Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Γραφική με Υπολογιστές

Εργασία 1 - Πλήρωση Τριγώνων

Λώλος Ιωάννης 10674

21 Απριλίου 2025

Περιεχόμενα

Σκοπός της εργασίας

Στόχο της εργασίας αποτελεί η υλοποίηση αλγορίθμων που στοχεύουν στην πλήρωση τριγώνων με τον κατάλληλο χρωματισμό. Για την υλοποίηση του ζητουμένου αναπτύχθηκαν και χρησιμοποιήθηκαν οι συναρτήσεις vector_interp, f_shading, t_shading, render_img και με την βοήθεια των δύο scripts demo_f και demo_t , έγινε επίδειξη τους, ώστε να διαπιστωθεί η ορθότητα τους.

Υλοποίηση των συναρτήσεων

2.1 Η συνάρτηση `vector_interp`

Η `vector_interp` αποτελεί μία βοηθητική συνάρτηση που παρέχει πολύ περιορισμένη λειτουργικότητα. Ουσιαστικά, πραγματοποιεί γραμμική παρεμβολή στο σημείο $p = (x, y)$ μεταξύ δύο διανυσμάτων V_1 και V_2, δεδομένων των σημείων $p_1 = (x_1, y_1)$ και $p_2 = (x_2, y_2)$ στα οποία αντιστοιχούν, υπό την προϋπόθεση ότι το p ανήκει στην ευθεία που ορίζουν τα σημεία. Το διάνυσμα που προκύπτει είναι:

$$V = (1 - \alpha)V_1 + \alpha V_2 \tag{2.1}$$

Το α προσδιορίζεται ως:

$$\alpha = \frac{coord - p_{1\,\text{dim}}}{p_{2\,\text{dim}} - p_{1\,\text{dim}}} \tag{2.2}$$

όπου το $coord$ αποτελεί όρισμα της συνάρτησης (τη συντεταγμένη προς παρεμβολή) και τα $p_{1\,\text{dim}}, p_{2\,\text{dim}}$ είναι είτε οι τετμημένες είτε οι τεταγμένες των p_1, p_2, ανάλογα με το αν το όρισμα dim είναι 1 ή 2. Πέρα από την βασική λειτουργία παρεμβολής, η συνάρτηση περιλαμβάνει ελέγχους ορθότητας:

- Έλεγχος ισότητας $p_1 = p_2$ (αποφυγή διαίρεσης με μηδέν),
- Έλεγχος εγκυρότητας του $dim \in \{1, 2\}$,
- Έλεγχος ότι το $coord$ βρίσκεται εντός του διαστήματος ορισμού.

Algorithm 1 `vector_interp`: Γραμμική παρεμβολή μεταξύ διανυσμάτων

Require: $p_1 = (x_1, y_1), p_2 = (x_2, y_2)$
Require: $V_1, V_2, coord, dim \in \{1, 2\}$
Ensure: Το παρεμβαλλόμενο διάνυσμα V

if $p_1 = p_2$ **then**
    return V_1
end if
if $dim \notin \{1, 2\}$ **then**
    raise `ValueError('dim must be 1 or 2')`
end if
$idx \leftarrow dim - 1$
$p_{1\,\text{dim}} \leftarrow p_1[idx], p_{2\,\text{dim}} \leftarrow p_2[idx]$
if $coord < \min(p_{1\,\text{dim}}, p_{2\,\text{dim}})$ **or** $coord > \max(p_{1\,\text{dim}}, p_{2\,\text{dim}})$ **then**
    raise `ValueError('coord is out of bounds')`
end if
if $p_{1\,\text{dim}} = p_{2\,\text{dim}}$ **then**
    return V_1
end if
$\alpha \leftarrow \dfrac{coord - p_{1\,\text{dim}}}{p_{2\,\text{dim}} - p_{1\,\text{dim}}}$
$V \leftarrow (1 - \alpha) \cdot V_1 + \alpha \cdot V_2$
return V

2.2 Η συνάρτηση `f_shading`

Η `f_shading` εφαρμόζει επίπεδη σκίαση σε ένα τρίγωνο που ορίζεται από τρεις δισδιάστατες κορυφές και τα αντίστοιχα χρώματά τους. Για τους σκοπούς της συνάρτησης δημιουργήθηκε μία βοηθητική κλάση, η `Edge`, η οποία περιέχει πληροφορίες όπως η κλίση και οι συντεταγμένες αρχής και τέλους κάθε ακμής του τριγώνου προς πλήρωση.
Αρχικά, υπολογίζεται το χρώμα με το οποίο θα πληρωθεί το τρίγωνο ως ο μέσος όρος των χρωμάτων των τριών κορυφών:

$$color_{avg} = (\bar{R}, \bar{G}, \bar{B}), \quad \bar{C} = \frac{C_1 + C_2 + C_3}{3}, \quad C \in \{R, G, B\} \tag{2.3}$$

Στη συνέχεια δημιουργούνται τα αντικείμενα `edges` από τις κορυφές του τριγώνου, οι οποίες έχουν δοθεί ως ορίσματα. Για λόγους αποδοτικότητας, περιορίζεται η περιοχή ενδιαφέροντος του αλγορίθμου (region of interest) στα διαστήματα των x και y για τα οποία υπάρχουν ακμές.

Ακολουθεί το τμήμα του αλγορίθμου που βασίζεται στη μέθοδο *scanline*. Για κάθε τιμή της τεταγμένης y εντός της περιοχής ενδιαφέροντος, υπολογίζονται οι `active_edges`, δηλαδή οι ακμές που τέμνονται με την οριζόντια ευθεία $y = \text{const}$ για τη συγκεκριμένη γραμμή. Τα σημεία τομής υπολογίζονται και χρησιμοποιούνται στο επόμενο στάδιο του αλγορίθμου.
Εφόσον υπάρχουν ακριβώς δύο σημεία τομής, στρογγυλοποιούνται στις πλησιέστερες ακέραιες τιμές και χρωματίζονται όλα τα pixels μεταξύ τους με το προκαθορισμένο μέσο χρώμα (`average color`).

Algorithm 2 `f_shading`: Εφαρμογή επίπεδης σκίασης σε τρίγωνο

Require: img: εικόνα $H \times W \times 3$
Require: $vertices$: λίστα 3 κορυφών $[x_i, y_i]$
Require: $vcolors$: λίστα 3 χρωμάτων $[r_i, g_i, b_i]$
Ensure: Ενημερωμένη εικόνα με σκιασμένο τρίγωνο

```
avg_color ← μέσος όρος των vcolors
Κατασκευή 3 ακμών edges από τις κορυφές
min_y, max_y ← εύρος y τιμών του τριγώνου
for y από min_y έως max_y do
    active_edges ← ακμές με y ∈ [min_y, max_y)
    x_intersects ← τομές των ακμών με τη γραμμή y
    if len(x_intersects) = 2 then
        x_1, x_2 ← ταξινομημένα x_intersects
        Γέμισε τα pixels img[y, x_1 : x_2] με avg_color
    end if
end for
return img
```

2.3 Η συνάρτηση `t_shading`

Η `t_shading` εφαρμόζει σκίαση με υφή (*texture shading*) σε τρίγωνο, χρησιμοποιώντας τις δισδιάστατες συντεταγμένες UV των κορυφών του. Ο αλγόριθμος γεμίζει το τρίγωνο με τιμές από μία εξωτερική εικόνα (texture image), παρεμβάλλοντας τις UV συντεταγμένες ώστε να αντιστοιχούν στα κατάλληλα σημεία της texture εικόνας.
Για κάθε γραμμή y στην περιοχή του τριγώνου, υπολογίζονται οι ενεργές ακμές (`active_edges`) και τα αντίστοιχα σημεία τομής στον άξονα x καθώς και τα UV διανύσματα μέσω της βοηθητικής συνάρτησης `vector_interp`. Έπειτα, για κάθε x μεταξύ των δύο τομών, γίνεται δεύτερη παρεμβολή των UV συντεταγμένων κατά μήκος της γραμμής ώστε να προκύψει το τελικό uv σημείο, το

οποίο μετατρέπεται σε δείκτη pixel και χρησιμοποιείται για την απόδοση του τελικού χρώματος από την εικόνα υφής.

Algorithm 3 `t_shading`: Εφαρμογή texture shading σε τρίγωνο

Require: img: εικόνα $H \times W \times 3$
Require: $vertices$: κορυφές τριγώνου $[x_i, y_i]$
Require: uv: πίνακας 3×2 με UV συντεταγμένες
Require: $textImg$: εικόνα texture
Κατασκευή 3 ακμών $edges$ με θέσεις και UV από τις κορυφές
$min\_y, max\_y \leftarrow$ εύρος y τιμών του τριγώνου
for y από min_y έως max_y **do**
    $active\_edges \leftarrow$ ακμές με $y \in [min_y, max_y)$
    if όχι 2 τομές **then**
        continue
    end if
    Υπολογισμός x_0, x_1 μέσω παρεμβολής
    Υπολογισμός uv_0, uv_1 μέσω `vector_interp`
    for x από x_0 έως x_1 **do**
        $uv \leftarrow$ παρεμβολή μεταξύ uv_0, uv_1 με `vector_interp`
        Μετατροπή uv σε δείκτη $[u, v]$ στην εικόνα υφής
        Απόδοση χρώματος στο $img[y, x]$
    end for
end for
return img

2.4 Η συνάρτηση `render_img`

Η `render_img(faces, vertices, vcolors, uvs, depth, shading, texImg)` αξιοποιεί όλες τις προηγούμενες συναρτήσεις ώστε να παραγάγει την τελική εικόνα. Αρχικά, δημιουργείται ένας πίνακας `img` διαστάσεων $512 \times 512 \times 3$, καθώς και ένας κενός πίνακας `face_depths`.
Στη συνέχεια, για κάθε στοιχείο του πίνακα `faces`, υπολογίζεται το βάθος του αντίστοιχου τριγώνου ως ο μέσος όρος των τιμών `depth` στις κορυφές του και προστίθεται στον πίνακα faces. Μετά το πέρας της διαδικασίας, ο πίνακας ταξινομείται κατά φθίνουσα σειρά βάθους, έτσι ώστε τα μακρινά τρίγωνα να σχεδιαστούν πρώτα.
Τέλος, για κάθε τρίγωνο του πίνακα καλείται η συνάρτηση `f_shading` εφόσον `shading = "f"`, αλλιώς η `t_shading` αν `shading = "t"`, με ορίσματα την εικόνα και τα στοιχεία του τριγώνου. Μετά την επεξεργασία όλων

των τριγώνων επιστρέφεται η τελική εικόνα.

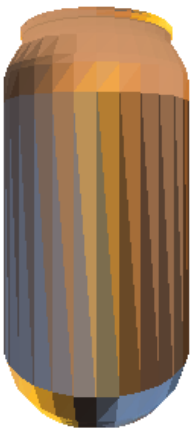
Algorithm 4 `render_img`: Τελική απόδοση εικόνας με επιλογή σκίασης

Require: $faces$, $vertices$, $vcolors$, $depth$, $shading$
Ensure: img: εικόνα $512 \times 512 \times 3$

Αρχικοποίησε img και $triangles \leftarrow []$
for κάθε f στο $faces$ **do**
    $vtx \leftarrow$ κορυφές από $vertices$ με δείκτες f
    $col \leftarrow$ χρώματα από $vcolors$ με δείκτες f
    $z \leftarrow$ μέσος όρος των $depth[f]$
    Κατασκεύασε $triangle \leftarrow \{\texttt{vertices}, \texttt{color}, \texttt{depth}\}$
    Προσθήκη $triangle$ στο $triangles$
end for
Ταξινόμηση $triangles$ κατά `depth` φθίνουσα (πίσω πρώτα)
for κάθε $triangle$ στο $triangles$ **do**
    if $shading =$ `"f"` **then**
        `f_shading`(img, $triangle.vertices$, $triangle.color$)
    else if $shading =$ `"t"` **then**
        `t_shading`(img, $triangle.vertices$, $triangle.uv$, $textImg$)
    end if
end for
return img

Αποτελέσματα

Στην παρούσα ενότητα παρουσιάζονται τα αποτελέσματα της απόδοσης τριγώνων με την χρήση των δύο τεχνικών σκίασης που παρουσιάστηκαν στο chapter 2. Για τις δοκιμές χρησιμοποιήθηκαν κοινά δεδομένα εισόδου, ώστε να διευκολυνθεί η άμεση σύγκριση των παραγόμενων εικόνων.
Οι εικόνες που ακολουθούν προέκυψαν από την εφαρμογή της συνάρτησης `render_img` με κατάλληλη επιλογή παραμέτρων σκίασης.



Σχήμα 3.1: Απόδοση τριγώνων με επίπεδη σκίαση (`flat shading`).

Σχήμα 3.2: Απόδοση τριγώνων με σκίαση υφής (`texture shading`).

Όπως φαίνεται στα fig. 3.1 και fig. 3.2, η χρήση `textures shading` επιτρέπει λεπτομερέστερη απεικόνιση, ενώ η `flat shading` προσφέρει μία πιο αφαιρετική αλλά υπολογιστικά απλούστερη προσέγγιση. Οι χρόνοι εκτέλεσης μετρήθηκαν και βρέθηκαν κατα μέσο όρο περίπου 3.07s για την textured shading και 0.54s για την flat shading στον υπολογιστή μου (AMD Ryzen 7 4700U, Ubuntu 24.04)

Συζήτηση - Συμπεράσματα

Στην παρούσα εργασία υλοποιήθηκαν βασικές τεχνικές σκίασης τριγώνων στο πλαίσιο ενός απλού rasterization pipeline. Πιο συγκεκριμένα, αναπτύχθηκαν και εφαρμόστηκαν οι μέθοδοι `flat shading` και `texture shading`, μέσω των συναρτήσεων `f_shading` και `t_shading` αντίστοιχα.
Η επίπεδη σκίαση επιτυγχάνει μία ομοιόμορφη χρωματική απόδοση σε κάθε τρίγωνο, προσφέροντας εξαιρετική υπολογιστική απόδοση και απλότητα στην υλοποίηση. Ωστόσο, υστερεί σε ρεαλισμό, καθώς δεν υπάρχει εσωτερική διαφοροποίηση του χρώματος.
Αντίθετα, η σκίαση με υφή παρέχει τη δυνατότητα εξαιρετικά λεπτομερούς απόδοσης βασισμένης σε UV συντεταγμένες και τη χρήση εξωτερικών εικόνων για την κάλυψη των επιφανειών. Ωστόσο είναι υπολογιστικά σημαντικά πιο απαιτητική.

4.1 Μελλοντικές Βελτιώσεις

Θεωρώ πως η παρούσα προσέγγιση μπορεί να επεκταθεί με τους εξής τρόπους:

- Μετάβαση των αλγορίθμων σε μία γλώσσα υψηλότερων αποδόσεων, όπως η `C++`
- Βελτιστοποίηση των ήδη υπαρχόντων συναρτήσεων προς αποφυγή περιττών επαναλήψεων, υπολογισμών και αναθέσεων
- Παραλληλοποίηση των αλγορίθμων σε εκτενέστερο επίπεδο από αυτό της βιβλιοθήκης `numpy`

Η εργασία αυτή θέτει τις βάσεις για περαιτέρω εξερεύνηση του 3D rendering μέσω rasterization, τόσο σε ακαδημαϊκό όσο και σε εφαρμοστικό επίπεδο.