Γραφική με Υπολογιστές

Εργασία #3

Κωτούλας Εμμανουήλ 9697

Το demo.py διαβάζει το αρχείο h3.npy, που πρέπει να βρίσκεται στο ίδιο directory με το demo.py, και προβάλλει τα τρίγωνα που δίνονται με τρισδιάστατες συντεταγμένες, βρίσκει το κατάλληλο χρώμα σύμφωνα με τον φωτισμό και τα εμφανίζει σε μια εικόνα 512 επί 512 pixel. Δεν χρείαζεται κάποιο εξωτερικό όρισμα και τρέχει με την εντολή python3.7 demo.py

Περιγραφή συναρτήσεων

- Ambient_light(k_a, I_a) :

Η συνάρτηση αυτή υπολογίζει την συνεισφορά του διάχυτου φωτός στο φωτισμό ενός σημείου του αντικειμένου. Εκτελεί έναν απλό πολλαπλασιασμό.

- Diffuse_light(P, N, color, k_d, light_poisitions, light_intensities):

Η συνάρτηση αυτή υπολογίζει την συνεισφορά της διάχυτης ανάκλασης στον φωτισμό ενός σημείου του αντικειμένου. Υπολογίζει το διάνυσμα L μοναδιαίο έτσι όπως αναφέρεται στην θεωρία και εκτελεί τις πράξεις όπως παρουσιάζονται στην σελίδα 97 των σημειώσεων.

- Specular_light(P, N, color, cam_pos, k_s, n, light_positions, light_intensities):

Η συνάρτηση αυτή υπολογίζει την συνεισφορά της κατοπτρικής ανάκλασης στον φωτισμό ενός σημείου του αντικειμένου. Υπολογίζει το L μοναδιαίο και το V μοναδιαίο όπως αναφέρονται στην θεωρία και εκτελεί τις πράξεις όπως αναφέρονται στην θεωρία στην σελίδα 99 των σημειώσεων.

- Calculate_normals(vertices, face_indices):

Η συνάρτηση αυτή υπολογίζει τα κανονικά διανύσματα για όλα τα σημεία. Για να επιτευχθεί αυτό αρχικά υπολογίζει τα κανονικά διανύσματα για όλα τα τρίγωνα και έπειτα κάνει average όλα τα κανονικά διανύσματα στα οποία ανήκει το σημείο αυτό. Για τον αρχικό υπολογισμό των κανονικών διανυσμάτων πραγματοποιείται εξωτερικό γινόμενο ανάμεσα στα διανύσματα των πλευρών.

- Render_object(shader, focal, eye, lookat, up, bg_color, M, N, H, W, verts, vert_colors, face_indices, k_a, k_d, k_s, n, light_position, light_intensities, I_a):

Η συνάρτηση αυτή καλεί τις υπόλοιπες συναρτήσεις που υλοποιήθηκαν στις τρεις εργασίες. Αρχικά υπολογίζει τα κανονικά διανύσματα, έπειτα βρίσκει τα κέντρα βάρους των τριγώνων που θα χρειαστούν για τον υπολογισμό του φωτισμού. Στην συνέχεια προβάλει τα τρίγωνα και τα κάνει rasterize. Τέλος καλεί την render της πρώτης εργασίας για το γέμισμα των τριγώνων.

- Shade_gouraud(verts_p, verts_n, verts_c, bcoords, cam_pos, k_a, k_d, k_s, n, lights_positions, light_intensities, I_a, X):

Η συνάρτηση αυτή είναι η ίδια με την συνάρτηση που υλοποίησα για την πρώτη εργασία με την διαφορά ότι το χρώμα υπολογίζεται καλώντας τις συναρτήσεις που υλοποιήθηκαν (ambient_light, diffuse_light και specular_light).

- Shade_phong(verts_p, verts_n, verts_c, bcoords, cam_pos, k_a, k_d, k_s, n, lights_positions, light_intensities, I_a, X):

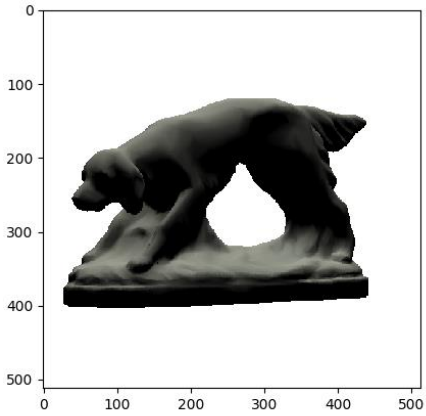
Η συνάρτηση αυτή είναι η ίδια με την gouraud με την διαφορά ότι στην shade_phong για να υπολογιστεί ο φωτισμός σε κάθε σημείο πρώτα κάνουμε γραμμική παρεμβολή στα κανονικά διανύσματα των σημείων όπως κάναμε και στην gouraud για τα χρώματα, και έπειτα υπολογίζουμε τον φωτισμό ξεχωριστά για κάθε σημείο.

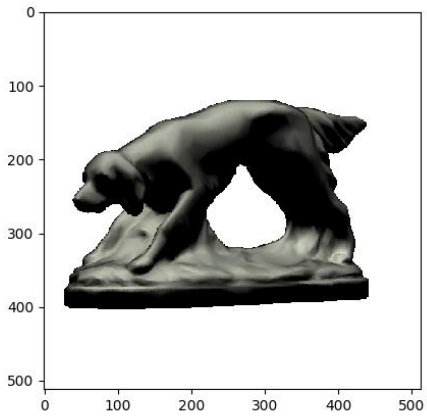
Στις συναρτήσεις render_object, shade_gouraud και shade_phong προστέθηκε μια μεταβλητή lights η οποία μεταφέρει την πληροφορία για το ποια είδη φωτισμού θα χρησιμοποιηθούν καθώς ζητείται να παραχθεί μια εικόνα για κάθε είδος φωτισμού και μια με όλα τα είδη φωτισμού.

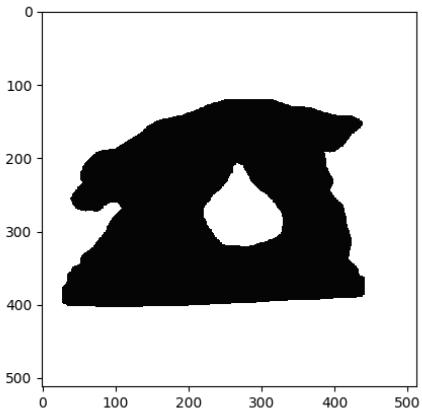
Επίσης το αρχείο που δίνεται ονομάζεται h3.npy και όχι hw3.npy όπως αναφέρετε στις σημειώσεις.

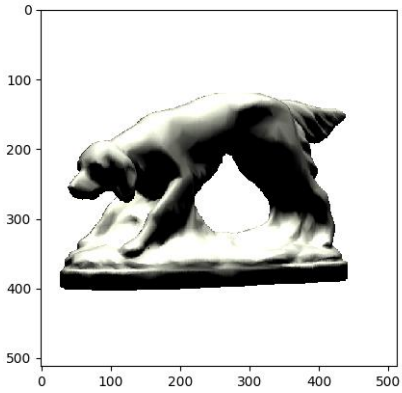
Αποτελέσματα

Gouraud:

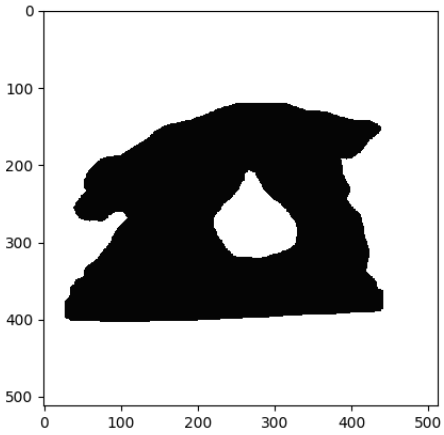


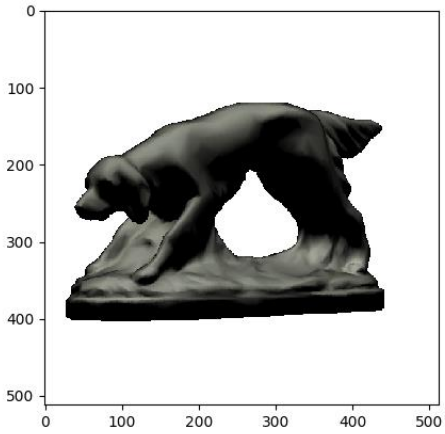


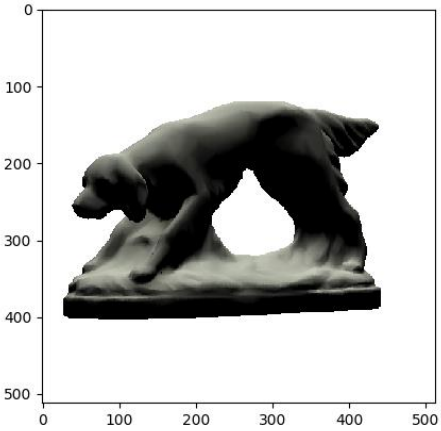


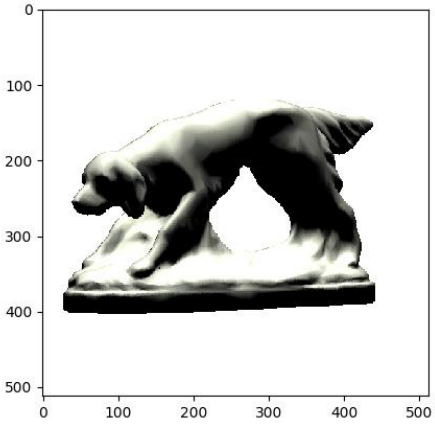


Phong:


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500