ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Γραφική με Υπολογιστές Αναφορά

Εργασία 1

Διακολουκάς Δημήτριος
ΑΕΜ 10642

Email: ddiakolou@ece.auth.gr

Περιεχόμενα

Κεφάλαιο 1

Περιγραφή Προβλήματος και Βοηθητικές Συναρτήσεις

Η παρούσα εργασία έχει ως στόχο την υλοποίηση μεθόδων για την απόδοση χρώματος (shading) σε τριγωνικά πλέγματα που προβάλλονται από έναν 3D χώρο σε δισδιάστατο (2D) καμβά.

Το αρχείο εισόδου `hw1.npy` περιέχει δεδομένα για τις κορυφές (vertices), τα χρώματα (vcolors), τους δείκτες θέσεων τριγώνων (faces ή `t_pos_idx`), τις τιμές βάθους (depth) και τις συντεταγμένες υφής (UV coordinates). Τα δεδομένα αυτά αξιοποιούνται από τη συνάρτηση `render_img` για να αποδοθεί η τελική εικόνα.

Ο καμβάς στον οποίο γίνεται η σχεδίαση είναι διαστάσεων $M \times N = 512 \times 512$ και έχει λευκό υπόβαθρο. Κάθε τρίγωνο σχεδιάζεται σε σειρά, ξεκινώντας από το πιο μακρινό (μεγαλύτερη τιμή `depth`) προς το πιο κοντινό.

Σκοπός: Να χρωματιστούν K τρίγωνα (από τον πίνακα `faces`) με βάση δύο διαφορετικές μεθόδους:

- **Flat shading:** Κάθε τρίγωνο χρωματίζεται με το μέσο όρο των χρωμάτων των κορυφών του.
- **Texture shading:** Χρήση εικόνας υφής (`textImg`) με παρεμβολή UV συντεταγμένων.

Βοηθητική Συνάρτηση `vector_interp`

Για τις ανάγκες της παρεμβολής τιμών κατά μήκος ευθύγραμμων τμημάτων (π.χ., για υπολογισμό UV συντεταγμένων ή RGB χρωμάτων), χρησιμοποιούμε τη βοηθητική συνάρτηση `vector_interp`. Έστω δύο σημεία $p_1 = (x_1, y_1)$ και $p_2 = (x_2, y_2)$, με αντίστοιχα διανύσματα V_1 και V_2 (π.χ. $V_1, V_2 \in \mathbb{R}^3$ για RGB ή $V_1, V_2 \in \mathbb{R}^2$ για UV).
Θεωρούμε ένα ενδιάμεσο σημείο $p = (x, y)$ που βρίσκεται πάνω στο ευθύγραμμο τμήμα p_1p_2. Αν θέλουμε να υπολογίσουμε τη διανυσματική τιμή V στο σημείο p μέσω παρεμβολής στον άξονα x ή y, τότε:

$$t = \frac{c - c_1}{c_2 - c_1}$$

$$V = (1 - t) \cdot V_1 + t \cdot V_2$$

όπου:

- c είναι η τετμημένη x ή τεταγμένη y του σημείου p
- c_1, c_2 είναι οι αντίστοιχες συντεταγμένες των p_1 και p_2
- $t \in [0, 1]$ είναι ο παράγοντας παρεμβολής
- V είναι το διάνυσμα που προκύπτει στην ενδιάμεση θέση p

Αυτός ο τύπος χρησιμοποιείται εκτενώς για τον υπολογισμό ενδιάμεσων χρωμάτων και UV συντεταγμένων κατά τον χρωματισμό (shading) ενός τριγώνου και υλοποιείται στο αρχείο `vector_interp`. Είναι βασική για το `texture shading`, καθώς χρησιμοποιείται για την παρεμβολή UV συντεταγμένων σε κάθε scanline και σημείο του τριγώνου.

Κεφάλαιο 2

Διαδικασία Χρωματισμού Τριγώνων και Ψευδοκώδικας

Σε αυτή την εργασία, εφαρμόστηκαν δύο διαφορετικές τεχνικές για τον χρωματισμό των τριγώνων που σχηματίζουν την προβολή ενός 3D αντικειμένου στον δισδιάστατο καμβά (2D canvas). Οι τεχνικές αυτές είναι οι εξής:

- Flat Shading που είναι απλός, ομοιόμορφος χρωματισμός με βάση το μέσο όρο χρωμάτων των κορυφών κάθε τριγώνου.
- Texture Mapping που είναι για χαρτογράφηση υφής με χρήση UV συντεταγμένων που επιτρέπουν την απεικόνιση μιας εικόνας υφής πάνω στο γεωμετρικό πλέγμα.

Και στις δύο περιπτώσεις, ο αλγόριθμος διασχίζει κάθε τρίγωνο ξεχωριστά, και με βάση τη μέθοδο που έχει επιλεχθεί, χρωματίζει τα εσωτερικά του σημεία.

2.1 Flat Shading

Η τεχνική flat shading βασίζεται στον απλό και γρήγορο υπολογισμό του χρώματος κάθε τριγώνου. Το τελικό χρώμα του τριγώνου προκύπτει από το διανυσματικό μέσο όρο των χρωμάτων των τριών κορυφών του. Έπειτα, όλα τα πιξελς που εμπίπτουν εντός του τριγώνου βάφονται με αυτό το ενιαίο χρώμα.

Διαδικασία υλοποίησης

1. Για κάθε τρίγωνο, προσδιορίζεται το ελάχιστο παραλληλόγραμμο (bounding box)που το περικλείει.
2. Για κάθε εσωτερικό pixel εντός του bounding box:
    - Υπολογίζεται αν το πιξελ ανήκει στο τρίγωνο με χρήση της edge function, η οποία βασίζεται σε εσωτερικά γινόμενα.
    - Αν το πιξελ βρίσκεται εντός του τριγώνου, τότε αποδίδεται σε αυτό το *μέσο χρώμα* των κορυφών.

Ψευδοκώδικας

Flat Shading Pseudocode

```
flat_color = mean([vcolor0, vcolor1, vcolor2])

for j in range(min_y, max_y):
    for i in range(min_x, max_x):
        if point_in_triangle((i + 0.5, j + 0.5)):
            img[j, i] = flat_color
```

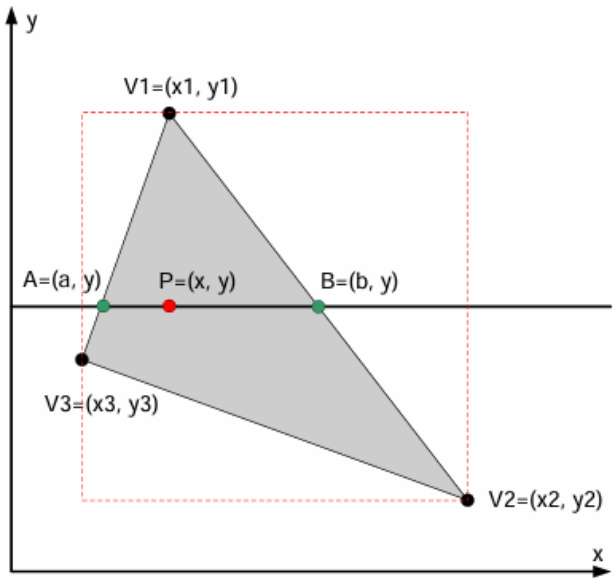
Η πλήρης υλοποίηση αυτής της τεχνικής βρίσκεται στο αρχείο `flat_shading.py` και ειδικότερα στη συνάρτηση `f_shading`, η οποία καλείται για κάθε τρίγωνο μέσα από τη render_img.

2.2 Texture Shading (UV Mapping)

Η τεχνική texture mapping προσφέρει πιο ρεαλιστικό αποτέλεσμα, καθώς επιτρέπει σε κάθε τρίγωνο να «προβάλλει» μια περιοχή της εικόνας υφής πάνω του, με βάση τις UV συντεταγμένες των κορυφών του. Η μέθοδος αυτή χρησιμοποιεί την βοηθητική συνάρτηση `vector_interp` για γραμμική παρεμβολή.

Διαδικασία υλοποίησης

1. Ταξινομούνται οι κορυφές του τριγώνου κατά αύξουσα σειρά ως προς τον άξονα y.
2. Για κάθε scanline y (δηλ. οριζόντια γραμμή):
   - Υπολογίζονται τα σημεία τομής A και B στα δύο άκρα του τριγώνου (αριστερό και δεξί), χρησιμοποιώντας την `vector_interp` ώστε να εντοπιστούν τα UV των σημείων αυτών.
   - Εντός του διαστήματος $[A_x, B_x]$, γίνεται νέα παρεμβολή των UV για κάθε x και εξάγεται το κατάλληλο χρώμα από την εικόνα υφής.
3. Κάθε pixel (x, y) βάφεται με βάση το χρώμα που εντοπίζεται στην υφή στις συντεταγμένες UV.



Σχήμα 2.1: Παράδειγμα γραμμικής παρεμβολής κατά το `Texture Mapping`. Το σημείο $P = (x, y)$ αποκτά το χρώμα του μέσω παρεμβολής των UV συντεταγμένων στα σημεία A και B, τα οποία προκύπτουν με παρεμβολή πάνω στις ακμές του τριγώνου. Έτσι γίνεται και η βασική υλοποίηση στον κώδικα

Ψευδοκώδικας

Texture Shading Pseudocode

```
for y in min_y to max_y:
    if y < v1.y:
        A = vector_interp(v0, v1, v0, v1, y, dim=2)
        uv_A = vector_interp(v0, v1, uv0, uv1, y, dim=2)
    else:
        A = vector_interp(v1, v2, v1, v2, y, dim=2)
        uv_A = vector_interp(v1, v2, uv1, uv2, y, dim=2)

    B = vector_interp(v0, v2, v0, v2, y, dim=2)
    uv_B = vector_interp(v0, v2, uv0, uv2, y, dim=2)

    for x in ceil(A.x) to floor(B.x):
        uv_P = vector_interp(A, B, uv_A, uv_B, x, dim=1)
        color = texture_img[uv_P]
        img[y, x] = color
```

Η τεχνική αυτή υλοποιείται πλήρως στο αρχείο `texture_shading.py` και στη συνάρτηση `t_shading`. Η χρήση της παρεμβολής επιτρέπει την απόδοση ρεαλιστικών λεπτομερειών της υφής πάνω στην επιφάνεια του τριγώνου.

Κεφάλαιο 3

Υλοποίηση και Κλήση Προγραμμάτων

Σε αυτό το κεφάλαιο περιγράφεται η γενική δομή και ο τρόπος εκτέλεσης των προγραμμάτων που υλοποιήθηκαν για τις τεχνικές Flat Shading και Texture Shading. Η συνολική ροή περιλαμβάνει την ανάγνωση των δεδομένων, την επιλογή τεχνικής απόδοσης και την αποτύπωση του τελικού αποτελέσματος στον καμβά.

3.1 Δομή Αρχείων και Κώδικα

Η εφαρμογή αποτελείται από τα παρακάτω κύρια αρχεία:

- `flat_shading.py`– Υλοποιεί την τεχνική Flat Shading.
- `texture_shading.py`– Υλοποιεί την τεχνική Texture Shading.
- `vector_interp.py`- Περιέχει τη βοηθητική συνάρτηση γραμμικής παρεμβολής `vector_interp`.
- `render_utils.py`– Περιέχει τη συνάρτηση `render_img` η οποία οργανώνει τη διαδικασία χρωματισμού.
- `demo_f.py`– Κύριο πρόγραμμα εκτέλεσης για τεχνική Flat Shading.
- `demo_g.py`– Κύριο πρόγραμμα εκτέλεσης για τεχνική Texture Shading.

3.2 Συνάρτηση `render_img`

Η `render_img()`είναι η βασική συνάρτηση απόδοσης. Αναλαμβάνει την προετοιμασία του καμβά, την ταξινόμηση των τριγώνων βάσει βάθους (για ορθό χειρισμό επικάλυψης) και την κλήση της κατάλληλης μεθόδου χρωματισμού για κάθε τρίγωνο.

Περιγραφή Λειτουργίας

- Δημιουργεί εικόνα 512×512 με λευκό φόντο.
- Υπολογίζει το μέσο βάθος κάθε τριγώνου με χρήση του πίνακα `depth`.
- Ταξινομεί τα τρίγωνα σε σειρά από το πιο μακρινό στο πιο κοντινό.
- Για κάθε τρίγωνο:

- Αν η τεχνική είναι `flat`, καλείται η `f_shading()`.
- Αν η τεχνική είναι `texture`, καλείται η `t_shading()`.

- Επιστρέφει την τελική εικόνα.

Ψευδοκώδικας

render_img() Pseudocode

```
function render_img(faces,vertices,vcolors,uvs,depth,shading,texImg):

    img = create_white_canvas(512, 512)
    triangle_depths = mean(depth[faces], axis=1)
    sorted_indices = sort_descending(triangle_depths)

    for i in sorted_indices:
        face = faces[i]
        triangle_vertices = vertices[face]

        if shading == 'f':
            triangle_colors = vcolors[face]
            img = f_shading(img,triangle_vertices,triangle_colors)

        else if shading == 't':
            triangle_uv = uvs[face]
            img = t_shading(img,triangle_vertices,triangle_uv,texImg)

    return img
```

3.3 Τρόπος Κλήσης Προγραμμάτων

Υπάρχουν δύο κύρια εκτελέσιμα προγράμματα:

1. Flat Shading: `demo_f.py`

- Χρησιμοποιεί την τεχνική `f_shading`.
- Δεν χρησιμοποιεί UV συντεταγμένες ή εικόνα υφής.
- Το αποτέλεσμα αποθηκεύεται ως `flat_result.png`.

Εκτέλεση:

```
python demo_f.py
```

2. Texture Shading: `demo_g.py`

- Χρησιμοποιεί την τεχνική `t_shading`.
- Φορτώνει εικόνα υφής με χρήση της `load_png_image()`.
- Το αποτέλεσμα αποθηκεύεται ως `texture_result.png`.

Εκτέλεση:

```
python demo_g.py
```

3.4 Ροή Δεδομένων

Η συνολική ροή των δεδομένων έχει ως εξής:

`hw1.npy` → `main()` → `render_img()` → `f_shading / t_shading` → Τελική εικόνα

Τα δεδομένα περιλαμβάνουν:

- $L \times 2$ πίνακα κορυφών (`v_pos2d`)
- $L \times 3$ πίνακα χρωμάτων (`v_clr`)
- $L \times 1$ πίνακα βάθους (`depth`)
- $L \times 2$ πίνακα UV συντεταγμένων (`v_uvs`)
- $K \times 3$ πίνακα συνδεσιμότητας (`t_pos_idx`)

3.5 Αποτελέσματα

Κάθε δεμο παράγει εικόνα με τις ακόλουθες ιδιότητες:

- Διαστάσεις: $512 \times 512 \times 3$
- Αρχικά λευκός καμβάς
- Εμφανίζει το αντικείμενο ως ένωση χρωματισμένων τριγώνων
- Αποθηκεύεται τοπικά σε μορφή `PNG`

Κεφάλαιο 4

Αποτελέσματα, Οπτικοποίηση, Παρατηρήσεις

Σε αυτό το κεφάλαιο παρουσιάζονται τα παραγόμενα αποτελέσματα από τις τεχνικές Flat Shading και Texture Mapping που υλοποιήθηκαν στα αντίστοιχα προγράμματα. Για λόγους σύγκρισης και κατανόησης των διαφορών, παρατίθενται τόσο οι τελικές εικόνες όσο και η εικόνα υφής που χρησιμοποιήθηκε στην τεχνική χαρτογράφησης υφής.

4.1 Περιγραφή Εισόδου – Αρχείο `hw1.npy`

Το αρχείο `hw1.npy`περιέχει όλα τα απαραίτητα δεδομένα για την απόδοση της προβολής ενός 3D αντικειμένου στον δισδιάστατο καμβά. Πρόκειται για ένα `Python dictionary` που περιλαμβάνει τις εξής δομές δεδομένων:

- `v_pos2d`: NumPy array διαστάσεων $(9641, 2)$. Περιέχει τις δισδιάστατες συντεταγμένες των κορυφών του αντικειμένου μετά την προβολή τους στον καμβά (σε pixels).
- `v_uvs`: NumPy array διαστάσεων $(9641, 2)$. Περιέχει τις κανονικοποιημένες UV συντεταγμένες για κάθε κορυφή, οι οποίες χρησιμοποιούνται για τη χαρτογράφηση υφής.
- `v_clr`: NumPy array διαστάσεων $(9641, 3)$. Περιέχει τις χρωματικές τιμές (RGB) των κορυφών, με τιμές στο διάστημα $[0, 1]$.
- `t_pos_idx`: Πίνακας τύπου `TrackedArray` (από το `trimesh`) διαστάσεων $(7504, 3)$. Κάθε γραμμή του πίνακα περιέχει τρεις δείκτες προς τις κορυφές που σχηματίζουν ένα τρίγωνο.
- `depth`: NumPy array διαστάσεων $(9641,)$. Περιέχει την τιμή βάθους (βάσει προβολής) κάθε κορυφής στο αρχικό τρισδιάστατο μοντέλο. Το βάθος χρησιμοποιείται για τον σωστό χρωματισμό από πίσω προς τα εμπρός (z-buffering).

Οι παραπάνω πληροφορίες φορτώνονται στην αρχή κάθε `demo` αρχείου με την εντολή:

```
data = np.load("hw1.npy", allow_pickle=True).item()
```

και στη συνέχεια διαβάζονται με βάση τα αντίστοιχα `keys`.

Αξίζει να σημειωθεί ότι το μοντέλο αποτελείται από 9641 κορυφές και 7504 τρίγωνα, τα οποία καλύπτουν ολόκληρη την επιφάνεια του αντικειμένου.

4.2 Οπτικοποίηση - Demos Output

Σχήμα 4.1: Απόδοση του αντικειμένου με τεχνική Flat Shading.

Σχήμα 4.2: Απόδοση του αντικειμένου με τεχνική Texture Mapping.

Σχήμα 4.3: Εικόνα υφής που χρησιμοποιήθηκε για την τεχνική Texture Mapping.

4.3 Παρατηρήσεις

- Η τεχνική **Flat Shading** είναι πιο απλή υπολογιστικά και οδηγεί σε ένα «μονοχρωματικό» αποτέλεσμα ανά τρίγωνο. Αυτό όμως προκαλεί μια απότομη μετάβαση χρωμάτων μεταξύ των τριγώνων, ειδικά σε περιοχές με έντονες μεταβολές στο χρώμα.
- Η τεχνική **Texture Mapping** αποδίδει ένα πολύ πιο ρεαλιστικό αποτέλεσμα, καθώς η εικόνα υφής «τυλίγεται» γύρω από το αντικείμενο με βάση τις UV συντεταγμένες. Το αποτέλεσμα εξαρτάται άμεσα από την ποιότητα και σωστή τοποθέτηση της υφής.
- Όπως φαίνεται στις παραπάνω εικόνες, η υφή με την εικόνα του Αγίου Γεωργίου εφαρμόζεται επιτυχώς επάνω στην κυλινδρική επιφάνεια του αντικειμένου, αποδεικνύοντας τη σωστή λειτουργία του αλγορίθμου.
- Σημαντικό ρόλο παίζει η `vector_interp`, η οποία εξασφαλίζει τη σωστή γραμμική παρεμβολή UV συντεταγμένων κατά μήκος κάθε `scanline`, έτσι ώστε κάθε `pixel` να αντιστοιχεί σε σωστή θέση της εικόνας υφής.

4.4 Συμπεράσματα

Η υλοποίηση πέτυχε τους στόχους της, προσφέροντας δύο σαφώς διακριτές μεθόδους χρωματισμού. Η Flat Shading προτείνεται για εφαρμογές με περιορισμένους πόρους, ενώ η Texture Mapping είναι ιδανική για φωτορεαλιστική απόδοση γεωμετρικών μοντέλων. Η σωστή προετοιμασία των δεδομένων (UV, βάθος, κανονικοποίηση) είναι κρίσιμη για την επιτυχία κάθε τεχνικής.

Βιβλιογραφία

[1] https://docs.opencv.org/4.x/d9/df8/tutorial_root.html