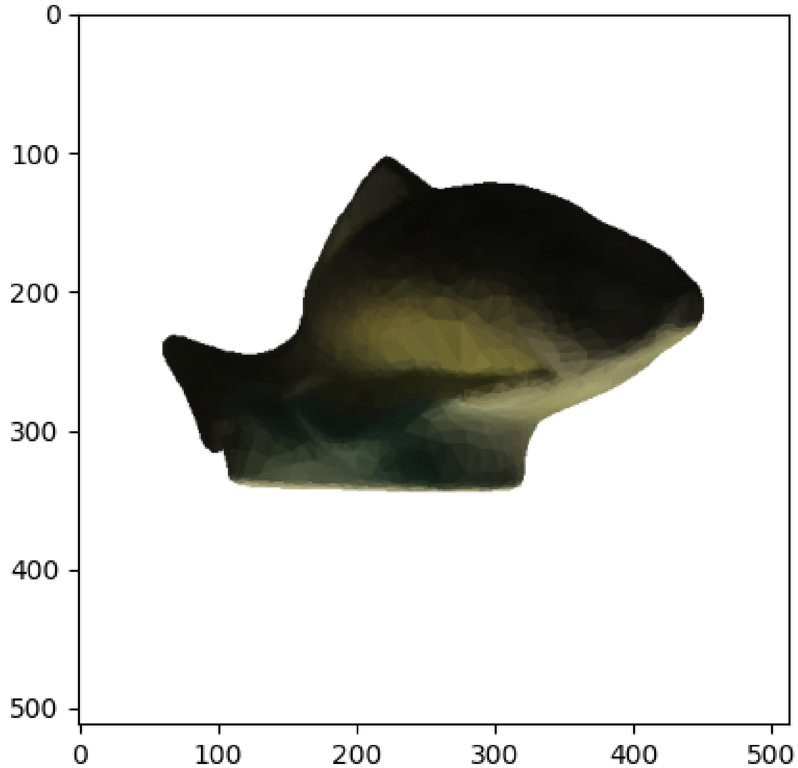


Figure 2: Το αντικείμενο χρωματισμένο με τρόπο flat.

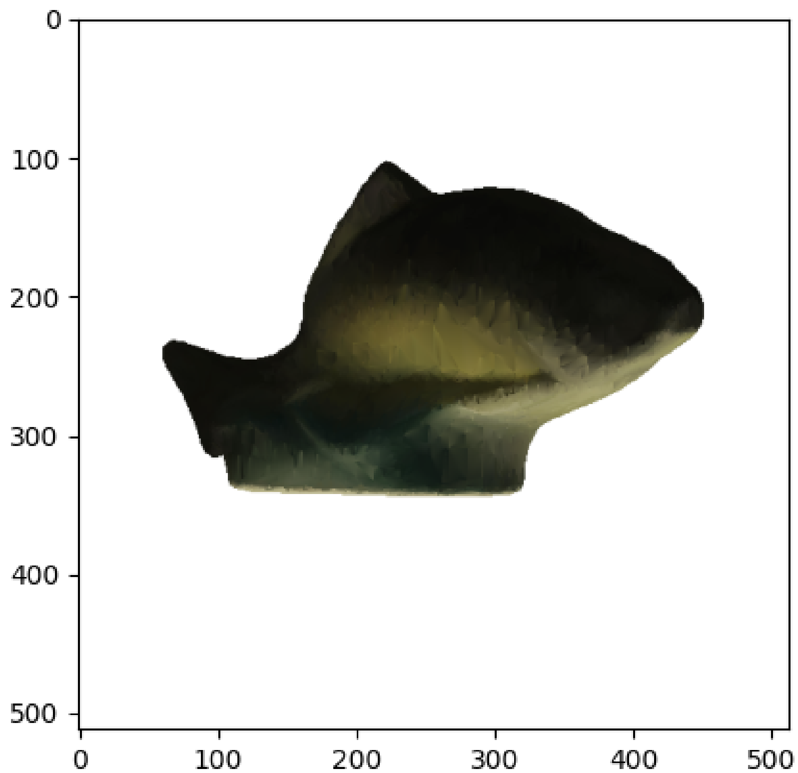


Figure 3: Το αντικείμενο χρωματισμένο με τρόπο Gouraud

4 Ο Αλγόριθμος Scanline με χρήση βαρών

Μπορούμε να αλλάξουμε λίγο την παραπάνω μέθοδο, και να την κάνουμε πιο εύρωστη ως προς σφάλματα στρογγυλοποίησης. Αντί να βρούμε ενεργά σημεία και ενεργές ακμές, μπορούμε να μετασχηματίσουμε τις συντεταγμένες κάθε σημείου του τετραπλεύρου του σχήματος 1, έτσι ώστε κάθε συντεταγμένη στο τετράπλευρο να δηλώνει που βρίσκεται το pixel σε σχέση με τις κορυφές του τριγώνου. Μετασχηματίζουμε τις συντεταγμένες (x,y) του pixel στις συντεταγμένες (a,b,c) έ.ω.:

$$a = \frac{f_{12}(x, y)}{f_{12}(x_1, y_1} \tag{1}$$

$$b = \frac{f_{02}(x, y)}{f_{02}(x_2, y_2)} \tag{2}$$

$$c = \frac{f_{12}(x, y)}{f_{12}(x_3, y_3)} \tag{3}$$

Όπου

$$f_{ab}(x, y) = (y_a - y_b)x + (x_b - x_a)y + x_a y_b + x_b y_a \tag{4}$$

Ο έλεγχος για το αν πρέπει να χρωματιστεί ένα pixel η όχι εξαρτάται από το διάστημα που βρίσκονται τα (a,b,c). Αν $(a, b, c) \in [0, 1]$, τότε χρωματίζεται το pixel. Το χρώμα που επιλέγεται για το pixel, αν δουλεύουμε με λειτουργία flat, τότε και πάλι είναι ο μέσος όρος χρώματος των κορυφών. Αν από την άλλη πρέπει να εφαρμόσουμε χρωματισμό Gouraud το χρώμα είναι απλά:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} R_1 & R_2 & R_3 \\ G_1 & G_2 & G_3 \\ B_1 & B_2 & B_3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{5}$$

Εν τέλει τα αντικείμενα που εμφανίζονται με τον χρωματισμό είναι:

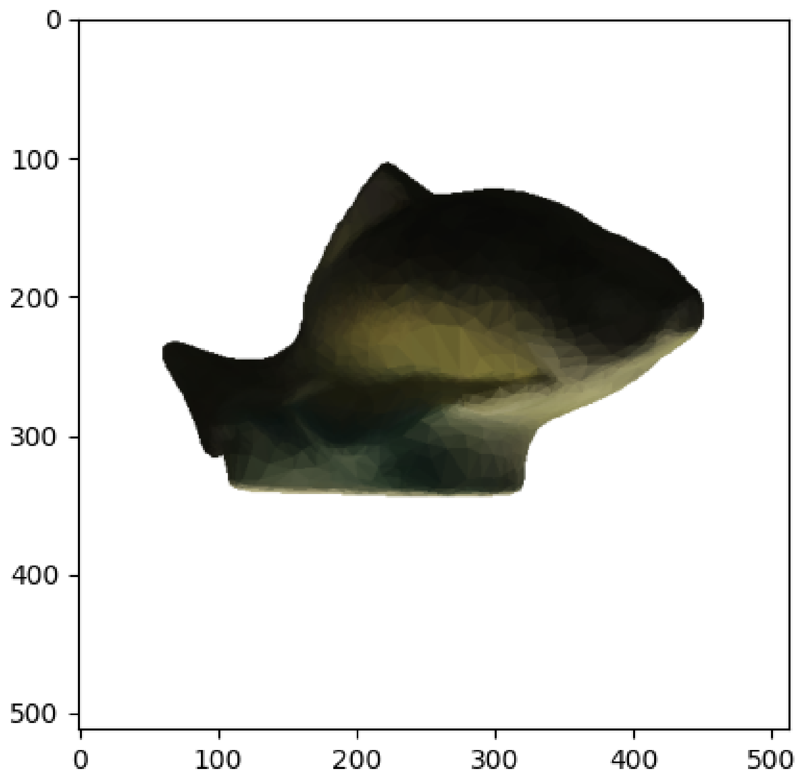


Figure 4: Το αντικείμενο χρωματισμένο με τρόπο flat με συντεταγμένες κέντρου βάρους

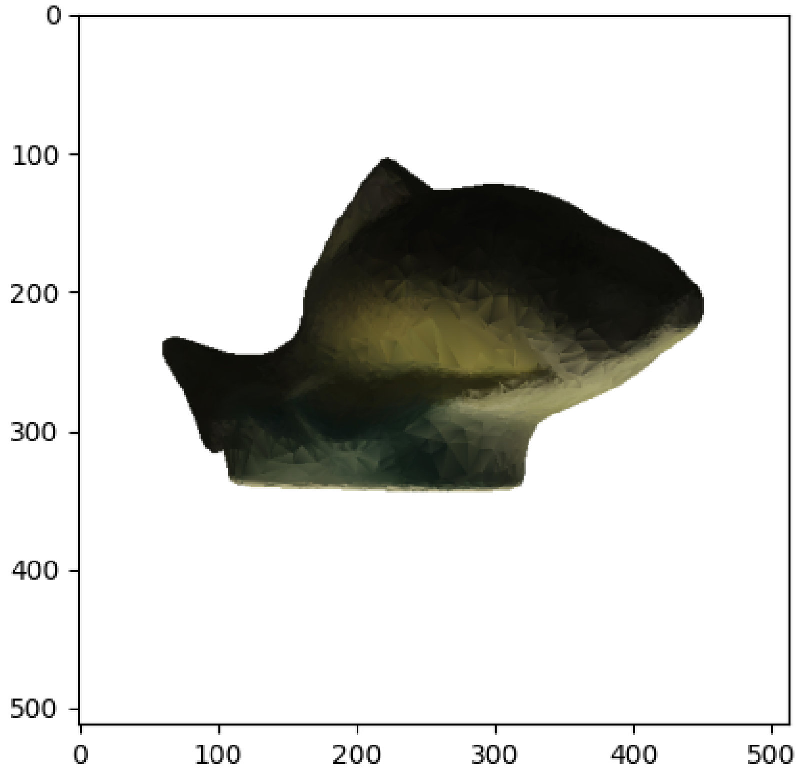


Figure 5: Το αντικείμενο χρωματισμένο με τρόπο Gouraud με συντεταγμένες κέντρου βάρους

Μπορούμε να δούμε ότι δεν υπάρχει κάποια ουσιαστική διαφορά στις εικόνες που χρωματίστηκαν με τον παραδοσιακό τρόπο με τις εικόνες που χρησιμοποιούν συντεταγμένες κέντρου βάρους σε τρίγωνο. Ο αλγόριθμος ακολουθεί την παρακάτω λογική:

Algorithm 3 Χρωματισμός Τριγώνου Εναλλακτικός

procedure Χρωματισμος Τριγωνου Εναλλακτικος($Image$, $2DVertices$, $VerticeColors$, $ShadeMode$)
  Ταξινομηση Κορυφων($2DVertices$)
  Εύρεση $x_{min}, x_{max}, y_{min}, y_{max}$
  for $y_{min} \leq y \leq y_{max}$ **do**
    for $x_{min} \leq x \leq x_{max}$ **do**
      $a, b, c \leftarrow$ Υπολογισμος Νεων Συντεταγμενων(x, y, 2D Vertices)
      if $a, b, c \in [0, 1]$ **then**
        Χρωματισμος Pixel(x, y)
      end if
    end for
  end for

Όπου ο αλγόριθμος χρωματισμός pixel αποδίδει στο pixel ή το flat χρώμα ή το Gouraud χρώμα της σχέσης (5).

5 Βάφιμο Αντικειμένων

Ενώ και οι 2 αλγόριθμοι που αναλύθηκαν πιο πριν είναι αρκετά ικανοποιητικοί, αφορούν τον χρωματισμό ενός τριγώνου. Δίνεται, λοιπόν, και η συνάρτηση Render:

Algorithm 4 Render

procedure RENDER($2DVertices$, $2DTriangleFaces$ $VerticeColors$, $Depth$,$ShadeMode$)
  $Image \leftarrow$ CREATECAMVAS($N, M, RGB \leftarrow True$)
  $TriangleDepth \leftarrow$ MEAN($2DTriangleVerticesDepth$) ▷ Μέσος όρος βάθους τριγώνων
  $Triangle \leftarrow$ SORT($TrianglesFaces, arg \leftarrow TriangleDepth$) ▷ Ταξινόμιση Τριγώνων Με βάση το βάθος
  for Τρίγωνο ∈ Τρίγωνα **do**
    Εξαγωγή Πληροφοριών Τριγώνου
    Εικόνα ← COLOR TRIANGLE($Triangle, TriangleInfo, ShadeMode$)
  end for

6 Αποτελέσματα και Συμπεράσματα

Είδαμε πως υλοποιούνται δύο διαφορετικής φιλοσοφίας αλγόριθμοι οι οποίοι όμως παράγουν το ίδιο αποτέλεσμα, στις εικόνες (2)-(4). Ενώ ο κλασικός αλγόριθμος είναι καλός, μπορεί να αρχίσει να υποφέρει από σφάλματα στρογγυλοποίησης, ειδικά σε ακμές με πολύ μεγάλη ή πολύ μικρή κλίση. Από την άλλη, ο αλγόριθμος που χρησιμοποιεί τα κέντρα βάρους μπορεί να γίνει πολύ αργός. Είναι πολύ λογικό να απαιτούμε πολύ καλή απόδοση από τέτοιου είδους αλγόριθμους.