Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Ηλεκτρονικής και Υπολογιστών

ΓΡΑΦΙΚΗ ΜΕ ΥΠΟΛΟΓΙΣΤΕΣ

Εργασία 3η : Θέαση και Φωτισμός

Εαρινό εξάμηνο 2023/24

Δεϊρμεντζόγλου Ιωάννης
Τομέας Ηλεκτρονικής και Υπολογιστών
Α.Ε.Μ.: 10015
Email: deirmentz@ece.auth.gr

Περιεχόμενα

Εισαγωγή

Αντικείμενο της παρούσας εργασίας ήταν να αξιοποιηθούν συναρτήσεις των προηγουμένων με κάποιες αλλαγές και η δημιουργία κάποιων που αφορούν τον φωτισμό προκειμένου να δημιουργηθεί ένα πλήρες πλαίσιο δημιουργίας φωτογραφιών μίας εικονικής σκηνής. Πιο συγκεκριμένα, στην προηγούμενη εργασία δημιουργήθηκε το στάδιο της προβολής τρισδιάστατων σκηνικών στο πέτασμα της κάμερας και ύστερα στην οθόνη του υπολογιστή. Σε αυτή την εργασία σκοπός είναι να συμπεριληφθούν αλγόριθμοι που σχετίζονται με την θέαση του αντικειμένου ως προς μια ή περισσότερες πηγές φωτός για την προβολή του αντικειμένου.

Α. Φωτισμός και υλικό επιφάνειας

Αρχικά, ζητείται η συνάρτηση , η οποία υπολογίζει το φωτισμό ενός σημείου *point*, το οποίο ανήκει σε μία επιφάνεια με υλικό τύπου- Phong λόγω του διάχυτου φωτισμού από το περιβάλλον, διάχυτης ανάκλασης, και κατοπτρικής ανάκλασης.

- `I = light(point, normal, vcolor, cam_pos, ka, kd, ks, n, lpos, lint, I_a, lightOption)`

  Για τον υπολογισμό την ένταση της τριχρωματικής ακτινοβολίας $I = [Ir, Ig, Ib]T$, που ανακλάται από το σημείο P , οι υπολογισμοί βασίστηκαν στο κεφάλαιο 8 των σημειώσεων του μαθήματος.

  Η συνάρτηση επιστρέφει ένα διάνυσμα 1×3 που περιέχει την τριχρωματική ένταση ακτινοβολίας που ανακλάται από το σημείο. Η λειτουργία της συνάρτησης περιλαμβάνει τα εξής βήματα: Αρχικά, υπολογίζεται το διάχυτο φως από το περιβάλλον: με τον πολλαπλασιασμό του συντελεστή ka με την ένταση του περιβαλλοντικού φωτός I_a, η οποία βρίσκεται στο npy αρχείο της εκφώνησης και προστέθηκε ως όρισμα στην συναρτηση. Οι συνιστώσες διάχυτου και κατοπτρικού φωτός αρχικοποιούνται σε μηδενικά διανύσματα.

  Για μια πηγή φωτός υπολογίζεται το διάνυσμα από το σημείο προς την πηγή φωτός **SP = lpos - point** και κανονικοποιείται για να δώσει το μοναδιαιο διάνυσμα **L**. Έπειτα, υπολογίζεται το συνημίτονο της γωνίας μεταξύ του κανονικού διανύσματος και του διανύσματος του φωτός cosa και υπολογίζεται η διάχυτη αντανάκλαση πολλαπλασιάζοντας την ένταση του φωτός με τον συντελεστή διάχυτης αντανάκλασης και το **cosa**. Στην συνέχεια, υπολογίζονται το διάνυσμα από το σημείο προς την κάμερα **CP = cam_pos - point** και κανονικοποιείται για να δώσει την κατεύθυνση της θέασης και το διάνυσμα της αντανάκλασης του φωτός και το συνημίτονο cosb_a της γωνίας μεταξύ αυτού και του διανύσματος της θέασης. Έτσι, υπολογίζεται η κατοπτρική αντανάκλαση

πολλαπλασιάζοντας την ένταση του φωτός με τον συντελεστή κατοπτρικής αντανάκλασης και το cosb_a υψωμένο στην σταθερά Phong.

Για τον υπολογισμό του φωτισμού για πολλές πηγές φωτός ακολουθείται παρόμοια διαδικασία για καθε πηγή ξεχωριστά όπως παραπάνω και προστίθενται τα αποτελέσματα στα συνολικά διανύσματα διάχυτης και κατοπτρικής αντανάκλασης.

Τέλος για τον συνδυασμό των συνιστωσών φωτισμού προστέθηκε ακόμα μια βοηθητική μεταβλητή ως όρισμα στην συναρτηση. Ανάλογα με την επιλογή lightOption, συνδυάζονται οι συνιστώσες περιβαλλοντικού, διάχυτου και κατοπτρικού φωτός για να προκύψει η συνολική ένταση φωτισμού:

- **'Combined':** Συνδυάζονται όλες οι συνιστώσες.
- **'Ambient'**: Χρησιμοποιείται μόνο η περιβαλλοντική συνιστώσα.
- **'Diffusion'**: Χρησιμοποιείται μόνο η διάχυτη συνιστώσα.
- **'Specular'**: Χρησιμοποιείται μόνο η κατοπτρική συνιστώσα.

Σε περίπτωση λανθασμένης επιλογής, η συνάρτηση επιστρέφει None.

Τέλος, η συνολική ένταση φωτισμού περιορίζεται έτσι ώστε οι τιμές να βρίσκονται στο εύρος μεταξύ 0 και 1 και επιστρέφεται ως αποτέλεσμα της συνάρτησης. Η συναρτηση βρίσκεται στο αρχείο **IlluminationCalculationA.py** .

B.1 Υπολογισμός κανονικών διανυσμάτων επιφάνειας

Το επόμενο βήμα που πρέπει να υλοποιηθεί είναι ο υπολογισμός των συντεταγμένων των κάθετων διανυσμάτων σε κάθε σημείο (κορυφή) της επιφάνειας που ορίζει το αντικείμενο. Το 3Δ αντικείμενο αποτελείται από N_T τρίγωνα.

- **normals = calculate_normals(verts, faces)**

  Η συνάρτηση calculate_normals υπολογίζει τα κανονικά διανύσματα για κάθε κορυφή σε ένα τρίγωνο τα οποία υπολογίζονται σύμφωνα με τον κανόνα του δεξιόστροφου κοχλία από τις κορυφές του κάθε τριγώνου. Συγκεκριμένα, οι ονομαζόμενες κορυφές A,B,C δημιουργούν τα πλευρικά κανονικοποιημένα διανύσματα AB,BC και από το εξωτερικό γινόμενο αυτών $AB \times BC$ υπολογίζεται το κανονικό διάνυσμα του κάθε τριγώνου.

  Οι παράμετροι της συνάρτησης είναι: ο πίνακας **verts 3 × Nv** με τις συντεταγμένες των κορυφών του αντικειμένου, ο πίνακας **faces 3×N_T** που περιγράφει τα τρίγωνα. Η k-οστή στήλη του faces περιέχει τους αύξοντες αριθμούς των κορυφών του k-οστού τριγώνου του αντικειμένου, $1 \le k \le N_T$

  Η συναρτηση επιστρέφει έναν πίνακα normals με τα κανονικοποιημένα κανονικά διανύσματα για κάθε κορυφή.

Για κάθε τρίγωνο στη λίστα faces, εξάγονται οι δείκτες των κορυφών που το συνθέτουν. Οι κορυφές του τριγώνου A, B και C εξάγονται από τους δείκτες και υπολογίζονται τα διανύσματα ακμής AB και AC, που κανονικοποιούνται για να δώσουν τα κανονικοποιημένα διανύσματα nAB και nAC. Το κανονικό διάνυσμα του τριγώνου υπολογίζεται με το εξωτερικό γινόμενο των nAB και nAC. Έπειτα, το υπολογισμένο κανονικό διάνυσμα προστίθεται στα κανονικά διανύσματα των κορυφών που συνθέτουν το τρίγωνο και κανονικοποιούνται ώστε να έχουν μήκος 1. Πιο συγκεκριμένα, κάθε τέτοιο διάνυσμα αθροίζεται στα άλλα του ίδιου τριγώνου $N(\mathrm{t}riangleIndex)=\Sigma_i^{faces}(AB\times BC)_i$, ώστε μετά από την άθροιση να βρούμε το επιστρεφόμενο ζητούμενο κανονικοποιώντας το διάνυσμα $N(\mathrm{t}riangleIndex)$ για κάθε τρίγωνο. Ο πίνακας normals επιστρέφεται ως αποτέλεσμα της συνάρτησης. . Η συναρτηση βρίσκεται στο αρχείο **calcNormalsB1.py**

B2. Συνάρτηση φωτογράφισης

Το επόμενο βήμα που πρέπει να υλοποιηθεί είναι η συνάρτηση φωτογράφισης, η οποία δημιουργεί την έγχρωμη φωτογραφία *img* ενός 3D αντικειμένου υπολογίζοντας το χρώμα με βάση τα μοντέλα φωτισμού της ενότητας Α.

- ***img = render_object(shader, focal, eye, lookat, up, bg_color, M,N,H,W, verts, vert_colors, faces, ka, kd, ks, n, lpos, lint, lamb)***

Η συναρτηση αυτή είναι παρόμοιας λογικής με αυτήν που υλοποιήθηκε στις προηγούμενες εργασίες. Η συναρτηση υλοποιεί την τελική απεικόνιση του αντικειμένου βάσει των δοσμένων δεδομένων αυτού. Συγκεκριμένα, καλεί την συνάρτηση προοπτικής κάμερας με στόχο **lookat()** ώστε να υπολογίσει τον πίνακα περιστροφής R και το διάνυσμα μετατόπισης t και στην συνέχεια η **perspective_project** για τις συντεταγμένες των 2D σημείων πάνω στο πέτασμα της εικονικής κάμερας. Στην συνέχεια καλείται η συνάρτηση απεικόνισης **rasterize()** ώστε να μετατρέψει τα σημεία αυτά σε pixel της εικόνας. Επιπλέον, καλείται η calculate_normals που αναφέρθηκε παραπάνω, η οποία υπολογίζει τα κανονικά διανύσματα των κορυφών του αντικειμένου.
Η συνάρτηση επίσης δημιουργεί έναν καμβά με διαστάσεις MxN και γεμίζει κάθε pixel με το χρώμα φόντου bg_color. Ο καμβάς αυτός θα χρησιμοποιηθεί για την απόδοση της τελικής εικόνας.

Στην συνέχεια, ανάλογα με την παράμετρο shader που καθορίζει τον τύπο σκίασης που θα χρησιμοποιηθεί (**'gouraud'** ή '**phong**'), η συνάρτηση εφαρμόζει την αντίστοιχη σκίαση σε κάθε τρίγωνο του αντικειμένου καλώντας είτε την **shade_gouraud()** είτε την **shade_phong()**. Για κάθε πρόσωπο (τρίγωνο) στη λίστα facesSorted, ανακτώνται οι δείκτες των κορυφών του τριγώνου από τη λίστα faces. Με βάση τους δείκτες των κορυφών που ανακτήθηκαν, εξάγονται τα χρώματα των κορυφών του τριγώνου από τη λίστα vert_colors.

Αυτά τα χρώματα θα χρησιμοποιηθούν για τον υπολογισμό της σκίασης. Με βάση τους δείκτες των κορυφών που ανακτήθηκαν, εξάγονται οι 2D συντεταγμένες των κορυφών του τριγώνου από τη λίστα verts2d. Αυτές οι συντεταγμένες έχουν προκύψει από την προβολή των 3D κορυφών στην 2D επιφάνεια της οθόνης. Το κέντρο βάρους **bcoords** του τριγώνου υπολογίζεται ως ο μέσος όρος των συντεταγμένων των τριών κορυφών του τριγώνου και είναι είσοδος στις συναρτήσεις σκίασης που αναλύονται παρακάτω.

Η τελική εικόνα επιστρέφεται ως αποτέλεσμα της συνάρτησης. Η εικόνα αυτή περιέχει την απόδοση του τρισδιάστατου αντικειμένου με βάση τη σκίαση που εφαρμόστηκε και όλες τις ρυθμίσεις και παραμέτρους που καθορίστηκαν.

Σημειώνεται ότι στην συγκεκριμένη συναρτηση καλούνται οι 2 επόμενες που θα αναλυθούν παρακάτω όπως επίσης και οι συναρτήσεις που αναφέρθηκαν, αυτούσιες από την 2η εργασία. Παρατίθενται ωστόσο και σε αυτό το αρχείο για λογούς πληρότητας. Ο κώδικας βρίσκεται στο αρχείο **renderObjectB2.py**

B3. Gouraud Shading

Το επόμενο βήμα που πρέπει να υλοποιηθεί είναι η συναρτηση shade_gouraud() , η οποία υπολογίζει το χρώμα στις κορυφές του δοθέντος τριγώνου με βάση το πλήρες μοντέλο φωτισμού (χρησιμοποιώντας δηλαδή τις συναρτήσεις της ενότητας Α) και στη συνέχεια χρησιμοποιεί γραμμική παρεμβολή χρώματος για την εύρεση του χρώματος στα εσωτερικά σημεία του τριγώνου, με την **vector_interp()** της εργασίας #1.

- **shade_gouraud(vertsp, vertsn, vertsc, bcoords, cam_pos, ka, kd, ks, n, lpos, lint, lamb,X, lightOption)**

  Σημειώνεται ότι στην συγκεκριμένη συναρτηση έχει προστεθεί η μεταβλητή lightOption που παρουσιάστηκε παραπάνω. Ουσιαστικά η συγκεκριμένη συναρτηση διαφέρει ελάχιστα από τις προηγούμενες εργασίες και πληρώνει το τρίγωνο. Πλέον λαμβάνεται υπόψιν ο υπολογισμός του χρώματος βάσει των μοντέλων φωτισμού και για τον υπολογισμό της τριχρωματικης ακτινοβολίας καλείται η συναρτηση light της ενότητας Α. Συγκεκριμένα, για κάθε κορυφή του τριγώνου υπολογίζεται η ένταση χρώματος βάσει των προαναφερθέντων μοντέλων φωτισμού οπου το σημείο point αντιστοιχεί στο βαρυκεντρο του τριγώνου με συντεταγμένες bcoords. Τέλος, ακολουθείται ο γνωστός αλγόριθμος χρωματισμού του Gouraud βάσει των νέων τριών χρωμάτων των κορυφών μέσω της κλήσης της συνάρτησης g_shading από τις προηγούμενες εργασίες. Τέλος, η σκιασμένη εικόνα Υ επιστρέφεται ως αποτέλεσμα της συνάρτησης.

  Σημειώνεται πως ο κώδικας για τις παραπάνω συναρτήσεις βρίσκεται στο αρχείο GouraudPhongShading.py . Επίσης, επισυνάπτεται το αρχείο bresenham.py από την εργασία 1 για λόγους πληρότητας.

B4. Phong Shading

Το επόμενο ζήτημα της εργασίας είναι η υλοποίηση της συνάρτησης shade_phong() η οποία υπολογίζει το χρώμα των σημείων του τριγώνου πραγματοποιώντας παρεμβολή τόσο στα κανονικά διανύσματα όσο και στα χρώματα των κορυφών.

Αρχικά, ορίζονται 2 βοηθητικές συναρτήσεις , οι οποίες θα χρησιμοποιηθούν μέσα στην shade_phong() και ουσιαστικά σκοπός τους είναι η παρεμβολή στα κανονικά διανύσματα και στα διανύσματα των κορυφών. Πιο συγκεκριμένα ορίζεται η συναρτηση :

- **interpolate_normal(x1, x2, x, N1, N2)**

  Η συνάρτηση interpolate_normal υπολογίζει το normal vector σε ένα σημείο x βάσει των κανονικών διανυσμάτων σε δύο άλλα σημεία, x1 και x2. Χρησιμοποιεί γραμμική παρεμβολή για να βρει το κανονικό διάνυσμα στο σημείο x. Αρχικά, γίνεται έλεγχος της περίπτωσης τα 2 σημεία να είναι πολύ κοντά και ουσιαστικά N1 και N2 να ταυτίζονται. Σε αυτήν την περιπτωση επιστρέφεται το N1. Εάν δεν ισχύει τότε κατά τα γνωστά ακολουθείται η παρακάτω διαδικασία γραμμικής παρεμβολής:

```
lambdaP = (x2 - x) / (x2 - x1)
N = lambdaP * N1 + (1 - lambdaP) * N2
```

  Το κανονικό διάνυσμα N που προκύπτει από την παρεμβολή κανονικοποιείται και αποτελεί την έξοδο της συνάρτησης.

- **interpolate_color(x1, x2, x, color1, color2)**

  Η συναρτηση αυτή έχει ακριβώς την ιδιά λειτουργία με την προηγουμένη μόνο που σε αυτήν την περιπτωση η γραμμική παρεμβολή εφαρμόζεται για τον υπολογισμό του χρώματος στο σημείο x.

- **shade_phong(vertsp, vertsn, vertsc, bcoords, campos, ka, kd, ks, n, lpos, lint, lamb,X, lightOption)**

  Η συναρτηση αυτή έχει βασικά στοιχεία της πλήρωσης τριγώνων με Gouraud αλλά στην προκειμένη περίπτωση ο αλγόριθμος του Phong αντί για μόνο παρεμβολή μεταξύ των σημείων για τον υπολογισμό του χρώματος, πρέπει να γίνει παρεμβολή για τον υπολογισμό του φωτισμού τόσο στα κανονικά διανύσματα αλλά και στα χρώματα των κορυφών του τρίγωνου. Πιο συγκεκριμένα, για το κάθε τρίγωνο, η συναρτηση θα υπολογίζει τα κανονικά διανύσματα των αρχικών σημείων (δηλαδή, πριν την προβολή) κατά μήκος των ενεργών πλευρών εκτελώντας γραμμική παρεμβολή στα κανονικά διανύσματα των κορυφών της πλευράς. Για κάθε εσωτερικό σημείο, θα υπολογίζει το κανονικό διάνυσμα κατά μήκος του scanline εκτελώντας γραμμική παρεμβολή στα κανονικά διανύσματα που αντιστοιχούν στα ενεργά σημεία της πλευράς. Παρόμοια διαδικασία θα πραγματοποιείται και για τα χρώματα των σημείων. Έχοντας υπολογίσει το κανονικό διάνυσμα και το

χρώμα για ένα σημείο, το χρώμα του θα προκύπτει χρησιμοποιώντας τη συνάρτηση *light*(). Ουσιαστικά, η συναρτηση είναι παρόμοια με την shade_gouraud και αυτό που αλλάζει είναι ο υπολογισμός του χρώματος. Αντί να υπολογίζεται το χρώμα σημείων (ενεργές πλευρές και ενεργά σημεία) με την vector_interp γίνεται γραμμική παρεμβολή για να βρεθεί το κανονικό του διάνυσμα με την βοήθεια της **interpolate_normal()** και στην συνέχεια καλείται η **interpolate_color()** για το χρώμα και η το σημείο της εικόνας φωτίζεται με κλήση της **light().** Παρακατω παρατίθεται ένα ενδεικτικό παράδειγμα της κλήσης των συναρτήσεων:

```
Για κάποιο ενεργό σημείο :
     interpN = interpolate_normal(κορυφές[0][0], κορυφές[1][0],
     σημείο[0], vertsn[0], vertsn[1])

     χρώμα_κορυφής = interpolate_color ( κορυφές[0][0],
     κορυφές[1][0], σημείο[0], λίστα(vertsc[0]), λίστα(vertsc[1]))

     εικόνα[σημείο[1]][σημείο[0]] = light (bcoords, interpN,
     χρώμα_κορυφής, θέση_κάμερας, ka, kd, ks, n, lpos, lint,
     lamb, επιλογή_φωτός)
```

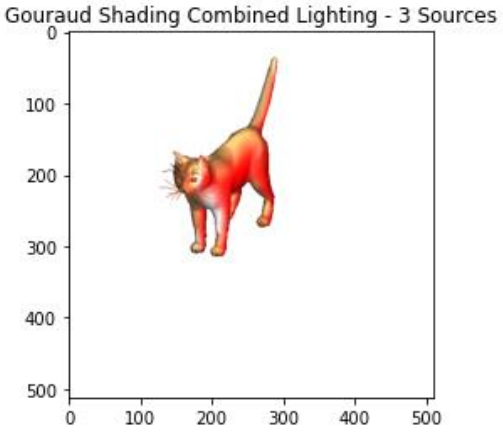
Τέλος, η σκιασμένη εικόνα Y επιστρέφεται ως αποτέλεσμα της συνάρτησης. Σημειώνεται πως ο κώδικας για τις παραπάνω συναρτήσεις βρίσκεται στο αρχείο GouraudPhongShading.py .

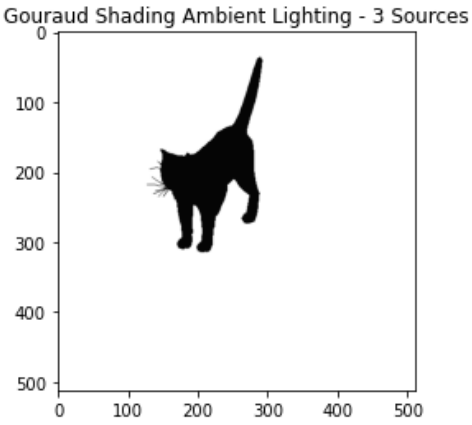
Demo

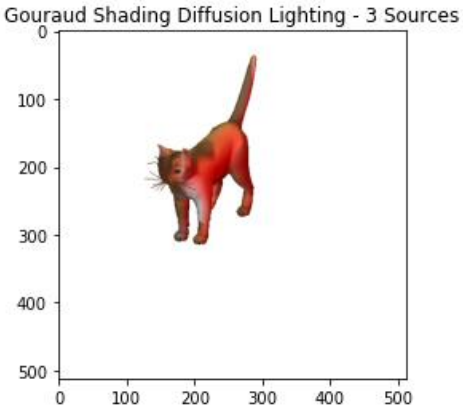
Παρακατω παρατίθενται αποτελέσματα για διάφορες περιπτώσεις φωτισμού και αριθμού πηγών. Συγκεκριμένα παρατίθενται αποτελέσματα και με τις 3 πήγες φωτισμού όπως δίνονται από την εκφώνηση αλλά και για κάθε μια ξεχωριστά τόσο με phong shading αλλά και gouraud. Ο κώδικας βρίσκεται στο αρχείο demo.py .

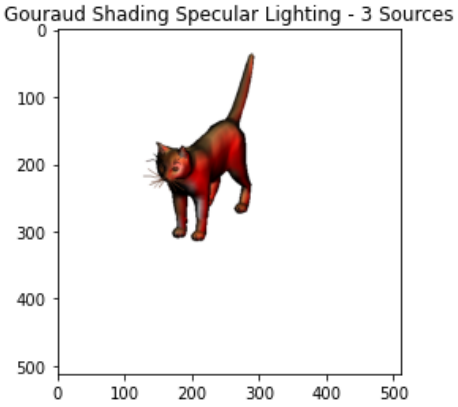
Gouraud Shading

Αποτελέσματα για 3 πήγες φωτισμού

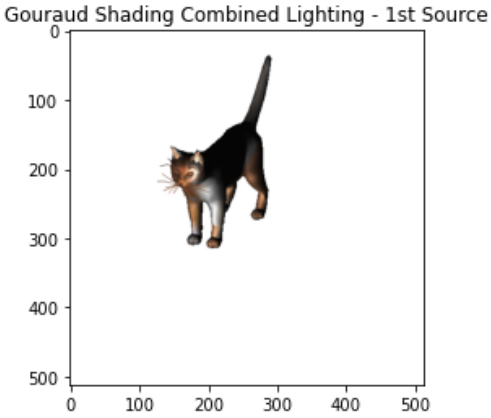


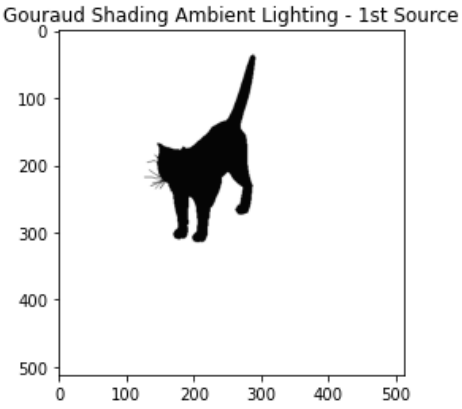


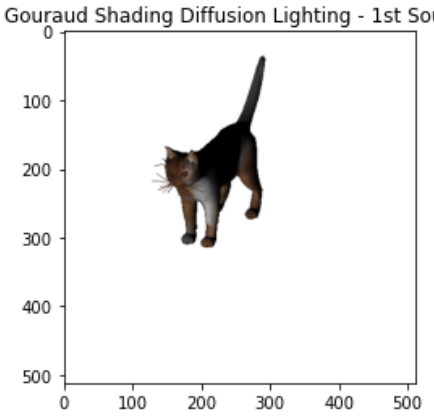


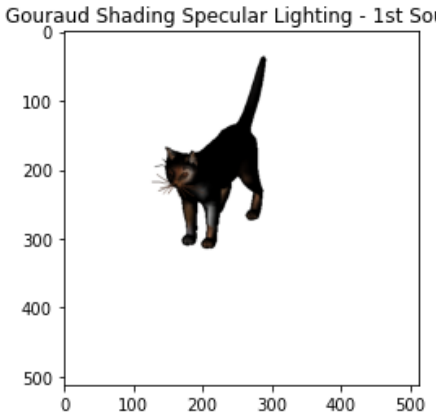


Αποτελέσματα για την 1η πηγή φωτισμού

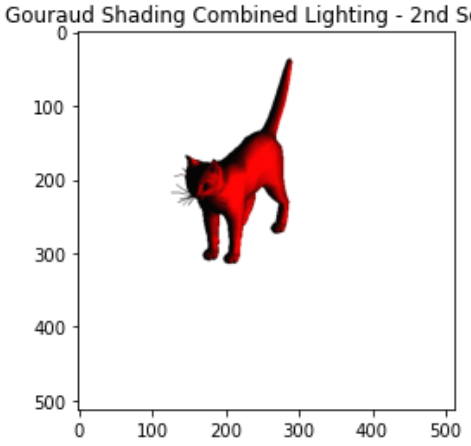


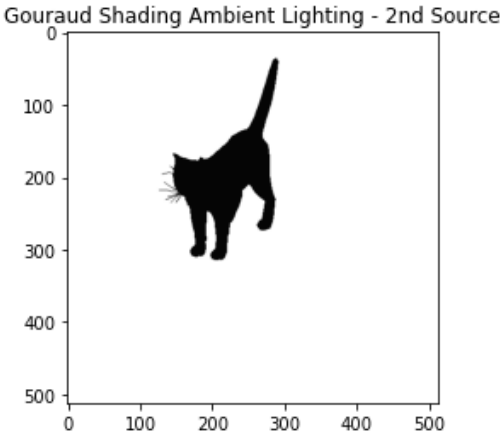


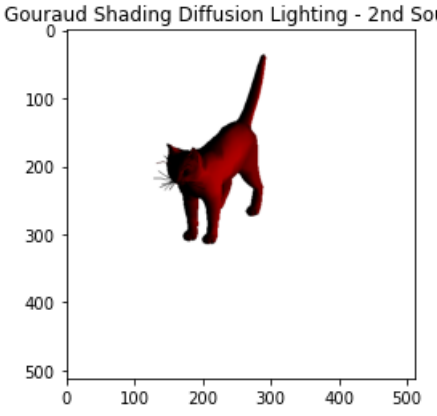


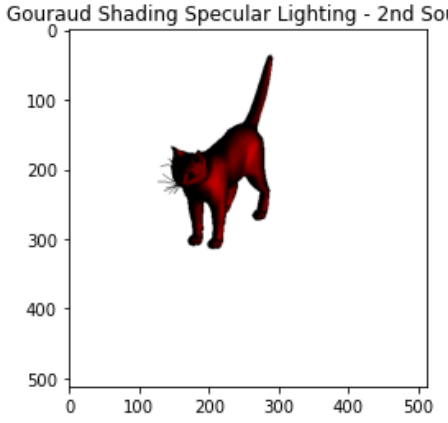


Αποτελέσματα για την 2η πηγή φωτισμού

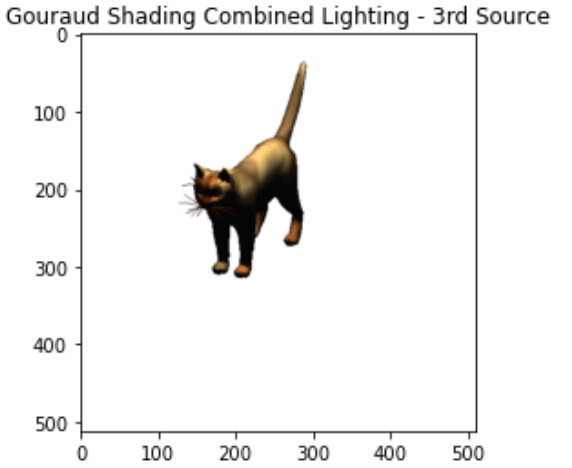


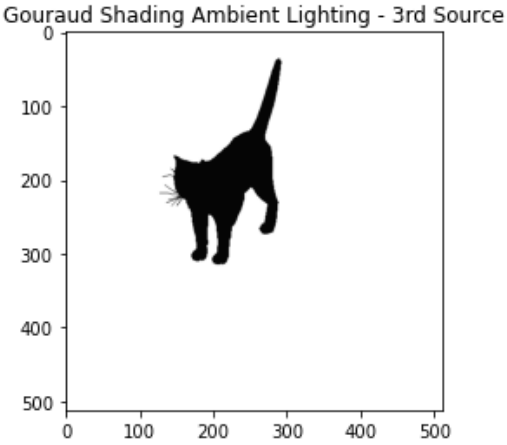


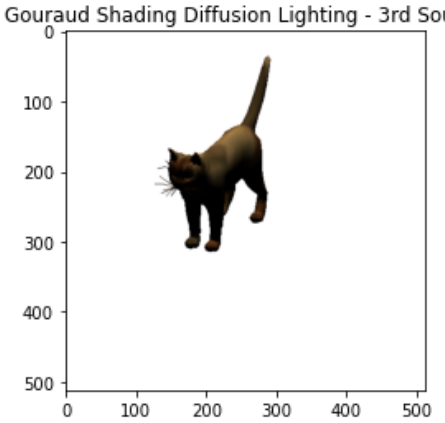


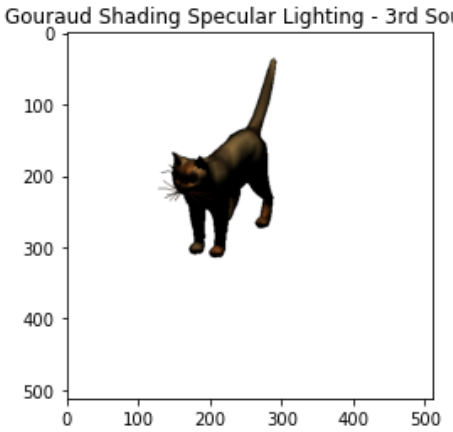


Αποτελέσματα για την 3η πηγή φωτισμού



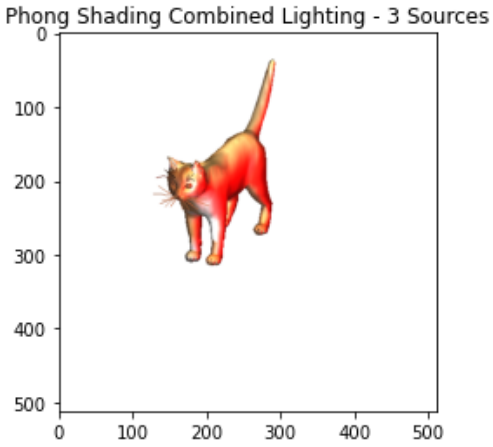


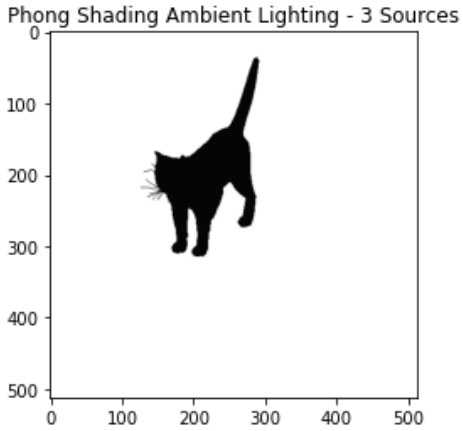


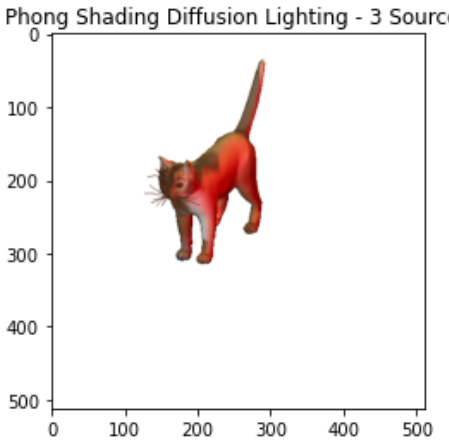


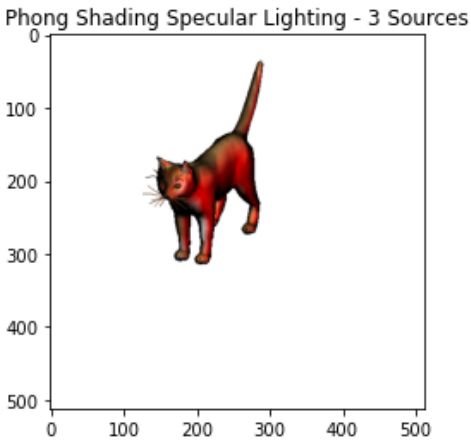
Phong Shading

Αποτελέσματα για 3 πήγες φωτισμού

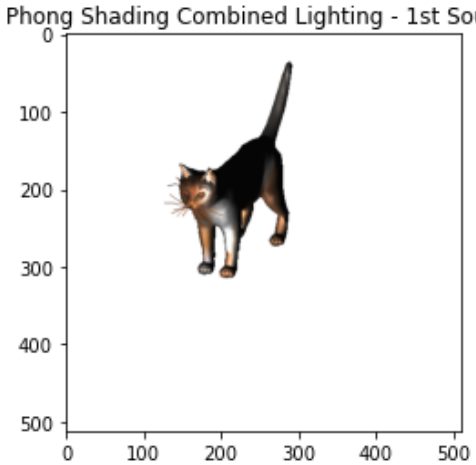


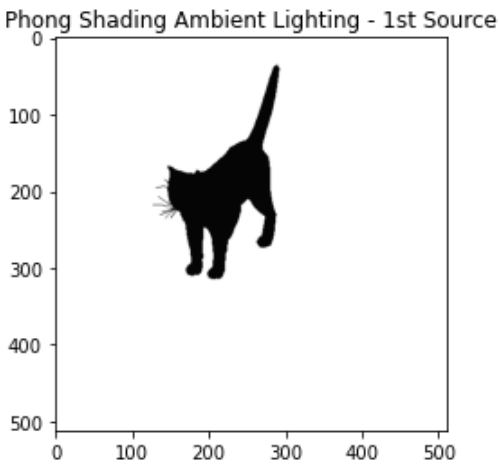


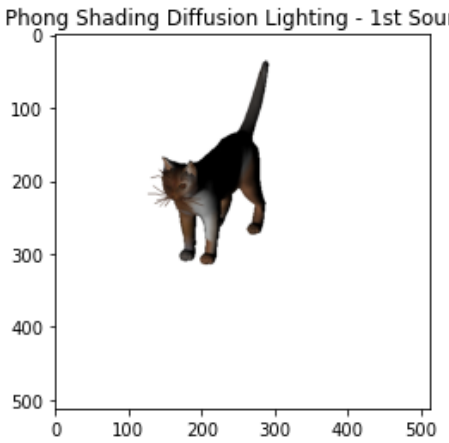


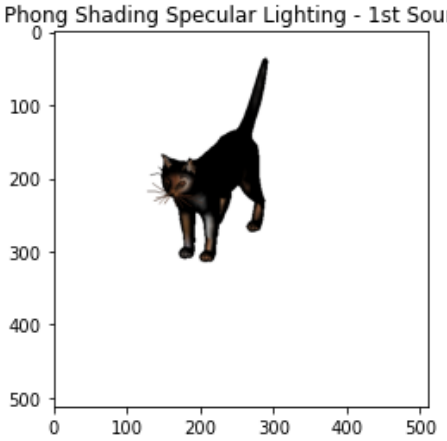


Αποτελέσματα για την 1η πηγή φωτισμού

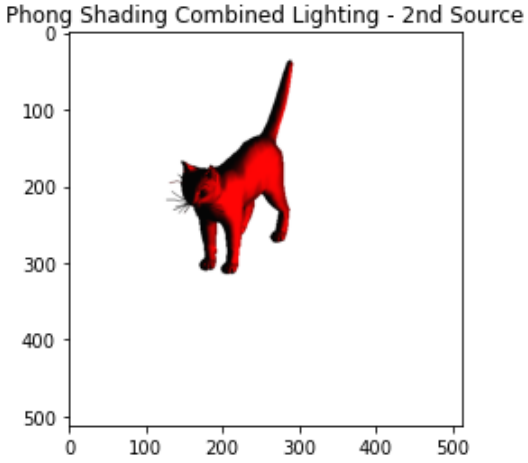


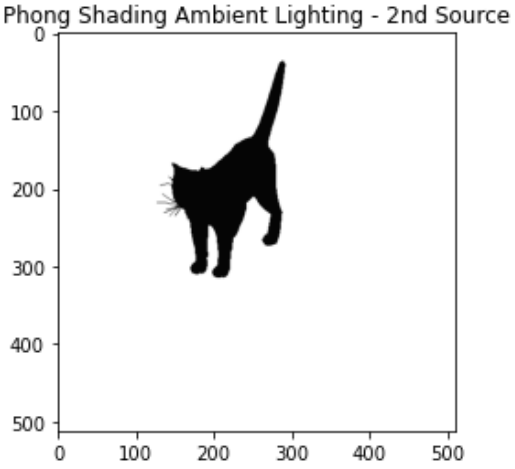


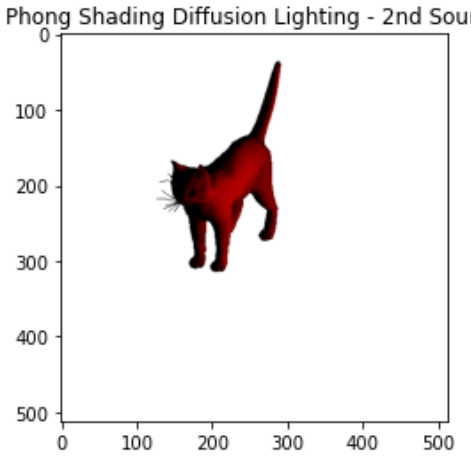


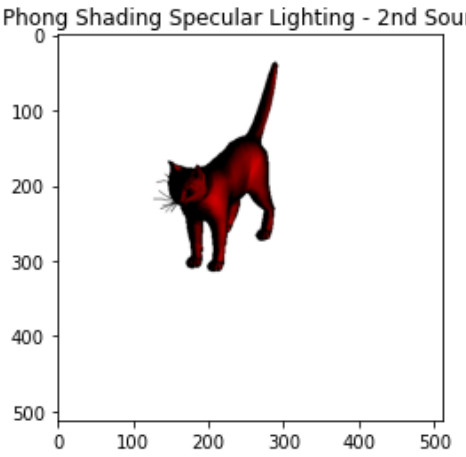


Αποτελέσματα για την 2η πηγή φωτισμού

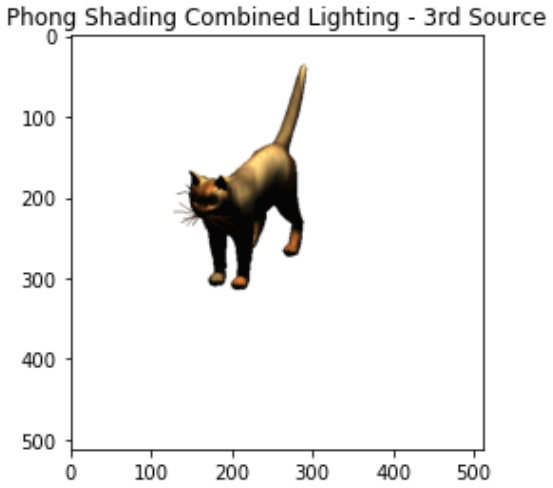


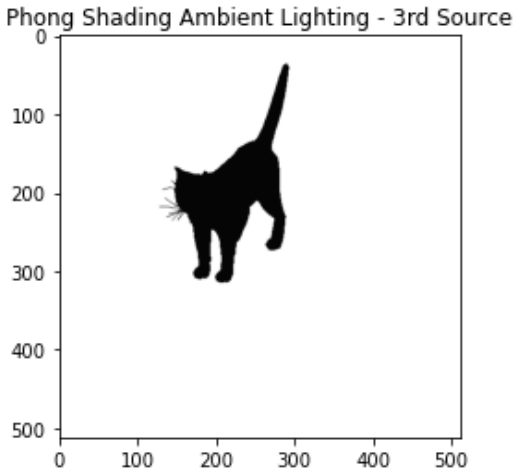


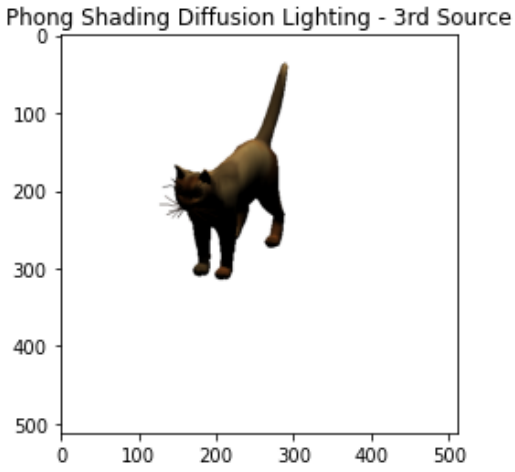


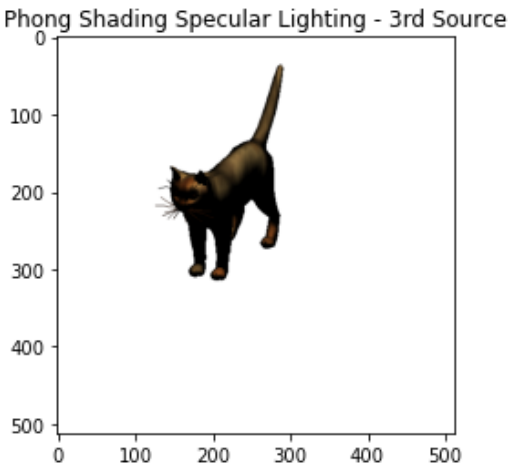


Αποτελέσματα για την 3η πηγή φωτισμού









EXTRA ΠΑΡΑΔΟΤΕΟ

Για το εξτρά παραδοτέο αρχικά ζητούνταν η υλοποίηση της συνάρτησης bilerp() ,η οποία επιστρέφει το χρώμα που αντιστοιχεί στις δοθείσες 2Δ συντ/νες *uv* μέσω διγραμμικής παρεμβολής στην εικόνα εισόδου texture_map. Οι 2 μεταβλητές της συνάρτησης είναι ***uv***: 2Δ συντ/νες 1×2 και **texture_map**: η εικόνα της υφής του αντικειμένου $M \times M \times 3$.

Για την υλοποίηση αυτής της συνάρτησης και γενικά για την κατανόηση του ζητήματος της υφής χρησιμοποιήθηκε το βιβλίο 'Γραφικά και Οπτικοποίηση' και συγκεκριμένα το κεφάλαιο 14 που αφορά την υφή. Συνεπώς :

- ***color = bilerp(uv, texture_map)***
  Η συνάρτηση bilerp εκτελεί bilinear παρεμβολή για να ανακτήσει το χρώμα που αντιστοιχεί στις δοσμένες δισδιάστατες συντεταγμένες UV από τον χάρτη

υφής (texture map). Η bilinear παρεμβολή χρησιμοποιείται συχνά στη γραφική υπολογιστική για την ομαλή μετάβαση μεταξύ χρωμάτων ή τιμών. Αρχικά, οι συντεταγμένες u και v μετατρέπονται από την περιοχή [0, 1] στην περιοχή [0, width-1] και [0, height-1] αντίστοιχα αφού πρώτα γίνει έλεγχος αν οι συντεταγμένες u και v είναι NaN. Στην συνέχεια, υπολογίζονται οι συντεταγμένες των τεσσάρων pixels που περιβάλλουν το σημείο (x, y). Έπειτα, υπολογίζει τις αποστάσεις του σημείου (x, y) από τα pixels στα οποία βρίσκεται κοντά. Δηλαδή :

- **dx = x - x0** # Απόσταση από το σημείο στο αριστερό pixel στην x κατεύθυνση
- **dy = y - y0** # Απόσταση από το σημείο στο πάνω pixel στην y κατεύθυνση

Από τα γειτονικά pixels εξάγονται τα χρώματα από την εικόνα texture_map ως εξής :

- **texelColor00 = texture_map[y0, x0] # Χρώμα στο (x0, y0)**
- **texelColor10 = texture_map[y0, x1] # Χρώμα στο (x1, y0)**
- **texelColor01 = texture_map[y1, x0] # Χρώμα στο (x0, y1)**
- **texelColor11 = texture_map[y1, x1] # Χρώμα στο (x1, y1)**

Τέλος, για να υπολογιστεί το τελικό χρώμα οπου είναι και η έξοδος της συνάρτησης στο σημείο (x, y), συνδυάζονται τα χρώματα των γύρω pixels με βάρη που εξαρτώνται από τις αποστάσεις:

```
texelColor = ( texelColor00 * (1 - dx) * (1 - dy) + texelColor10
* dx * (1 - dy) + texelColor01 * (1 - dx) * dy + texelColor11 *
dx * dy)
```

Παρακατω παρουσιάζεται λίγο πιο αναλυτικά η διαδικασία της διγραμμικης παρεμβολής που χρησιμοποιήθηκε (σελ. 451-460)

Για ένα δεδομένο σημείο (x, y) στην εικόνα, εντοπίζονται οι συντεταγμένες των τεσσάρων κοντινότερων γειτονικών σημείων:

- (x1,y1) είναι το πάνω-αριστερά σημείο
- (x2,y1) είναι το πάνω-δεξιά σημείο
- (x1,y2) είναι το κάτω-αριστερά σημείο
- (x2,y2) είναι το κάτω-δεξιά σημείο

Οι συντεταγμένες των τεσσάρων κοντινότερων σημείων (x1, y1), (x2, y1), (x1, y2) και (x2, y2) υπολογίζονται ως εξής:

- Το (x1, y1) είναι το σημείο με τις συντεταγμένες αριστερά και πάνω από το (x, y). Υπολογίζεται ως το πλησιέστερο ακέραιο μικρότερο ή ίσο με το x για την x συντεταγμένη και το πλησιέστερο ακέραιο μικρότερο ή ίσο με το y για την y συντεταγμένη.
- Το (x2, y1) είναι το σημείο με τις συντεταγμένες δεξιά και πάνω από το (x, y). Υπολογίζεται ως το πλησιέστερο ακέραιο μεγαλύτερο από το x για την x-συντεταγμένη και το πλησιέστερο ακέραιο μικρότερο ή ίσο με το y για την y-συντεταγμένη.
- Το (x1, y2) είναι το σημείο με τις συντεταγμένες αριστερά και κάτω από το (x, y). Υπολογίζεται ως το πλησιέστερο ακέραιο μικρότερο ή ίσο με το x για την x-συντεταγμένη και το πλησιέστερο ακέραιο μεγαλύτερο από το y για την y-συντεταγμένη.
- Το (x2, y2) είναι το σημείο με τις συντεταγμένες δεξιά και κάτω από το (x, y). Υπολογίζεται ως το πλησιέστερο ακέραιο μεγαλύτερο από το x για την x-συντεταγμένη και το πλησιέστερο ακέραιο μεγαλύτερο από το y για την y-συντεταγμένη.

Εφαρμόζεται γραμμική παρεμβολή κατά μήκος των δύο οριζόντιων γραμμών που περνούν από τα σημεία:

Για την πρώτη οριζόντια γραμμή μεταξύ (x1, y1) και (x2, y1):

$$f(x, y1) = (x2 - x) / (x2 - x1) * f(x1, y1) + (x - x1) / (x2 - x1) * f(x2, y1)$$

Για τη δεύτερη οριζόντια γραμμή μεταξύ (x1, y2) και (x2, y2):

$$f(x, y2) = (x2 - x) / (x2 - x1) * f(x1, y2) + (x - x1) / (x2 - x1) * f(x2, y2)$$

Τέλος, εφαρμόζεται γραμμική παρεμβολή κατά μήκος της κάθετης γραμμής που περνά από τα f(x, y1) και f(x, y2):

$$f(x, y) = (y2 - y) / (y2 - y1) * f(x, y1) + (y - y1) / (y2 - y1) * f(x, y2)$$

Βοηθητικές συναρτήσεις

- `interpolate_uv(uv1, uv2, uv3, w1, w2, w3)`
  Η συναρτηση έχει ως εισόδους :

  A. **uv1, uv2, uv3**: Οι UV συντεταγμένες των τριών κορυφών του τριγώνου. Αυτές είναι οι συντεταγμένες υφής για κάθε κορυφή του τριγώνου.
  B. **w1, w2, w3**: Τα βαρυκεντρικά βάρη του σημείου μέσα στο τρίγωνο. Αυτά είναι τα βάρη που καθορίζουν τη σχετική θέση του σημείου μέσα στο τρίγωνο σε σχέση με τις κορυφές του.

  Η συνάρτηση interpolate_uv εκτελεί παρεμβολή UV συντεταγμένων για ένα σημείο μέσα σε ένα τρίγωνο χρησιμοποιώντας τα βαρυκεντρικά βάρη (barycentric weights).

- ***barycentric_coordinates(p, p1, p2, p3)***
  Η συνάρτηση barycentric_coordinates(p, p1, p2, p3) υπολογίζει τις βαρυκεντρικές συντεταγμένες ενός σημείου p εντός ενός τριγώνου που ορίζεται από τις κορυφές p1, p2 και p3. Αρχικά, η συνάρτηση υπολογίζει τα διανύσματα p1p2, p1p3 και p1p μεταξύ των σημείων p1, p2, p3 και p. Στην συνέχεια, υπολογίζει τις βαρυκεντρικές συντεταγμένες **lambda1, lambda2** και **lambda3** ως το ποσοστό της περιοχής του τριγώνου που ανήκει σε κάθε κορυφή ως εξής (σελ. 457-58) :

1. $\lambda_1 = \frac{\left|\overrightarrow{PP_2} \times \overrightarrow{PP_3}\right|}{\left|\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}\right|}$
2. $\lambda_2 = \frac{\left|\overrightarrow{P_1P} \times \overrightarrow{P_1P_3}\right|}{\left|\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}\right|}$
3. $\lambda_3 = 1 - \lambda_1 - \lambda_2$

Επιπλέον, δημιουργείται και η συναρτηση **shade_texture()** και η συναρτηση **texture_shading()**

texture_shading(img, vertices, triangle_uvs, texture_map)

Η συνάρτηση **texture_shading** είναι υπεύθυνη για την εφαρμογή υφής σε ένα τρίγωνο εντός μιας εικόνας. Η διαδικασία αυτή περιλαμβάνει την αντιστοίχιση σημείων του τριγώνου σε σημεία μιας υφής, ώστε το τρίγωνο να χρωματιστεί σύμφωνα με τα δεδομένα της υφής. Η συναρτηση βασίζεται στις προηγούμενες υλοποιήσεις για το gouraud shading. Αυτό που αλλάζει είναι ότι γίνεται παρεμβολή στο χρώμα των κορυφών του εκάστοτε τριγώνου κατά τη διάρκεια του shading, να γίνεται παρεμβολή στις συντ/νες uv των κορυφών, και να χρωματίζετε το κάθε σημείου του τριγώνου χρησιμοποιώντας το χρώμα που προκύπτει από τη συνάρτηση *bilerp* όπως περιγράφεται από την εκφώνηση.

Παράμετροι Συνάρτησης

- img: Ένας πίνακας διαστάσεων MxNx3 που περιέχει τις τιμές RGB της εικόνας.
- vertices: Ένας πίνακας 3x2 που περιέχει τις συντεταγμένες των κορυφών του τριγώνου.
- triangle_uvs: Ένας πίνακας 3x2 που περιέχει τις συντεταγμένες UV για κάθε κορυφή του τριγώνου.
- texture_map: Ένας πίνακας MxMx3 που περιέχει τις τιμές RGB της υφής.

Η συνάρτηση επιστρέφει την ενημερωμένη εικόνα με την υφή εφαρμοσμένη στο τρίγωνο.

Αρχικά η συναρτηση υπολογίζει τα ελάχιστα και μέγιστα άκρα τόσο για την τετμημένη όσο και για την τεταγμένη (xk,yk,min,max), ενώ παράλληλα εκτελούνται έλεγχοι για οριζόντιες και κάθετες πλευρές στο τρίγωνο. Για κάθε ακμή του τριγώνου, υπολογίζονται τα pixels που την αποτελούν και ελέγχεται αν υπάρχει οριζόντια γραμμή στο πάνω ή κάτω μέρος του τριγώνου. Για κάθε σημείο ακολουθείται η εξης διαδικασία : Υπολογίζονται οι βαρυκεντρικές συντεταγμένες lambda1, lambda2, lambda3 για το σημείο point σε σχέση με τις κορυφές p1, p2, p3 του τριγώνου κλήση της συνάρτησης barycentric_coordinates() .Έπειτα, υπολογίζονται οι συντεταγμένες UV του σημείου point με βάση τις UV συντεταγμένες των κορυφών του τριγώνου και τις βαρυκεντρικές συντεταγμένες μέσω της συνάρτησης interpolate_uv. Χρησιμοποιώντας τις UV συντεταγμένες, γίνεται διγραμμική παρεμβολή (bilinear interpolation)/ χρήση της bilerp στον χάρτη υφής texture_map για να βρεθεί το χρώμα. Το χρώμα εφαρμόζεται στο αντίστοιχο σημείο της εικόνας img. Έπειτα, η συναρτηση υλοποιεί τον αλγόριθμο σάρωσης γραμμής (scanline) για την πλήρωση του τριγώνου με το μέσο χρώμα των κορυφών αφού υπολογιστούν η κατωτάτη και ανώτερη γραμμή σάρωσης. Ακολουθεί ο υπολογισμός των αρχικών ενεργών γραμμών και των αρχικών ενεργών σημείων του τριγώνου και τα σημεία χρωματίζονται με παρόμοιο τρόπο με αυτά των ακμών.

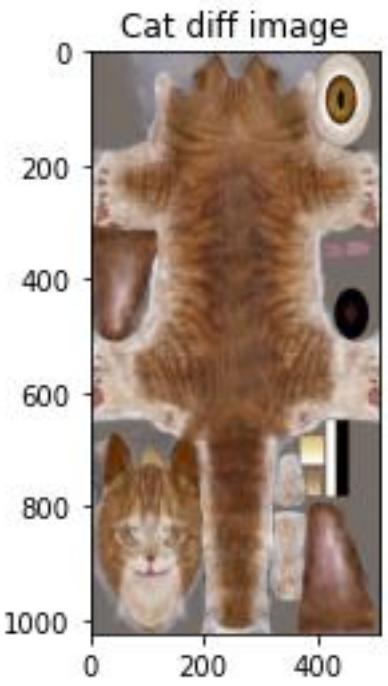
shade_texture(vertsp, vertsn, bcoords, cam_pos, ka, kd, ks, n, lpos, lint, lamb, X, lightOption, triangle_uvs, texture_map)

Η συναρτηση αυτή απλά καλεί την texture_shading() και επιστρέφει την ανανεωμένη εικόνα.

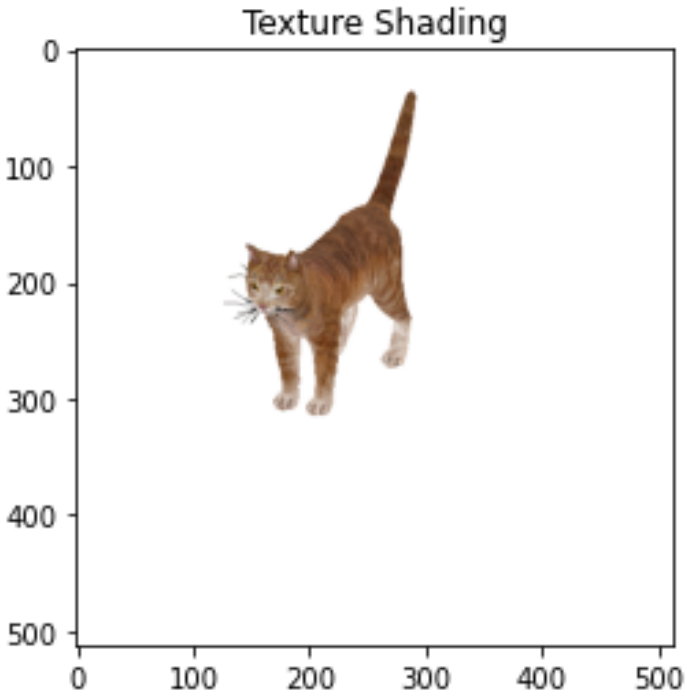
- ***render_object_texture(shader, focal, eye, lookatP, up, bg_color, M, N, H, W, verts, vert_colors, faces, ka, kd, ks, n, lpos, lint, lamb, lightOption, uvs, face_uv_indices, texture_map)***

Η τελευταία συναρτηση είναι η ανανεωμένη έκδοση της render_object() οπου έχουν προστεθεί 3 μεταβλητές εισόδου : οι uv συντεταγμένες των κορυφών, ο πίνακας 3 × *Nfaces* που αντιστοιχίζει τις κορυφές κάθε τριγώνου σε συντ/νες uv και το texture map. Η διαφοροποίηση της εμπλουτισμένης render_object_texture() είναι ότι ο shader μπορεί να πάρει και την επιλογή 'texture', οπου για κάθε τρίγωνο καλείται η shade_texture που αναλύθηκε παραπάνω, αφού υπολογιστούν οι uv συντεταγμένες για τις κορυφές κάθε τρίγωνου με την βοήθεια του πίνακα face_uv_indices και υπολογιστεί και το βαρύκεντρο του τρίγωνου.

Παρακατω παρατίθενται η εικόνα που δόθηκε ως texture_map και το τελικό αποτέλεσμα με χρήση της εμπλουτισμένης συνάρτησης για shader = 'texture'



Σχήμα : Texture map



Σχήμα : Αποτέλεσμα για texture shading