# Γραφική με Υπολογιστές
# -Εργασία #3-

Θωμάς Πλιάκης
tpliakis@ece.auth.gr
AEM: 9018

11 Ιουλίου 2022

## ambient_light

Η συνάρτηση **ambient_light** απλά εκτελεί έναν πολλαπλασιαμό μεταξύ μιας σταθεράς και ενός διανύσματος για να υπολογίσει την ένταση της τριχρωματικής ακτινοβολίας του διάχυτου φωτισμού:

$$I = ka * Ia$$

## diffuse_light

Η συνάρτηση **diffuse_light** για να υπολογίσει την ένταση της τριχρωματικής ακτινοβολίας στην διάχυτη ανάκλαση εκτελεί τα παρακάτω βήματα:

1. Υπολογίζει το διανυσμα μεταξύ του σημείου **P** και της πηγής του φωτός

$$\vec{l} = light\_positions - P$$

2. Κανονικοποιεί αυτό το διάνυσμα

$$\hat{L} = \frac{\vec{l}}{||l||}$$

3. Και τέλος βρίσκει την ακτινοβολία εκτελώντας την πράξη. Υπολογίζοντας το εσωτερικό διάνυσμα μεταξύ του κανονικού διανύσματος και του διανύσματος L.

$$I = light\_intensities * color * kd * \vec{L} \cdot \vec{N}$$

## specular_light

Η συνάρτηση **diffuse_light** για να υπολογίσει την ένταση της τριχρωματικής ακτινοβολίας στην κατοπτρική ανάκλαση εκτελεί τα παρακάτω βήματα:

1. Υπολογίζει το διάνυσμα μεταξύ του σημείου **P** και της πηγής του φωτός και το κανονικοποιεί

$$\vec{l} = light\_positions - P$$

$$\hat{L} = \frac{\vec{l}}{||l||}$$

2. Υπολογίζει το διάνυσμα μεταξύ του σημείου **P** και του παρατηρητή, δηλαδή της κάμερας, και το κανονικοποιεί

$$\vec{v} = cam\_pos - P$$

$$\hat{V} = \frac{\vec{v}}{||v||}$$

3. Και τέλος βρίσκει την ακτινοβολία εκτελώντας την πράξη

$$I = light\_intensities * color * ks * ((2 * \vec{N} * (\vec{N} \cdot \vec{L}) - \vec{L}) \cdot \vec{V})^n$$

## calculate_normals

Η συνάρτηση **calculate_normals** υπολογίζει τον πίνακα των κάθετων διανυσμάτων σε κάθε κορυφή με τα παρακάτω βήματα:

1. Υπολογίζει το διάνυσμα μεταξύ της πρώτης κορυφής και της δεύτερης και το διάνυσμα της δεύτερης και τρίτης κορυφής και παίρνει το εξωτερικό γινόμενο τους ώστε να βρει το κάθετο διάνυσμα στην επιφάνεια και το κανονικοποιεί.

$$\vec{N} = v\vec{0}1 \times v\vec{1}2$$

$$\hat{N} = \frac{\vec{N}}{||N||}$$

2. Υπολογίζει έναν πίνακα που αποθηκέυει σε ποια τρίγωνα ανήκει κάθε κορυφή.

3. Και τέλος με την βοήθεια του πίνακα αυτού υπολογίζει το κάθετο διάνυσμα στην κάθε κορυφή ως τον μέσο όρο των κάθετων διανυσμάτων των τριγώνων που η κορυφή αυτή συμμετέχει.

## render_object

Ο κώδικας στην συνάρτηση **render_object** περιγράφεται στα παρακάτω βήματα:

1. Γίνεται υπολογισμός των κάθετων διανυσμάτων των τριγώνων με την **calculate_normals**.

2. Γίνεται η προβολή των τριγώνων στις 2 διαστάσεις, μετατρέπονται οι ίντσες της κάμερας σε pixels και υπολογίζεται το βάθος κάθε κορυφής με την συνάρτηση **rasterize** της 2 εργασίας.

3. Βάφεται το backgroung της φωτογραφίας.

4. Δημιουργείται η λίστα με τις 4 φωτογραφίες.

5. Γίνεται υπολογισμός του βάθους του κάθε τριγώνου και αποθηκέυεται στο διάνυσμα **Dm**.

6. Ταξινομείται ο πίνακας με τα βάθη των κορυφών κατά φθίνουσα σειρά. Με την ίδια σειρά ταξινομείται και ο πίνακας **faces**, με την βοήθεια του **D_order** που περίεχει την σειρά που ταξινομήθηκε ο **Dm**.

7. Υπολογίζεται το κάντρο βάρους του τριγώνου στον διάνυσμα bcoords.

8. Ανάλογα με ποιο από τα 2 **mode** χρησιμοποιήσουμε, επιλέγετε η κατάλληλη συνάρτηση, με **shader ,["phong" , "gouraud"]**.

## shade_gouraud

Η συνάρτηση **shade_gouraud** υπολογίζει τον φωτισμό κάθε κορυφής ενός τριγώνου μέσω των συναρτήσεων της ενότητας Α και στην συνέχεια καλεί την **shade_triangle** της πρώτης εργασίας για να χρωματίσει το τρίγωνο με την μέθοδο **gouraud**. Η συνάρτηση γεμίζει με χρώμα τα τρίγωνα 4 διαφορετικών φωτογραφιών όπως ζητούνται στην εκφώνηση, και τις επιστρέφει στην συνάρτηση **render_object** ως μια λίστα.

## shade_phong

Η συνάρτηση **shade_phong** ουσιαστικά είναι η συνάρτηση **shade_triangle** της πρώτης εργασίας αλλά κάθε φορά που θα δώσει χρώμα σε κάποιο σημείο ακολουθεί τα εξής βήματα:

1. Υπολογισμός του κάθετου διανύσματος σε κάθε σημείο ως εξής:

   - Αν το τρίγωνο είναι σημείο υπολογίζεται ως ο μέσος όρος των κάθετων διανυσμάτων των κορυφών.

   - Σε κάθε άλλη περίπτωση ως γραμμική παρεμβολή μεταξύ των κορυφών ή των ενεργών σημείων που αυτό βρίσκεται. Τα κάθετα διανύσματα στα ενεργά σημεία υπολογίζονται με γραμμική παρεμβολή των αντίστοιχων κορυφών που βρίσκονται ανάμεσα.

2. Υπολογισμός του χρώματος του σημείου με παρόμοιο τρόπο όπως τα κάθετα διανύσματα.

3. Υπολογιμός του φωτισμού με τις συναρτήσεις της ενότητας Α και τον αντίστοιχων κάθετων διανυσμάτων και χρωμάτων που υπολογίστηκαν για κάθε σημείο.

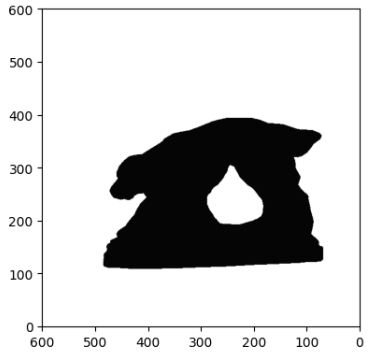
Δηλαδή δεν έπρεπε να γίνει καμιά αλλαγή στο προς βρίσκουμε τα σημεία που βρίσκονται πάνω ή μέσα στο κάθε τρίγωνο, απλά έπρεπε να γίνει προσθήκη όλων των γραμμικών παρεμβολών. Επίσης δημιουρηθηκε μια λίστα με 4 φωτογραφίες όπως ζητήθηκε στην εκφώνηση παρόμοια με την συνάρτηση **shade_gouraud**.

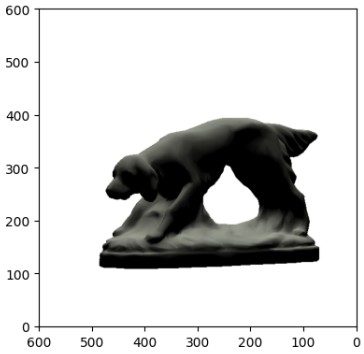
Στο τέλος πραθέτω τις συναρτήσεις από τις προηγούμενες εργασίες που χρησιμοποίησα:

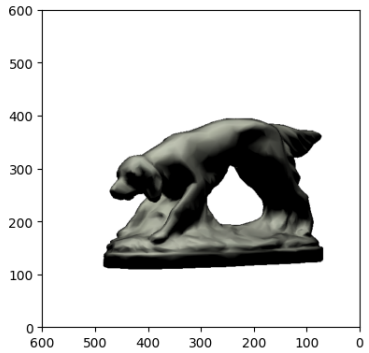
# Σχόλια

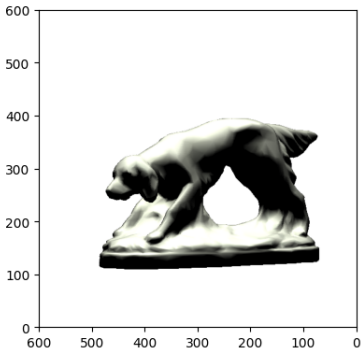
- Τα αποτελέσματα στην συνάρτηση **shade_gouraud** είναι ικανοποιητικά αλλά στην συνάρτηση **shade_phong** είναι λάθος. Για κάποιο λόγο που δεν μπόρεσα να βρω δεν χρωματίζονται σωστά τα τριγώνα αν και προσπάθησα να μην αλλάξω την διαδικασία πλήσρωσης του τριγώνου.
- Η μόνη παραδοχή που έγινε ήταν στην δημιουργία των λιστών των 4 φωτογρφιών για κάθε συνάρτηση ώστε να καταφέρω να τις δημιουργήσω τρέχοντας μόνο το αρχείο demo.py ώστε να μην πειράζω τον κώδικα για κάθε εικόνα.
- Χρειάστηκαν περίπου 6 λέπτα για να παραχούν όλες οι εικόνες στον υπολογιστή μου.

# Αποτελέσματα για gouraud(ambient,diffuse,specular,all3together):

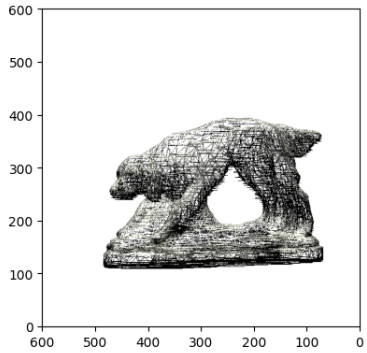
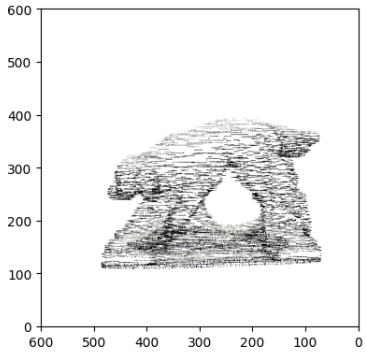
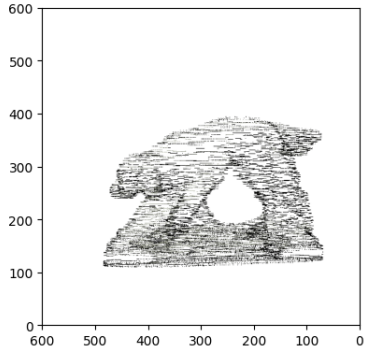
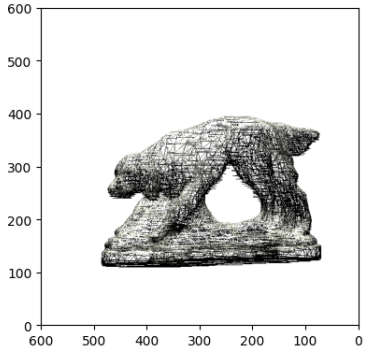


## Αποτελέσματα για phong(ambient,diffuse,specular,all3together):

## interpolate_color

Η συνάρτηση **vector_interp** υλοποιεί πολύ απλά μια γραμμική παρεμβολή μεταξύ των σημείων που δίνονται ως ορίσματα με τον παρακάτω τρόπο:

$$r1 = ||x2 - x||$$

$$r2 = ||x - x1||$$

$$percent = \frac{r1}{r1 + r2}$$

$$value = percent * C1 + (1 - percent) * C2$$

Όπου r1 και r2 είναι οι απόστασεις των ενεργών σημείων από το σημείο της παρεμβολής, r1 + r2 είναι η απόσταση των σημείων x1, x2. Με την διαίρεση γίνεται μια κανονικοποίηση για να έχουμε το ποσοστό μεταξύ 0 και 1 που πρέπει να συμμετέχει κάθε σημείο στον χρωματισμό.

Ο κώδικας, λόγω της βιβλιοθήκης **numpy** μπορεί να υλοποιεί την γραμμική παρεμβολή με μεταβλητές οποιονδήποτε διαστάσεων.

## shade_triangle

Επειδή η υλοποίηση των δύο **mode ,["flat" , "gouraud"]**, ξεκίνησε διαφορετκά, η ενοποιήση των 2 διαδικασιών έγινε με ένα **if ... else** ανάλογα το όρισμα **shade_t**.

Η συνάρτηση **shade_triangle** με όρισμα **shade_t = "flat"** αποδίδει ένα χρώμα σε κάθε τρίγωνο, τον μέσο όρο των χρωμάτων των κορυφών του. Τα βήματα του αλγορίθμου περιγράφονται παρακάτω:

1. Υπολογισμός και αποθήκευσή του μέσου όρου των χρωμάτων στον 1x3 πίνακα **c**.
2. Υπολογισμός των ελάχίστων και μεγίστων συντεταγμένων της κάθε πλευράς, καθώς και της κλίσης τους, **Ykmin,Xkmin,Ykmax,Xkmax**.
3. Εύρεση των ολικών μεγίστων και ελαχίστων συντεταγμένων του τριγώνου, **Ymin,Xmin,Ymax,Ymin**.

4. Βρίσκουμε ποιες είναι οι αρχικές ενεργές πλευρές(παίρνουν το λογικό 1) και αποθηκεύουμε σε πίκανκα τις συντεταγμένες τους, που αποτελούν και τα ενεργά οριακά τους σημεία **ActiveSides,Xk**. Στον πίνακα **Xact** αποθηκεύονται οι τετμημένες των 2 ενεργών σημείων μόνο.

5. Μπαίνουμε στην κυρίως επανάληψη στην οποία:

   - Ελέγχουμε αν βρισκόμαστε σε κορυφή του τριγώνου.
   - Υλοποιείται το σκανάρισμα σε όλες τις τετμημένες για συγκεκριμένη τεταγμένη κατά την οποία αν κάποιο **pixel** βρίσκεται μέσα ή πάνω στο τρίγωνο χρωματίζεται.
   - Ενημερώνεται η λίστα των ενεργών πλεύρων.
   - Τέλος ενημερώνεται η λίστα με τα νέα ενεργά σημεία ανάλογα με την κλίση της κάθε πλευράς του τριγώνου.

Η συνάρτηση **shade_triangle** με όρισμα **shade_t = "gouraud"** έκτελεί τις ίδιες αρχικοποιήσεις με προηγουμένως αλλά και κάποιες ακόμα. Τα επιπλέον βήματα του αλγορίθμου που γίνονται είναι:

1. Αποθηκεύση των δεικτών των κορυφών κάθε ακμής στις λίστες **minindex, maxindex**.

2. Αποθηκεύση των δεικτών των 2 ενεργών σημείων κάθε ακμής στις λίστα **index** για προσέλαση των **colorsAct**, **Xact** πινάκων.

3. Αποθηκεύση του χρώματος των ενεργών οριακών σημείων στην λίστα **colorsAct**.

4. Όταν γίνεται ο χρωματσμός κάποιου σημείου εσωτερικά του τριγώνου, υπολογίζεται το χρώμα του από την **interpolate_color** και τα 2 ενεργά σημεία.

5. Ο υπολογισμός του χρώματος για τα ενεργά οριακά σημεία γίνεται με χρήση της **interpolate_color** από τις αντίστοιχες κορυφές κατά την ενημέρωση της λίστας με τα ενεργά σημεία.

Όλα τα παραπάνω γίνονται για τον χρωματισμό ενός αντικειμένου ολόκληρου που αποτελείται από πολλά τρίγωνα. Αυτό γίνεται στην συνάρτηση:

## affine_transform

Η πρώτη συνάρτηση εκτελεί τον μετασχηματισμό affine. Πρώτα κατασκευάζει τον πίνακα περιστροφής με βάση την φόρμουλα του Rodrigues μετά κατασκευάζει τον πίνακα μετασχηματισμού και στην συνέχεια γίνεται ο υπολογισμός με ομογενείς συντεταγμένες.

## system_transform

Αυτή η συνάρτηση αλλάζει το σύστημα συντεταγμένων των σημείων που δίνονται με βάση των πίνακα στροφής που δίνεται. Πρώτα δημιουργείται ο πίνακας μετασχηματισμού και μετά οι συντεταγμένες των κορυφών μετατρέπονται σε ομογενείς συντεταγμένες για να γίνουν οι πράξεις.

## project_cam

Η συνάρτηση αυτή «φωτογρφίζει» το αντικείμενο. Πρώτα γίνεται η αλλαγή συστήματος συντεταγμένων με την προηγούμενη συνάρτηση των σημείων απο το WCS στο σύστημα της κάμερας. Στην συνέχεια υπολογίζουμε την απόσταση από την κάμερα και με βάση την προοπτική προβολή οι 3διάστατες συντεταγμένες του γίνονται 2 διαστάσεων με πολλαπλασιαμό με την μεταβλήτη $\varphi$ και διαίρεση με το βάθος.

## project_cam_lookat

Εδώ υπολογίζονται τα μοναδιαία διανύσματα του πλαισιού της κάμερας τα οποία τα περνάμε στην προηγούμενη συνάρτηση ώστε να μπορέσει να στοχεύσει σωστά το αντικείμενο προς φωτογράφιση.

## rasterize

Η συνάρτηση αυτή ελέγχει αν όλα τα σημεία βρίσκονται μέσα στο πλάνο μας, ελέγχοντας αν οι διαστάσεις τους βρίσκονται μέσα στα εύρη (-camw/2, camw/2) στον οριζόντιο άξονα και (-camh/2, camh/2) στον κάθετο, μετακινεί την αρχή της εικόνας κάτω αριστερά και τέλος μετατρέπει τις ίντσες τους πετάσματος της κάμερας σε ακέραιες θέσεις (pixels) της εικόνας διαιρώντας τις συνολικές ιντσές προς τα συνολικά πιξελς για να πάρουμε την κλίμακα μετατροπής. Με την κλίμακα αυτή μπορούμε να βρούμε τα pixel των επιμέρους κορυφών με μια διαίρεση και με την συνάρτηση ceil της numppy για να γίνουν ακέραια.