ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Γραφική με Υπολογιστές Αναφορά

Εργασία 3

Διακολουκάς Δημήτριος
ΑΕΜ 10642

Email: ddiakolou@ece.auth.gr

Περιεχόμενα

1 Περιγραφή Προβλήματος και Ζητούμενα **2**
- 1.1 Εισαγωγή . . . 2
- 1.2 Δοσμένα Δεδομένα – `hw3.npy` και δοθείσα εικόνα . . . 2
- 1.3 Ζητούμενα . . . 3
- 1.4 Προσομοιώσεις και Απαιτήσεις . . . 4

2 Περιληπτική περιγραφή βοηθητικών συναρτήσεων **5**
- 2.1 Συνάρτηση γραμμικής παρεμβολής μεταξύ διανυσμάτων . . . 5
- 2.2 Συστήματα Συντεταγμένων και Προβολές . . . 5
- 2.3 Συσχέτιση με Επίπεδο Εικόνας . . . 6

3 Μοντελοποίηση Φωτισμού και Υλικού Επιφάνειας **8**
- 3.1 Θεωρητικό Υπόβαθρο . . . 8
- 3.2 Κλάση `MatPhong` . . . 8
- 3.3 Συνάρτηση `light` . . . 9

4 Ανάλυση Μεθόδων Σκιασμού και Object Rendering **11**
- 4.1 Υπολογισμός Κανονικών Διανυσμάτων: `calc_normals` . . . 11
- 4.2 Σκιασμός ανά Κορυφή: `shade_gouraud` . . . 11
- 4.3 Σκιασμός ανά Pixel: `shade_phong` . . . 12
- 4.4 Απόδοση Αντικειμένου: `render_object` . . . 13

5 Κλήση Συναρτήσεων, Αποτελέσματα και Παρατηρήσεις **14**
- 5.1 Διαδικασία Εκτέλεσης . . . 14
- 5.2 Αποτελέσματα ανά Τεχνική Σκιασμού και Μοντέλο Φωτισμού . . . 14
- 5.3 Απόδοση Μεμονωμένων Πηγών Φωτός (Phong Only) . . . 17
- 5.4 Συμπεράσματα και Παρατηρήσεις . . . 18

Κεφάλαιο 1

Περιγραφή Προβλήματος και Ζητούμενα

1.1 Εισαγωγή

Η τρίτη εργασία της σειράς μαθημάτων *Γραφική με Υπολογιστές* εστιάζει στη δημιουργία ενός πλήρους rendering pipeline για την απεικόνιση 3D αντικειμένων υπό φωτισμό. Το σύστημα αυτό βασίζεται στη γεωμετρική και φυσική μοντελοποίηση φωτός και υλικών, αξιοποιώντας τις υλοποιήσεις των προηγούμενων εργασιών (transformations, projection, rasterization) και επεκτείνοντάς τες με υποστήριξη φωτισμού τύπου Phong και σκιασμού Gouraud και Phong.

Καλούμαστε να υλοποιήσουμε απεικόνιση ενός δεδομένου 3D αντικειμένου, αξιοποιώντας μία εικόνα υφής και δεδομένα καμπύλωσης επιφάνειας (κανονικά διανύσματα), ώστε να παραχθούν στατικά frames (εικόνες) υπό διαφορετικές παραμέτρους φωτισμού και σκιασμού σύμφωνα με μεθόδους που ενδείκνυνται στην εκφώνηση.

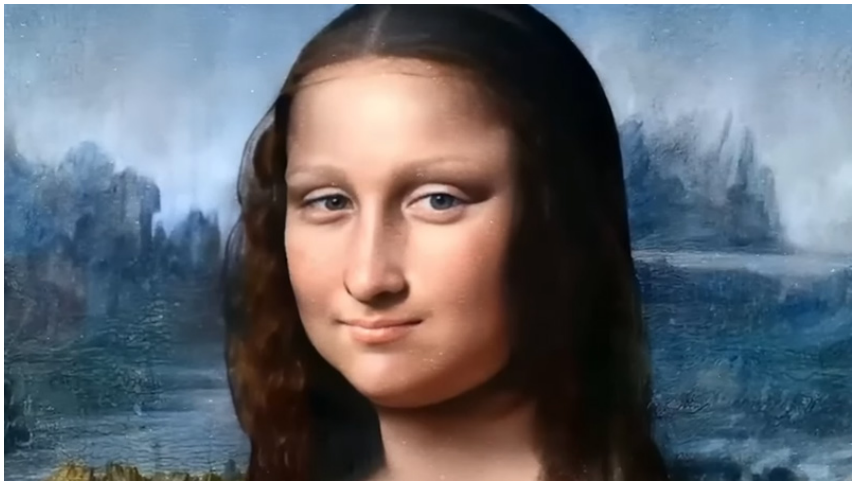
1.2 Δοσμένα Δεδομένα – `hw3.npy` και δοθείσα εικόνα

Το αρχείο `hw3.npy` περιέχει όλα τα δεδομένα της σκηνής που απαιτούνται για την παραγωγή εικόνων υπό φωτισμό. Επίσης για την υλοποίηση της εργασίας δόθηκε και μία εικόνα με αναπαράσταση της ***Mona Lisa*** ονόματι *Mona-Lisa-Exist-in-Real-Life-2635825581.png*. Τα δεδομένα είναι:

- **v_pos:** $3 \times N$ πίνακας με τις συντεταγμένες των κορυφών του αντικειμένου.
- **v_uvs:** $N \times 2$ πίνακας με τις UV συντεταγμένες υφής για κάθε κορυφή.
- **t_pos_idx:** $F \times 3$ πίνακας που ορίζει τα τρίγωνα μέσω δεικτών στις κορυφές.
- **tex:** Εικόνα υφής του αντικειμένου.
- **cam_pos, target, up:** Συντεταγμένες κάμερας και προσανατολισμού της.
- **plane_w, plane_h, focal, res_w, res_h:** Γεωμετρικά χαρακτηριστικά και ανάλυση του image plane.
- **l_pos:** Θέσεις σημειακών πηγών φωτός (point lights).

- **l_int:** Εντάσεις των πηγών φωτός (light intensity).
- **l_amb:** Ένταση περιβαλλοντικού φωτισμού (ambient light).
- **ka, kd, ks, n:** Συντελεστές υλικού (ambient, diffuse, specular, Phong exponent).

Η εικόνα που δόθηκε για την υλοποίηση της εργασίας είναι η παρακάτω:



Σχήμα 1.1: Reference εικόνα υφής από τα δεδομένα.

1.3 Ζητούμενα

Η εργασία απαιτεί την υλοποίηση συστήματος φωτισμού και σκιασμού. Πιο αναλυτικά:

1. **MatPhong:** Κλάση που περιγράφει τα χαρακτηριστικά του υλικού ενός αντικειμένου.
2. **light(...):** Συνάρτηση υπολογισμού της έντασης φωτός σε σημείο επιφάνειας υπό το μοντέλο Phong.
3. **calc_normals(...):** Υπολογισμός normals διανυσμάτων επιφάνειας για κάθε κορυφή του πλέγματος.
4. **render_object(...):** Κεντρική συνάρτηση που καλεί τις υπόλοιπες και αποδίδει μία πλήρη εικόνα της σκηνής.
5. **shade_gouraud(...):** Υλοποιεί σκιασμό Gouraud σε τρίγωνα.
6. **shade_phong(...):** Υλοποιεί σκιασμό Phong με παρεμβολή στα διανύσματα και στις UV συντεταγμένες.

1.4 Προσομοιώσεις και Απαιτήσεις

Η εργασία απαιτεί την παραγωγή 8 εικόνων:

1. Τέσσερις εικόνες με χρήση σκιασμού Gouraud:
   - μόνο ambient φωτισμός,
   - μόνο diffuse φωτισμός,
   - μόνο specular φωτισμός,
   - συνδυασμός και των τριών.
2. Τέσσερις αντίστοιχες εικόνες με χρήση σκιασμού Phong.

Επιπλέον, ζητείται να παραχθούν εικόνες με:

- Κάθε σημειακή πηγή φωτός ξεχωριστά.
- Όλες οι πηγές μαζί (combined lighting).

Όλες οι εικόνες θα αποθηκευτούν από το αρχείο `demo.py`, το οποίο πρέπει να τρέχει χωρίς εξωτερικά ορίσματα.

Κεφάλαιο 2

Περιληπτική περιγραφή βοηθητικών συναρτήσεων

2.1 Συνάρτηση γραμμικής παρεμβολής μεταξύ διανυσμάτων

Η `vector_interp(p1, p2, V1, V2, coord, dim)` υλοποιεί γραμμική παρεμβολή μεταξύ δύο διανυσμάτων V_1, V_2 που αντιστοιχούν σε δύο σημεία p_1 και p_2:

- Αν $p_1[\texttt{dim}] = p_2[\texttt{dim}]$, επιστρέφεται το V_1.
- Αλλιώς, υπολογίζεται συντελεστής $t = \frac{coord - p_1}{p_2 - p_1}$ και επιστρέφεται το παρεμβολημένο διάνυσμα $V = (1-t) \cdot V_1 + t \cdot V_2$.

Χρησιμοποιείται εκτενώς σε scanline rendering για την εύρεση τιμών εντός τριγώνου.

2.2 Συστήματα Συντεταγμένων και Προβολές

Παρακάτω περιγράφοται οι συναρτήσεις που χρησιμοποιήθηκαν και σε προηγούμενη εργασία για επεξεργασία σε συστήματα συντεταγμένων και προβολές. Πιο συγκεκριμένα:

- `world2view(pts, R, c0)`: Μετασχηματίζει σημεία από τον παγκόσμιο χώρο στο σύστημα συντεταγμένων της κάμερας με χρήση περιστροφής R και θέσης κάμερας c_0.

  Pseudo-code για `world2view`

  ```
  function world2view(pts, R, c0):
      if pts not 3xN:
          pts = transpose(pts)
      pts_cam = R @ (pts - c0)
      return transpose(pts_cam)
  ```

- `lookat(eye, up, target)`: Δημιουργεί τον πίνακα περιστροφής κάμερας ώστε να κοιτάζει από το σημείο `eye` προς το `target`, με προσανατολισμό το διάνυσμα `up`. Επιστρέφει τον πίνακα περιστροφής R και τη θέση κάμερας.

Pseudo-code για `lookat`

```
function lookat(eye, up, target):
    z = normalize(target - eye)
    x = normalize(cross(z, up))
    y = cross(x, z)
    R = [x, y, -z] as columns
    return R, eye
```

- `perspective_project(pts, focal, R, t)`: Πραγματοποιεί προβολή προοπτικής σημείων βάσει εστιακής απόστασης `focal` και της θέσης/προσανατολισμού κάμερας. Επιστρέφει τις 2D προβολές και τις αποστάσεις βάθους.

Pseudo-code για `perspective_project`

```
function perspective_project(pts, focal, R, t):
    pts_cam = world2view(pts, R, t)
    for each pt in pts_cam:
        x' = focal * pt[0] / pt[2]
        y' = focal * pt[1] / pt[2]
    return [x', y'], depths
```

Οι παραπάνω συναρτήσεις αποτελούν θεμελιώδη συστατικά ενός rendering pipeline, καθώς καθορίζουν τη μετάβαση από την περιγραφή της σκηνής στον παγκόσμιο χώρο (world coordinate system) στο σύστημα της κάμερας και τελικά στην προβολή επί του επιπέδου εικόνας. Η ορθότητα και η συνέπεια στους μετασχηματισμούς αυτούς είναι κρίσιμη για τη ρεαλιστική απεικόνιση της σκηνής υπό προοπτική.

2.3 Συσχέτιση με Επίπεδο Εικόνας

Παρακάτω περιγράφεται η συνάρτηση για μετατροπή σε εικονοστοιχεία που χρησιμοποιήθηκε και υλοποιήθηκε και στην Εργασία 2. Πιο συγκεκριμένα, η συνάρτηση `rasterize(pts_2d, plane_w, plane_h, res_w, res_h)` μετατρέπει τις συντεταγμένες από το επίπεδο της κάμερας (camera plane) σε συντεταγμένες εικονοστοιχείων (pixel coordinates), λαμβάνοντας υπόψη την ανάλυση και τις φυσικές διαστάσεις του αισθητήρα προβολής. Η συνάρτηση αυτή εξασφαλίζει ότι η απεικόνιση του αντικειμένου προβάλλεται σωστά πάνω στο image plane του τελικού render.

Pseudo-code για `rasterize`

```
function rasterize(pts_2d, plane_w, plane_h, res_w, res_h):
    scale_x = res_w / plane_w
    scale_y = res_h / plane_h
    center_x = res_w / 2
    center_y = res_h / 2
    for each (x, y) in pts_2d:
        px = round(x * scale_x + center_x)
        py = round(-y * scale_y + center_y)
        px, py = clamp(px, py)
    return [px, py]
```

Όλες οι παραπάνω συναρτήσεις είναι απαραίτητες για τη δημιουργία ενός πλήρους pipeline απόδοσης που ξεκινά από γεωμετρική αναπαράσταση και καταλήγει σε τελική εικόνα.

Η υλοποίηση των συναρτήσεων που περιέγραψα έχουν υλοποιηθεί στο αρχείο all_funcs_prev.py.

Κεφάλαιο 3

Μοντελοποίηση Φωτισμού και Υλικού Επιφάνειας

3.1 Θεωρητικό Υπόβαθρο

Η παρούσα ενότητα περιγράφει την υλοποίηση φωτισμού επιφανειών βάσει του κλασικού μοντέλου Phong reflection model. Το μοντέλο αυτό αποτελεί μία εμπειρική προσέγγιση της αλληλεπίδρασης φωτός και υλικού, με στόχο τη δημιουργία ρεαλιστικής απόδοσης φωτεινότητας ανά σημείο στην επιφάνεια ενός αντικειμένου.

Ο συνολικός φωτισμός σε ένα σημείο αποτελείται από τρεις επιμέρους συνιστώσες:

- **Ambient:** Αναπαριστά το φως που φτάνει έμμεσα στο σημείο λόγω ανάκλασης από άλλες επιφάνειες.
- **Diffuse:** Προκύπτει από την ευθεία πρόσκρουση του φωτός στο σημείο, εξαρτάται από τη γωνία του φωτισμού ως προς το κάθετο διάνυσμα της επιφάνειας.
- **Specular:** Μοντελοποιεί τη γυαλάδα και είναι ισχυρότερη όσο περισσότερο συμπίπτει η κατεύθυνση του παρατηρητή με την κατεύθυνση ανάκλασης.

3.2 Κλάση `MatPhong`

Για να περιγραφούν τα χαρακτηριστικά του υλικού κάθε επιφάνειας, ορίζεται η κλάση `MatPhong`. Κάθε υλικό προσδιορίζεται από τέσσερις παραμέτρους:

- `ka` : Συντελεστής ambient συνιστώσας.
- `kd` : Συντελεστής diffuse συνιστώσας.
- `ks` : Συντελεστής specular συνιστώσας.
- `n` : Εκθέτης Phong που καθορίζει την εστίαση της λάμψης (όσο μεγαλύτερος τόσο πιο έντονο και περιορισμένο το σπεσυλαρ).

3.3 Συνάρτηση `light`

Η βασική συνάρτηση φωτισμού είναι η `light(...)` και υλοποιεί το μοντέλο Phong ανά σημείο της επιφάνειας. Η συνάρτηση δέχεται ως είσοδο τη θέση ενός σημείου της επιφάνειας, το κανονικό διάνυσμα, το χρώμα κορυφής, τη θέση της κάμερας, τις ιδιότητες του υλικού και τις παραμέτρους φωτισμού της σκηνής (φωτεινές πηγές, εντάσεις, αμβιεντ φωτισμό). Αρχικά, το κανονικό διάνυσμα n κανονικοποιείται. Υπολογίζεται η ambient συνιστώσα ως $I_{amb} = k_a \cdot I_{env} \cdot c$, όπου I_{env} η περιβαλλοντική ένταση και c το τοπικό χρώμα. Έπειτα, για κάθε πηγή φωτός, η οποία μπορεί να είναι μοναδική ή μέλος λίστας, υπολογίζονται:

- Το διάνυσμα φωτός $L = \texttt{normalize}(l_i - pt)$.
- Το διάνυσμα παρατήρησης $V = \texttt{normalize}(cam\_pos - pt)$.
- Το ανακλώμενο διάνυσμα $R = \texttt{normalize}(2(n \cdot L)n - L)$.

Έπειτα υπολογίζονται οι δύο βασικές συνεισφορές:

- **Diffuse:** $I_{diff} = k_d \cdot \max(n \cdot L, 0)$
- **Specular:** $I_{spec} = k_s \cdot \max(R \cdot V, 0)^n$

Το τελικό χρώμα προστίθεται αθροιστικά από κάθε πηγή, με βάση:

$$I = I_{ambient} + \sum_i I_i \cdot \left(c \cdot I_{diff}^{(i)} + I_{spec}^{(i)} \right)$$

Οι τιμές περιορίζονται στο διάστημα $[0, 1]$ με `np.clip` ώστε να είναι έγκυρες τιμές RGB.

Pseudo-code για `light(...)`

```
function light(pt, nrm, vclr, cam_pos, mat, l_pos, l_int, l_amb):
    normalize nrm
    result = mat.ka * l_amb * vclr   # Ambient

    for each light in l_pos:
        L = normalize(light - pt)    # Direction to light
        V = normalize(cam_pos - pt)  # View direction
        R = normalize(2 * dot(nrm, L) * nrm - L)  # Reflection

        diff = mat.kd * max(dot(nrm, L), 0)
        spec = mat.ks * max(dot(R, V), 0) ^ mat.n

        result += l_int * vclr * diff
        result += l_int * spec

    return clamp(result, 0, 1)
```

Η υλοποίηση της συνάρτησης είναι σχεδιασμένη ώστε να υποστηρίζει τόσο μοναδικές όσο και πολλαπλές φωτεινές πηγές μέσω λίστας, και να χειρίζεται ορθά περιπτώσεις μη έγκυρης

γωνίας μέσω του $\max(\cdot, 0)$. Τα διανύσματα `L`, `V` και `R` κανονικοποιούνται ώστε οι γωνιακοί υπολογισμοί να είναι αριθμητικά σταθεροί. Το αποτέλεσμα επιστρέφει το συνολικό φωτισμό στο σημείο και χρησιμοποιείται σε συναρτήσεις όπως `shade_gouraud` και `shade_phong`. Οπτικά, η επίδραση των επιμέρους συνιστωσών ερμηνεύεται ως εξής:

- **Ambient:** εξασφαλίζει ότι ακόμη και σκιερές περιοχές έχουν κάποιο ελάχιστο φωτισμό.
- **Diffuse:** προσδίδει αίσθηση όγκου μέσω γωνίας πρόπτωσης του φωτός.
- **Specular:** προσδίδει γυαλάδα και την ψευδαίσθηση στιλπνότητας.

Η συνάρτηση `light(...)` αποτελεί το βασικό οικοδομικό στοιχείο για όλες τις ρουτίνες σκιασμού που ακολουθούν, καθώς καλείται σε κάθε σημείο (ή κορυφή) για τον υπολογισμό της τελικής φωτεινότητας.

Τα παραπάνω υλοποιούνται σε Python με σχολιασμό στο αρχείο MatPhong.py.

Κεφάλαιο 4

Ανάλυση Μεθόδων Σκιασμού και Object Rendering

Η παρούσα ενότητα παρέχει μία αναλυτική παρουσίαση και τεκμηρίωση των βασικών συναρτήσεων που χρησιμοποιούνται για την απόδοση (rendering) 3D μοντέλων με χρήση φωτισμού τύπου Phong, και τις δύο βασικές τεχνικές σκιασμού: Gouraud και Phong.

4.1 Υπολογισμός Κανονικών Διανυσμάτων: `calc_normals`

Η συνάρτηση `calc_normals` δέχεται ένα σύνολο από σημεία `pts` $\in \mathbb{R}^{3\times N}$ και έναν πίνακα από δείκτες τριγώνων `t_pos_idx` $\in \mathbb{Z}^{3\times M}$.
Για κάθε τρίγωνο με κορυφές $\vec{v}_0, \vec{v}_1, \vec{v}_2$, υπολογίζεται το κανονικό διάνυσμα επιφάνειας μέσω του:

$$\vec{n}_f = (\vec{v}_1 - \vec{v}_0) \times (\vec{v}_2 - \vec{v}_0)$$

Αυτό το διάνυσμα προστίθεται στις τρεις αντίστοιχες κορυφές. Τελικά, όλα τα διανύσματα κανονικοποιούνται ώστε να έχουν μοναδιαίο μέτρο.

Pseudo-code for `calc_normals`

```
function calc_normals(pts, t_pos_idx):
    nrm = zeros_like(pts)
    for each triangle:
        v0, v1, v2 = triangle vertices
        face_n = cross(v1 - v0, v2 - v0)
        add face_n to nrm at each vertex
    normalize all vectors
    return nrm
```

Έτσι, λοιπόν, υπολογίζει ομαλοποιημένα κανονικά διανύσματα για κάθε κορυφή βάσει των επιφανειακών normals των τριγώνων.

4.2 Σκιασμός ανά Κορυφή: `shade_gouraud`

Η συνάρτηση `shade_gouraud` εφαρμόζει την τεχνική Gouraud shading:

- Για κάθε κορυφή του τριγώνου, υπολογίζεται ο φωτισμός με τη συνάρτηση `light`.

- Παρεμβάλλονται οι τιμές χρώματος μεταξύ κορυφών σε κάθε scanline.
- Η υφή διαβάζεται μόνο μία φορά ανά κορυφή.

Pseudo-code for `shade_gouraud`

```
for i = 0 to 2:
    P_cam = v_pos[:, i]
    N_i = v_nrm[:, i]
    uv_i = v_uvs[:, i]
    texel = tex[uv_i]
    color[i] = light(P_cam, N_i, texel, ...)

for each scanline:
    compute A, B positions on scanline
    interpolate color between A and B
    for each pixel x between A and B:
        img[y, x] = interpolated color
```

Σε αυτή την συνάρτηση δηλαδή, υπολογίζεται ο φωτισμός στις κορυφές και παρεμβάλει το χρώμα ανά pixel για γρήγορη απόδοση με μειωμένη ακρίβεια.

4.3 Σκιασμός ανά Pixel: shade_phong

Η συνάρτηση `shade_phong` εφαρμόζει τον Phong shading ανά pixel:

- Παρεμβάλλονται κανονικά διανύσματα, υφές και θέσεις ανά pixel.
- Καλείται η `light(...)` για κάθε pixel ξεχωριστά.
- Δίνει ομαλό και ρεαλιστικό φωτισμό ακόμη και σε curved επιφάνειες.

Pseudo-code for `shade_phong`

```
for each scanline:
    compute A, B screen positions
    interpolate normals, UVs, and 3D positions from A to B
    for each pixel x between A and B:
        texel = tex[UV]
        color = light(P, N, texel, ...)
        img[y, x] = color
```

Η συνάρτηση αυτή συνοπτικά υπολογίζει φωτισμό για κάθε pixel, επιτυγχάνοντας πολύ ρεαλιστική απόδοση με υψηλό υπολογιστικό κόστος.

4.4 Απόδοση Αντικειμένου: `render_object`

Η συνάρτηση `render_object` είναι η βασική ρουτίνα απόδοσης (rendering pipeline) και εκτελεί τα εξής:

1. Υπολογίζει κανονικά μέσω `calc_normals`.
2. Υπολογίζει μετασχηματισμούς κάμερας μέσω `lookat` και μετασχηματίζει τις κορυφές στο χώρο της κάμερας.
3. Υλοποιεί προοπτική προβολή μέσω `perspective_project`.
4. Μετατρέπει συντεταγμένες σε pixels με `rasterize`.
5. Υπολογίζει το βάθος κάθε τριγώνου και τα σχεδιάζει με σειρά από το πίσω προς το μπροστά (depth sorting).
6. Καλεί τη συνάρτηση σκιασμού `shade_gouraud` ή `shade_phong` ανά τρίγωνο.

Pseudo-code for `render_object`

```
normals = calc_normals(v_pos, t_pos_idx)
R, t = lookat(eye, up, target)
v_cam = R @ v_pos + t
proj2d, depth = perspective_project(v_cam)
pix = rasterize(proj2d)
triangles = sort_by_depth(depth)

for each triangle in triangles:
    if shader == 'gouraud':
        shade_gouraud(...)
    else if shader == 'phong':
        shade_phong(...)
```

Η συνάρτηση αυτή πρακτικά ολοκληρώνει όλη τη ροή από υπολογισμό normals μέχρι την τελική απόδοση του αντικειμένου στην εικόνα.

Οι βασικές αυτές συναρτήσεις υλοποιούνται στο αρχείο all_funcs_new.py.

Κεφάλαιο 5

Κλήση Συναρτήσεων, Αποτελέσματα και Παρατηρήσεις

Η παρούσα ενότητα περιγράφει τη διαδικασία κλήσης της συνάρτησης render_object μέσα από το αρχείο `demo.py`. Ακολουθεί η απόδοση εικόνων με διαφορετικούς φωτισμούς και τεχνικές σκιασμού.

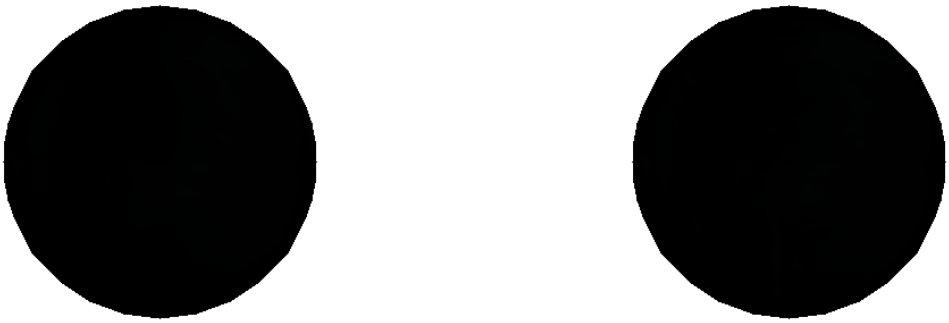
5.1 Διαδικασία Εκτέλεσης

Το πρόγραμμα:

- Φορτώνει γεωμετρικά δεδομένα (θέσεις κορυφών v_pos , UVs, τρίγωνα t_pos_idx).
- Ορίζει θέση κάμερας eye, διεύθυνση στόχευσης target και διάνυσμα up .
- Φορτώνει υφή από αρχείο ή χρησιμοποιεί λευκή υφή.
- Καλεί την render_object για κάθε συνδυασμό φωτισμού και σκιασμού.
- Αποθηκεύει τις εικόνες στον φάκελο `results/` που δημιουργείται.

5.2 Αποτελέσματα ανά Τεχνική Σκιασμού και Μοντέλο Φωτισμού

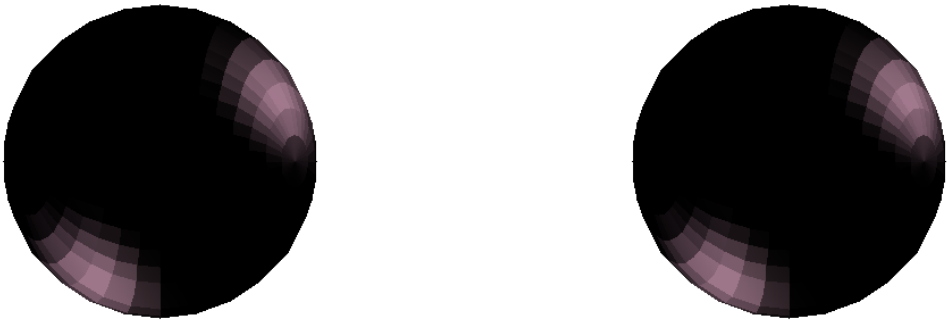
Στην παρούσα ενότητα παρουσιάζονται τα αποτελέσματα απόδοσης του αντικειμένου για διαφορετικές τεχνικές σκιασμού και συνιστώσες του μοντέλου φωτισμού, με στόχο την οπτική σύγκριση και εξαγωγή συμπερασμάτων.



Σχήμα 5.1: Gouraud Ambient (left) και Phong Ambient (right)

Σχήμα 5.2: Gouraud Ambient (ka = 1) (left) και Phong Ambient (ka = 1) (right)

Σχήμα 5.3: Gouraud Diffuse (left) και Phong Diffuse (right)



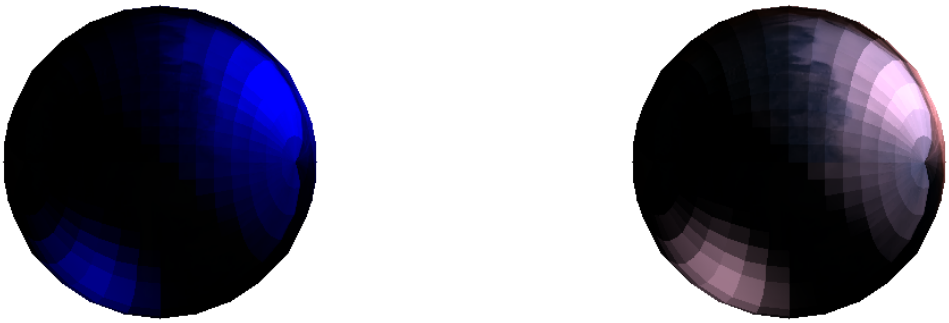
Σχήμα 5.4: Gouraud Specular (left) και Phong Specular (right)

Σχήμα 5.5: Gouraud All (left) και Phong All (right)

5.3 Απόδοση Μεμονωμένων Πηγών Φωτός (Phong Only)

Σε αυτή την ενότητα εξετάζουμε την επίδραση κάθε μεμονωμένης πηγής φωτός ξεχωριστά, χρησιμοποιώντας αποκλειστικά Phong shading, ώστε να αναδείξουμε τον ρόλο και τη συμβολή της καθεμιάς στη συνολική φωτεινότητα και εμφάνιση του αντικειμένου.

Σχήμα 5.6: Phong Light 1 (left) και Phong Light 2 (right)



Σχήμα 5.7: Phong Light 3 (left) και Phong All Lights (right)

5.4 Συμπεράσματα και Παρατηρήσεις

Η παρούσα εργασία ανέδειξε τη σημασία της επιλογής κατάλληλου μοντέλου σκιασμού και ρύθμισης παραμέτρων φωτισμού για ρεαλιστική απόδοση 3D αντικειμένων.

- Το Gouraud shading είναι υπολογιστικά αποδοτικό, καθώς ο φωτισμός υπολογίζεται μόνο στις κορυφές και παρεμβάλλεται στο εσωτερικό των τριγώνων. Ωστόσο, αυτό έχει ως αποτέλεσμα την απώλεια λεπτομερειών, ειδικά στα specular highlights, τα οποία μπορεί να «εξαφανιστούν» αν δεν βρίσκονται ακριβώς σε κορυφή. Το τελικό αποτέλεσμα ενδέχεται να φαίνεται πιο θολό ή παραμορφωμένο (blurred, distorted).

- Αντίθετα, το Phong shading , το οποίο υπολογίζει φωτισμό ανά pixel, προσφέρει πολύ πιο ρεαλιστική εμφάνιση, διατηρώντας λεπτομέρειες και εξομαλύνοντας τις μεταβάσεις φωτεινότητας. Η ακρίβεια στον φωτισμό είναι αισθητά βελτιωμένη, ειδικά σε καμπύλες ή λεπτομερείς επιφάνειες.

- Η ambient συνιστώσα είναι υπεύθυνη για την ομοιόμορφη βασική φωτεινότητα σε ολόκληρη την επιφάνεια του αντικειμένου. Η diffuse αντιπροσωπεύει την διάχυτη ανταν άκλαση φωτός και σχετίζεται με τη γωνία πρόσπτωσης του φωτός στην επιφάνεια. Η specular είναι υπεύθυνη για τις γυαλιστερές λάμψεις και εξαρτάται από τη γωνία ανάκλασης σε σχέση με τον παρατηρητή.

- Ο πλήρης φωτισμός δηλαδή ο συνδυασμός των ambient, diffuse και specular συνιστωσών σε συνδυασμό με την παρουσία πολλαπλών πηγών φωτός οδηγεί σε ρεαλιστικότερα και πιο αισθητικά αποτελέσματα. Ιδιαίτερα σε Phong shading, αυτές οι πηγές φωτός αλληλεπιδρούν σωστά με την καμπυλότητα και υφή της επιφάνειας.

- Επιπλέον, κατά μια πειραματική εφαρμογή που εφάσμοσα, παρατηρήθηκε ότι η πολύ μικρή τιμή της παραμέτρου ka = 0.01 (συντελεστής ambient φωτισμού) στο αρχείο hw3.npy είχε ως αποτέλεσμα σκοτεινές απεικονίσεις όταν χρησιμοποιούνταν μόνο

ambient φωτισμός. Αυτό αναδεικνύει τη σημασία σωστού καλιμπραρίσματος των φωτιστικών παραμέτρων ανάλογα με την υφή, το μέγεθος και την κάμερα του σκηνικού.

Τελική Παρατήρηση: Ο συνδυασμός Phong shading με πλήρεις φωτιστικές συνιστώσες και πολλαπλές πηγές φωτός προσφέρει το καλύτερο δυνατό αποτέλεσμα για ρεαλιστική 3D απόδοση. Η υπεροχή του είναι εμφανής στα διαγράμματα, όπου τα χαρακτηριστικά του αντικειμένου αποδίδονται με υψηλή πιστότητα και φυσικότητα.

Βιβλιογραφία

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

[2] https://imageio.readthedocs.io/en/stable/