Γραφική με Υπολογιστές
Εργασία 1: Πλήρωση Τριγώνων

Φίλιππος Ρωσσίδης AEM:10379

7 Απριλίου 2024

1 Εισαγωγή

Στην εργασία αυτή πραγματευόμαστε την πλήρωση τριγώνων δοσμένων των συντεταγμένων και των χρωμάτων των τριών κορυφών τους. Έπειτα θα χρησιμοποιηθούν οι συναρτήσεις πλήρωσης για να προβληθεί ένα σχήμα αποτελούμενο από πολλά τρίγωνα σε μία εικόνα.

2 Περιγραφή Λογικής του Αλγορίθμου

Ο αλγόριθμος που υλοποίησα βασίζεται στον αλγόριθμο πλήρωσης πολυγώνων των σημειώσεων, με κάποιες αλλαγές ώστε να τρέχει πιο αποτελεσματικά σε τρίγωνα.

Το πρώτο βήμα είναι ο έλεγχος εάν δύο ή παραπάνω κορυφές που δίνονται συμπίπτουν. Τότε δεν μπορεί να σχηματιστεί τρίγωνο, οπότε η συνάρτηση τερματίζει. Έπειτα από τα δωσμένα σημεία υπολογίζουμε τις μεταβλητές $ymin, ymax, xman, xmax, slope, pointmin, pointmax$ των τριών ευθειών ως εξίς: Αν τα δύο σημεία που ορίζουν την ευθεία είναι $(x_1, y_1), (x_2, y_2)$

$$ymin = min\{y_1, y_2\}, ymax = max\{y_1, y_2\}, xmin = min\{x_1, x_2\}, xmax = max\{x_1, x_2\}$$

$$slope = \frac{y_1 - y_2}{x_1 - x_2}$$

ή $slope = \infty$ αν $x_1 = x_2$

Οι μεταβλητές $pointmin$ και $pointmax$ δηλώνουν το ελάχιστο και μέγιστο κατα y σημείο της ευθείας. Υπολογίζονται ως εξίς:

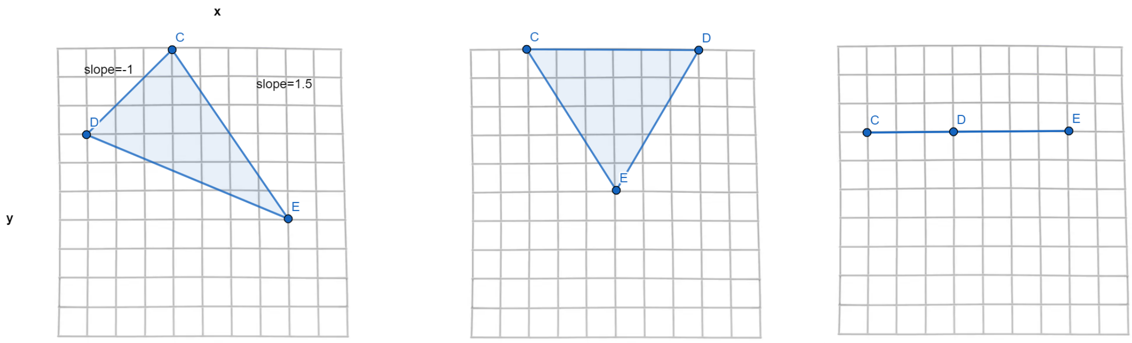
if $y1 = ymin$ **then**
    pointmin=(x1,y1) , pointmax=(x2,y2)
else
    pointmin=(x2,y2) , pointmax=(x1,y1)
end if

Αν η ευθεία είναι οριζόντια, τα $pointmin, pointmax$ είναι αυθαίρετα.

Έπειτα υπολογίζονται τα συνολικά $ymin, ymax$ δηλαδή το μικρότερο $ymin$ και το μεγαλύτερο $ymax$ και από τις τρείς γραμμές. Αυτά θα χρειαστούν στη συνέχεια για τα pixel που θα χρωματιστούν.

Το επόμενο βήμα είναι ο υπολογισμός των αρχικών ενεργών οριακών σημείων. Τα ενεργά οριακά σημεία για τρίγωνα θα είναι πάντα ακριβώς 2 όσο υπάρχουν. Θα κάνω επίσης τη σύμβαση ότι όταν το σημείο τομής της ευθείας y με το τρίγωνο είναι ένα, στα ενεργά οριακά οριακά σημεία εκείνου του ύψους y θα προστεθεί το ίδιο σημείο δύο φορές, με διαφορετικές κλίσεις για τις δύο διαφορετικές ευθείες που ανήκει.

Για τα αρχικά ενεργά οριακά σημεία κοιτάω ποιές ευθείες τέμνουν το ύψος $ymin$. Ξέρω ότι θα υπάρχουν τουλάχιστον δύο ευθείες, γιατί υπάρχει τουλάχιστον ένα σημείο σε εκείνο το ύψος. Αν τέμνουν και οι τρείς ευθείες, τουλάχιστον η μία από αυτές θα είναι οριζόντια. Παίρνω τότε σαν οριακά σημεία τα σημεία τομής των δύο μη οριζόντιων ευθειών με το $ymin$, επομένως κοιτάω για κάθε ευθεία αν τέμνει το $ymin$ και αν έχει μη μηδενική κλίση, τότε το ελάχιστο ως προς y σημείο της ($pointmin$) προσθέτω στα ενεργά οριακά σημεία, με την κλίση της ευθείας, που θα χρειαστεί αργότερα. Με τον παραπάνω αλγόριθμο, αν μόνο ένα σημείο υπάρχει στο $ymin$ τότε προστίθεται δύο φορές με δύο διαφορετικές κλίσεις, για τις δύο ευθείες που ανήκει, αν η μία ευθεία είναι οριζόντια προστίθονται τα δύο οριακά σημεία των άλλων δύο ευθειών με την κλίση τους, και τέλος αν και οι τρεις ευθείες είναι οριζόντιες στο $ymin$, τότε δεν προστίθεται κανένα σημείο. Αυτό δεν είναι πρόβλημα μιας και τότε το τρίγωνο δε θα έχει εσωτερικό.



Στα παραπάνω σχήματα φαίνονται οι τρείς αυτές περιπτώσεις. Στην πρώτη περίπτωση στα αρχικά ενεργά οριακά σημεία προστίθεται η ακμή C δύο φορές με κλίσεις -1 και 1.5, στη δεύτερη περίπτωση προστίθονται οι ακμές C και D με τις κλίσεις από τις μη οριζόντιες ευθείες και στην τρίτη περίπτωση δεν υπάρχουν οριακά σημεία αφού δεν υπάρχει εσωτερικό του τριγώνου.

Ακολουθεί η λούπα που χρωματίζει τα σωστά pixel.

Θα γίνεται μια επανάληψη για κάθε $y \in [ymin, ymax]$ όπου ξεκινόντας από το πιο αριστερό ενεργό οριακό σημείο μέχρι και πριν το πιο δεξιό χρωματίζεται το pixel (x,y) με τις τεχνικές που θα περιγραφτούν στη συνέχεια.

Τέλος ανανεώνονται τα ενεργά οριακά σημεία. Αρχικά ελέγχω αν υπάρχει ευθεία που τελειώνει στο τωρινό y ($line.ymax = y$) και αν ναι αφαιρώ από τα ενεργά οριακά σημεία το σημείο που ανήκει σε αυτή την ευθεία. Έπειτα ελέγχω αν υπάρχει ευθεία που ξεκινάει στο τωρινό y ($line.ymin = y$) και προσθέτω το ελάχιστο κατα y σημείο της ευθείας στα ενεργά οριακά σημεία . Αυτή η προσθήκη δεν πρέπει να πραγματοποιηθεί στην πρώτη επανάληψη (στο $ymin$) μιας και τα σημεία που ανήκουν στις ευθείες που ξεκινούν τότε έχουν ήδη προστεθεί. Έτσι μπάινει ο επιπλέον έλεγχος $y \neq ymin$. Και τέλος τα σημεία που έχουν μείνει στην λίστα ενεργών οριακών σημείων ανανεώνονται ως εξίς:

Αν το οριακό σημείο είναι το (x, y):

$$newpoint = (x + 1/slope, y + 1)$$

Αν $slope = \infty$:

$$newpoint = (x, y + 1)$$

Προκίπτει λοιπόν ο ψευδοκώδικας Algorithm 1 για την πλήρωση τριγώνου.

Algorithm 1 Πλήρωση Τριγώνου

```
function SHADING(image, vertices, vcolors)
    if any two vertices are the same then
        return
    end if
    for every line do
        Find ymin, ymax, xmin, xmax, slope, pointmin, pointmax
    end for
    Find ymin, ymax                                   ▷ total y min and max
    for every line do
        if line.ymin = ymin and line.slope ≠ 0 then
            Add line.pointmin to Active Boundary Points
        end if
    end for
    for y = ymin : 1 : ymax do
        Sort Active Boundary Points
        for x = ceil(ActiveBoundaryPoints[0]) : 1 : ceil(ActiveBoundaryPoints[1]) do
            Color Pixel                               ▷ More details below
        end for
                          ▷ Calculation of Active Boundary Points for next y:
        for every line do
            if y = line.ymax then
                Remove boundary point belonging to line
            else if y = line.ymin and line.slope ≠ 0 and y ≠ ymin then
                Add line.pointmin to Active Boundary Points
            end if
        end for
        for every Active Boundary Point do
            if slope = ∞ then
                Updated Boundary Point = [x,y+1]
            else
                Updated Boundary Point = [x+1/slope,y+1]
            end if
        end for
    end for
end function
```

3 Παραδοχές

Με τον τρόπο λειτουργίας του αλγορίθμου που περιγράφθηκε προκίπτουν κάποιες παραδοχές για το ποιά pixel χρωματίζονται.

- Οι οριζόντιες γραμμές θα χρωματίζονται.
- Ο χρωματισμός ξεκινάει από το αριστερό οριακό σημείο, ή αν η τετμημένη του οριακού σημείου δεν είναι ακέραια, ξεκινάει από το πρώτο pixel δεξιά του αριστερού οριακόυ σημείου, και συνεχίζει μέχρι και πριν το δεξιό οριακό σημείο. Το δεξιό οριακό σημείο δεν χρωματίζεται ακόμα και να είναι σε ακέραια τετμημένη.
- Αν τα οριακά σημεία συμπίπτουν δεν χρωματίζεται κανένα pixel.

4 Η συναρτήσεις vector_interp και double_vector_interp

Ένα ζητούμενο της εργασίας είναι η υλοποίηση της συνάρτησης γραμμικής παρεμβολής. Δέχεται σαν ορίσματα δύο σημεία p_1, p_2, δύο διανύσματα V_1, V_2, τις συντεταγμένες ενός σημείου $coord$ και μια τιμή dim για το αν η παρεμβολή θα γίνει κατά x ή κατά y. Ζητούμενο είναι η παραγωγή ενός διανύσματος για το σημείο με συντεταγμένη $coord$ ως γραμμική παρεμβολή των διανυσμάτων V_1, V_2 που ανήκουν στα σημεία p_1, p_2.

Για την παραγωγή αυτή θα δημιουργήσω μία τιμή λ ορισμένη ως: Αν $p_1 = (x_1, y_1)$ $p_2 = (x_2, y_2)$ και $dim = 1$:

$$\lambda = \frac{coord - x_1}{x_2 - x_1}$$

Αν $dim = 2$:

$$\lambda = \frac{coord - y_1}{y_2 - y_1}$$

Το αποτέλεσμα V θα είναι:

$$V = \lambda V2 + (1 - \lambda)V1$$

Καταλίγουμε στον ψευδοκώδικα:

Algorithm 2 Γραμμική Παρεμβολή

function VECTOR_INTERP(p1,p2,V1,V2,coord,dim)
  if $dim = 1$ **then**
    $\lambda = (coord - x_1)/(x_2 - x_1)$
  else if $dim = 2$ **then**
    $\lambda = (coord - y_1)/(y_2 - y_1)$
  end if
  return $\lambda V2 + (1 - \lambda)V1$
end function

Η συνάρτηση double_vector_interp θα χρησιμοποιηθεί μετέπειτα στο χρωματισμό με την τεχνική gouraud. Χρησιμοποιείται για την εύρεση ενός χρώματος (διανύσματος) στο εσωτερικό ενός τριγώνου από τα χρώματα (διανύσματα) στις κορυφές του. Πρώτα υπολογίζεται με τη συνάρτηση vector_interp το διάνυσμα των δύο οριακών σημείων από τις δύο κορυφές της ευθείας στην οποία ανήκουν, και έπειτα από αυτά τα δύο σημεία υπολογίζεται ξανά μέσω της συνάρτησης vector_interp το διάνυσμα στο θεμιτό σημείο.

Ιδιαίτερη προσοχή χρειάζονται τα ορίσματα στη συνάρτηση γραμμικής παρεμβολής που θα κλιθεί γιατί μπορεί να προκίψει διαίρεση με το 0. Ευτυχώς υπάρχουν δύο τρόποι υπολογισμού της γραμμικής παρεμβολής: ως προς x και ως προς y. Έτσι πριν καλέσουμε την vector_interp ως προς x ελέγχουμε αν οι τετμημένες των σημείων είναι διάφορες, και αντίστοιχα ως προς y. Από τη δομή του αλγορίθμου shading δε θα έπρεπε ποτέ να δωθεί το ίδιο σημείο δύο φορές στην συνάρτηση γραμμικής παρεμβολής ώστε και οι τετμημένες τους και οι τεταγμένες να είναι ίσες, έτσι στον κώδικα python σε αυτήν την περίπτωση καλείται η: raise Exception('division by zero')

Προκίπτει ο ψευδοκώδικας Algorithm 3.

Algorithm 3 Διπλή Γραμμική Παρεμβολή

```
function DOUBLE_VECTOR_INTERP(point, BoundaryLeft, BoundaryRight)
    Εύρεση Αριστερής και δεξιάς γραμμής από τα οριακά σημεία
    ΑΓ = Αριστερή Γραμμή
    ΔΓ = Δεξιά Γραμμή
    if Τετμημένες των δύο ακραίων σημείων της ΑΓ δεν είναι ίσες then
        V1 = vector_interp (ΑΓ.σημείο1,ΑΓ.σημείο2,χρώμα σημείου1, χρώμα σημείου2,BoundaryLeft.x,1)
    else if Τεταγμένες των δύο ακραίων σημείων της ΑΓ δεν είναι ίσες then
        V1 = vector_interp (ΑΓ.σημείο1,ΑΓ.σημείο2,χρώμα σημείου1, χρώμα σημείου2,BoundaryLeft.y,2)
    end if
    Το ίδιο για την δεξιά γραμμή αποθηκεύοντας το αποτέλεσμα στην V2
    if Τετμημένες των BoundaryLeft,BoundaryRight δεν είναι ίσες then
        V = vector_interp (BoundaryLeft,BoundaryRight,V1,V2,point.x,1)
    else if Τεταγμένες των BoundaryLeft,BoundaryRight δεν είναι ίσες then
        V = vector_interp (BoundaryLeft,BoundaryRight,V1,V2,point.y,2)
    end if
    return V
end function
```

5 Χρωματισμός

Η διαδικασία του χρωματισμού (color pixel στον αλγόριθμο Shading) θα γίνει με δύο μεθόδους: flat shading και gouraud shading. Οι συναρτήσεις που θα περιγραφούν παρακάτω είναι παραλαγές της συνάρτησης Shading που περιγράφτηκε παραπάνω, όπου στην εντολή color pixel προστίθενται διαφορετικοί τρόποι χρωματισμού.

5.1 Η Συνάρτηση f_shading

Ο flat χρωματισμός γίνεται χρωματίζοντας κάθε pixel του τριγώνου με τον διανυσματικό μέσο όρο των χρωμάτων των κορυφών του. Επομένως:

image(x,y) = (vcolors[0] + vcolors[1] + vcolors[2])/3 όπου vcolors[0] είναι το πρώτο στοιχείο του πίνακα vcolors ο οποίος περιέχει τα χρώματα των τριών κορυφών του τριγώνου. Το κάθε στοιχείο αυτού του πίνακα είναι ένα διάνυσμα που δηλώνει το χρώμα σε RGB. Έτσι το αποτέλεσμα image(x,y) θα είναι επίσης διάνυσμα (χρώμα).

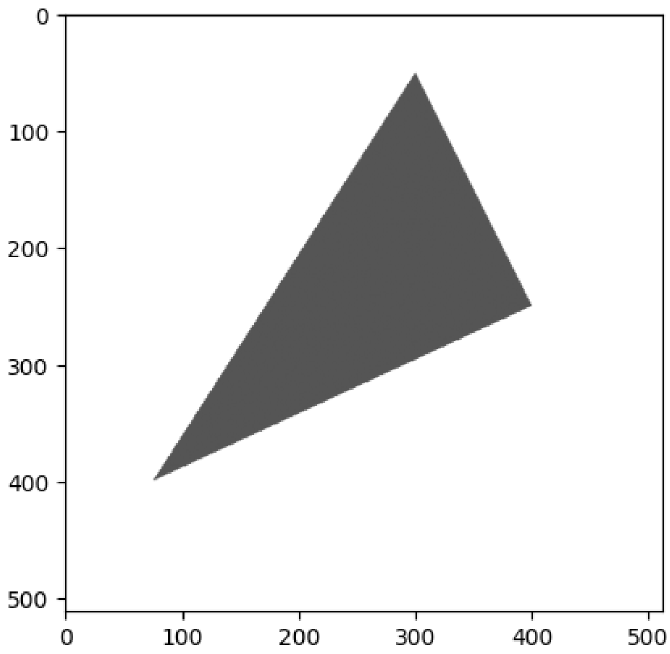
Για να οπτικοποιηθέι ο χρωματισμός της παραπάνω συνάρτησης θα χρησιμοποιήσω για παράδειγμα ένα τρίγωνο σε άσπρο καμβά 512×512 με συνεταγμένες κορυφών $(300, 50)$, $(75, 400)$, $(400, 250)$ και χρώματα $(0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 0)$ αντίστοιχα, δηλαδή μπλέ πράσινο και κόκκινο. Ο μέσος όρος των τριών αυτών χρωμάτων παράγει γκρι, επομένως περιμένουμε να δούμε ένα γκρι τρίγωνο με τις παραπάνω συντεταγμένες.

Η εικόνα που προβάλεται με τον κώδικα σε python που εξιγείται στη συνέχεια, φαίνεται στο σχήμα 1.

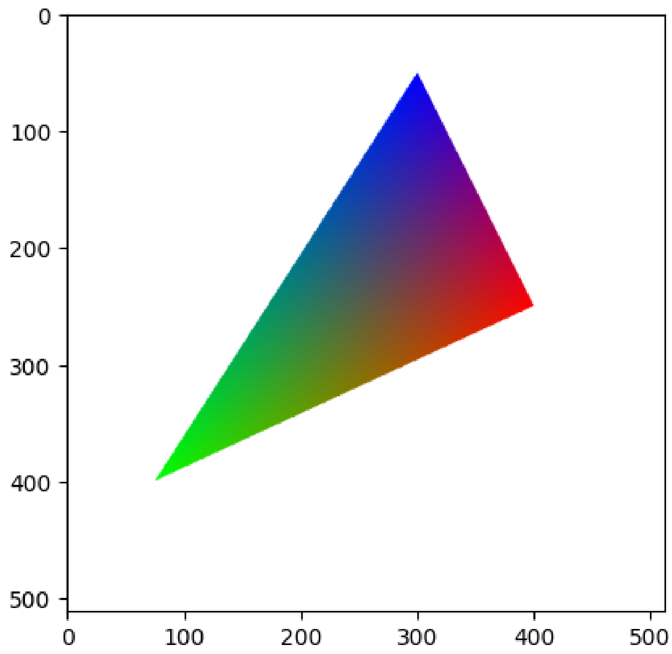
5.2 Η Συνάρτηση g_shading

Στη μέθοδο gouraud το χρώμα του εκάστοτε pixel υπολογόζεται με τη συνάρτηση double_vector_interp. Επομένως:

image(x,y) = double_vector_interp((x,y),LeftBoundaryPoint,RightBoundaryPoint)



Σχήμα 1: flat shading



Σχήμα 2: gouraud shading

Για την οπτικοποίηση της αυτής της συνάρτησης θα χρησιμοποιηθεί το ίδιο τρίγωνο με τα ίδια χρώματα με τη συνάρτηση f_shading. Τώρα περιμένουμε το τρίγωνο να μην έχει σταθερό χρώμα, αλλά να έχει στις κορυφές τα χρώματα που τους δόθηκαν και ενδιάμεσα μια διαβάθμιση.

Το αποτέλεσμα φαίνεται στο σχήμα 2.

6 Η συνάρτηση render_img

Συνάρτηση που δέχεται ως ορίσματα τους πίνακες faces, vertices, vcolors, depth και την τιμή shading και παράγει την εικόνα που δημιουργούν τα τρίγωνα που δίνονται στους πίνακες με τον παρακάτω τρόπο. faces είναι πίνακας που περιέχει για κάθε τρίγωνο τις κορυφές του ως αναφορές στους υπόλοιπους πίνακες. Ο πίνακας vertices περιέχει τις συντεταγμένες των κορυφών, ο πίνακας vcolors τα διανύσματα των χρωμάτων τους, ο πίνακας depth το βάθος τους. Τέλος η τιμή shading είναι είται f είται g ανάλογα με το είδος χρωματισμού ($flat$ ή $gouraud$).

Θα δημιουργηθεί αρχικά μια συνάρτηση (Find_Depth) που υπολογίζει το βάθος κάποιου τριγώνου ως το κέντρο μάζας των κορυφών του. Δηλαδή:

$$TriangleDepth = \frac{Depth(vertex1) + Depth(vertex2) + Depth(vertex3)}{3}$$

Στη συνάρτηση render_img αρχικά δημιουργείται ένας άσπρος καμβάς 512×512, υπολογίζεται το βάθος κάθε τριγώνου με τη συνάρτηση Find_Depth και τα τρίγωνα (faces) ταξινομούνται ως προς το βάθος τους από το πιο μακρινό στο πιο κοντινό. Έπειτα για κάθε τρίγωνο (face) καλείται η αντίστοιχη συνάρτηση f_shading ή g_shading ανάλογα με την τιμή shading.

7 Κώδικας σε Python

Οι παραπάνω αλγόριθμοι υλοποιήθηκαν με την γλώσσα προγραμματισμού python.

Στον κώδικα έγιναν οι εξίς παραδοχές:

- οι θέσεις των pixel ακολουθούν αρίθμηση από το 0 μέχρι το 511, όπου το pixel (0,0) βρίσκεται στο πάνω αριστερά άκρο της εικόνας

- τα διανύσματα αντιπροσωπεύονται με πίνακες 2 στοιχείων, όπου το πρώτο είναι η τετμημένη και το δεύτερο η τεταγμένη. Έτσι, στη στιγμή που χρωματίζουμε κάποιο pixel (x,y) χρωματίζουμε στη θέση image[y][x] ώστε η εικόνα να έχει τον σωστό προσανατολισμό.

Άρχικά όρισα δύο κλάσεις, αυτή των ευθειών (class line) και αυτή των οριακών σημείων (class BoundaryPoint). Οι κλάσεις αυτές ορίζονται σε ξεχωριστά αρχεία.

7.1 class line

Η κλάση line έχει παραμέτρους τα δύο σημεία απο τις κορυφές του τριγώνου που την ορίζουν, τα χρώματα τους και έναν χαρακτηριστικό αριθμό για να την εκφράζει. Από αυτά τα δύο σημεία υπολογίζονται τα $ymin, ymax, xmin, xmax$ η κλίση $slope$ και τα $pointmin, pointmax$ της ευθείας και αποθηκεύονται ως attributes.

7.2 class BoundaryPoint

Στην κλάση οριακών σημείων αποθηκεύονται ως attributes τα χαρακτηριστικά $point$, $slope$, $line\_number$, $line$ του οριακού σημείο καθώς χρειάζονται για τη ροή του κώδικα. $point$ είναι οι συντεταγμένες του σημείου, $slope$ η κλίση της ευθείας στην οποία ανήκει, $line\_number$ ο χαρακτηριστικός αριθμός της ευθείας και $line$ η αναφορά στο αντικείμενο της κλάσης $line$ στην οποία ανήκει το σημείο.

Οι παραπάνω δύο κλάσεις βοηθούν στην υλοποίση των επόμενων συναρτήσεων.

7.3 Οι συναρτήσεις vector_interp, double_vector_interp, f_shading, g_shading, render_img

Αποτελούν την υλοποίηση στην python των αντίστοιχων προαναφερθέντων συναρτήσεων. Είναι σχεδόν ένα προς ένα αντιστοιχεία του ψευδοκώδικα, περισσότερη λεπτομέρεια για τη λειτουργία του κώδικα python μπορείτε να βρείτε στα σχόλια του κώδικα όπου επεξειγούνται επακριβώς οι εντολές.

Για τις πράξεις με πίνακες χρησιμοποιήθηκε η βιβλιοθήκη numpy.

7.4 Αρχεία demo_f και demo_g

Είναι τα αρχεία που θα τρέξουν για την αποθήκευση σε εικόνες του δοσμένου σχήματος στο αρχείο hw1.npy.

Φορτώνονται τα δεδομένα σε αντίστοιχους πίνακες faces, vertices, vcolors, depth και καλείται η συνάρτηση render_img με τιμή shading $'f'$ στο demo_f και $'g'$ στο demo_g.

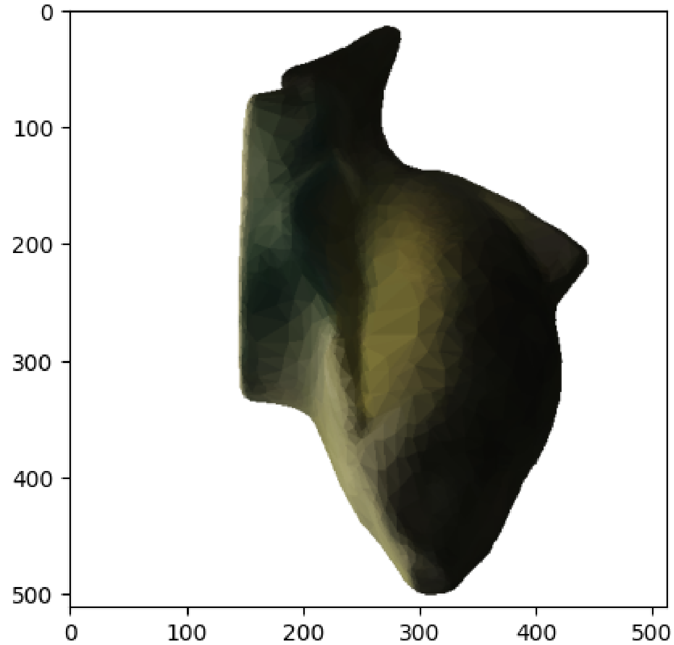
Ο παραγόμενος πίνακας προβάλλεται και αποθηκεύεται ως εικόνα με τη χρήση της βιβλιοθήλης matplotlib.

7.5 Ονόματα αρχείων

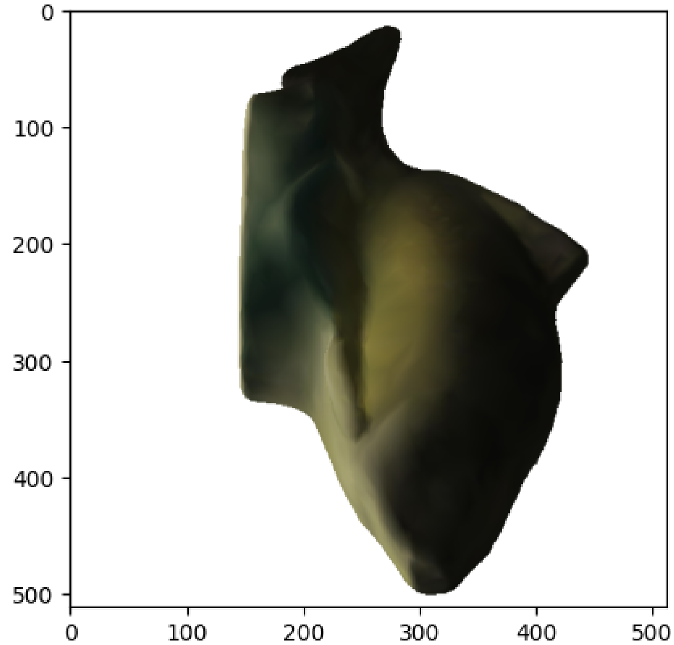
Ο ορισμός της κλάσης line γίνεται στο αρχείο με όνομα Line_Class.py, ο ορισμός της κλάσης BoundaryPoint στο αρχείο BoundaryPoint_Class.py, οι συναρτήσεις vector_interp και double_vector_interp ορίζονται στο αρχείο vector_interpolation.py, οι συναρτήσεις f_shading και g_shading στα αρχεία flat_shading.py και Gouraud_shading.py αντίστοιχα, η συνάρτηση render_img στο αρχείο render_image.py και τέλος υπάρχουν τα αρχεία demo_f.py και demo_g.py.

8 Αποτελέσματα

Οι εικόνες που παράγονται από τα δεδομένα του αρχείου hw1.npy με μεθόδους σκίασης flat και gouraud φαίνονται στα σχήματα 3 και 4 αντίστοιχα.



Σχήμα 3: Flat Shading



Σχήμα 4: Gouraud Shading

Με την flat shading επειδή κάθε τρίγωνο έχει σταθερό χρωματισμό και διαφορετικό από τα υπόλοιπα, ξεχωρίζουν ξεκάθαρα τα τρίγωνα μεταξύ τους, ενώ με την gouraud shading λόγο της γραμμικής παρεμβολής που διαβαθμίζει τις μεταβολές μεταξύ χρωμάτων, δεν ξεχωρίζουν τα μεμονομένα τρίγωνα και φαίνεται ένα πιο λείο σχήμα.