ΓΡΑΦΙΚΗ ΜΕ ΥΠΟΛΟΓΙΣΤΕΣ

ΕΡΓΑΣΙΑ ΙΙ - ΑΝΑΦΟΡΑ

ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2023

ΑΝΑΣΤΑΣΙΟΣ ΣΤΕΡΓΙΟΥ | 10079
Τ.Η.Μ.Μ.Υ. Α.Π.Θ. | anasster@ece.auth.gr

ΕΙΣΑΓΩΓΗ

Το παρόν έγγραφο αποτελεί μία αναφορά στο δεύτερο μέρος της εργασίας του μαθήματος *Γραφική με Υπολογιστές* με θέμα *«Μετασχηματισμοί και Προβολές»*. Στην αρχή, θα παρουσιαστεί εν συντομία η λειτουργία του προγράμματος, στη συνέχεια θα περιγραφεί αναλυτικότερα ο τρόπος λειτουργίας της κάθε συνάρτησης που υλοποιήθηκε, και τέλος θα παρουσιαστούν τα αποτελέσματα που παράχθηκαν.

ΛΕΙΤΟΥΡΓΙΑ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Στο πρόγραμμα αυτό, δημιουργήθηκαν αρχικά συναρτήσεις που πραγματοποιούν μετασχηματισμούς σε ένα σημείο του τρισδιάστατου χώρου, ή και σε ένα σύνολο τρισδιάστατων σημείων. Συγκεκριμένα, οι μετασχηματισμοί είναι μετατόπιση ενός σημείου κατά ένα διάνυσμα $\vec{t}$, περιστροφή ενός σημείου κατά μία γωνία θ rads δεξιόστροφα, περί άξονα παράλληλο σε διάνυσμα $\vec{u}$, ή συνδυασμός αυτών των δύο. Επιπλέον, υπάρχει και συνάρτηση αλλαγής συστήματος συντεταγμένων μέσω περιστροφής ή (και) μετατόπισης των αξόνων. Στη συνέχεια, δημιουργήθηκαν συναρτήσεις που πραγματοποιούν προοπτική προβολή ενός σημείου, ή ενός συνόλου από σημεία, στο πέτασμα μίας εικονικής κάμερας, μέσω υπολογισμού των μοναδιαίων διανυσμάτων της κάμερας από κάποιες παραμέτρους. Τέλος, υλοποιήθηκε μία συνάρτηση όπου αντιστοιχεί τις πραγματικές συντεταγμένες των σημείων στο πέτασμα, στις αντίστοιχες διακριτές και ακέραιες τιμές τους πάνω σε ένα πλέγμα από pixels. Τα σημεία αυτά κατόπιν χρωματίζονται μέσω της μεθόδου *gouraud shading* η οποία υλοποιήθηκε στην πρώτη εργασία. Επιπλέον, για την υλοποίηση χρωματισμού του αντικειμένου δημιουργήθηκαν δύο βοηθητικές συναρτήσεις για την ελάττωση της υπολογιστικής πολυπλοκότητας του προγράμματος. Κατά την επίδειξη του προγράμματος, υλοποιούνται κατά σειρά τρεις μετασχηματισμοί πάνω στα σημεία του αντικειμένου προς απεικόνιση. Σημειώνεται ότι καθώς στην πρώτη εργασία, τα αποτελέσματα της εργασίας δεν ήταν πλήρως σωστά λόγω λαθών στον κώδικα, ο κώδικας κάποιων συναρτήσεων της πρώτης εργασίας που θα χρησιμοποιηθεί στη δεύτερη έχει ορισμένες αλλαγές.

ΠΕΡΙΓΡΑΦΗ ΤΩΝ ΣΥΝΑΡΤΗΣΕΩΝ

Από εδώ και κάτω, θα περιγραφεί η λειτουργία όλων των συναρτήσεων, καθώς και παραδοχές που γίναν κατά την υλοποίησή τους.

I. R = rot_mat(theta, u)

Πρόκειται για μία συνάρτηση που παράγει τον πίνακα R με τον οποίο θα πολλαπλασιαστεί ένα σημείο του τρισδιάστατου χώρου κατά την περιστροφή του γύρω από έναν άξονα παράλληλο σε ένα μοναδιαίο διάνυσμα $\hat{u}$, κατά μία γωνία θ με αντιωρολογιακή φορά.

Στον κώδικα, η μεταβλητή `theta` είναι η γωνία θ (σε ακτίνια) κατά την οποία θα περιστραφεί αντιωρολογιακά το σημείο. Η μεταβλητή u είναι το διάνυσμα $\vec{u}$ στο οποίο είναι παράλληλος ο άξονας περιστροφής. Κατά την υλοποίηση της συνάρτησης, το διάνυσμα u κανονικοποιείται ως προς το μέτρο του, ώστε να είναι σίγουρα μοναδιαίο και να υλοποιηθεί σωστά η περιστροφή. Ο πίνακας R παράγεται σύμφωνα με τον τύπο του *Rodrigues*:

$$R = \begin{bmatrix} (1-\cos\theta)u_x^2 + \cos\theta & (1-\cos\theta)u_x u_y - \sin\theta\, u_z & (1-\cos\theta)u_x u_z + \sin\theta\, u_y \\ (1-\cos\theta)u_y u_x + \sin\theta\, u_z & (1-\cos\theta)u_y^2 + \cos\theta & (1-\cos\theta)u_y u_z - \sin\theta\, u_x \\ (1-\cos\theta)u_z u_x - \sin\theta\, u_y & (1-\cos\theta)u_z u_y + \sin\theta\, u_x & (1-\cos\theta)u_z^2 + \cos\theta \end{bmatrix}$$

όπου u_x, u_y και u_z είναι οι συνιστώσες του διανύσματος $\hat{u}$ κατά τους τρεις άξονες μετά την κανονικοποίησή του.

II. cq = rotate_translate(cp, theta, u, A, t)

Πρόκειται για μία συνάρτηση όπου περιστρέφει και μετατοπίζει ένα σύνολο τρισδιάστατων σημείων σημείων με συντεταγμένες c_p. Συγκεκριμένα, τα σημεία περιστρέφονται αριστερόστροφα ως προς άξονα που διέρχεται από σημείο A και είναι παράλληλος στο μοναδιαίο διάνυσμα $\hat{u}$ κατά θ rads, και στη συνέχεια μετατοπίζονται κατά ένα διάνυσμα $\vec{t}$. Ο συνολικός μετασχηματισμός αντιστοιχεί στη μαθηματική σχέση:

$$c_q = R \cdot c_p + (I - R) \cdot d$$

όπου d το διάνυσμα που συνδέει την αρχή των αξόνων με το σημείο A, R ο πίνακας περιστροφής και c_q οι συντεταγμένες των μετασχηματισμένων σημείων.

Στον κώδικα, τα σημεία `cp` αρχικά μετατοπίζονται κατά `d`, όπου `d` ένα διάνυσμα με τις συντεταγμένες του σημείου A. Στη συνέχεια, μέσω της συνάρτησης `rot_mat` υπολογίζεται ο πίνακας περιστροφής R για γωνία `theta` και διάνυσμα u. Στη συνέχεια, για ένα σύνολο σημείων μεγέθους N, πραγματοποιούνται με τη σειρά οι εξής μετασχηματισμοί:

- Μετατόπιση του συστήματος συντεταγμένων με κέντρο το σημείο A
- Περιστροφή των σημείων `cp` μέσω του πίνακα R
- Επαναφορά στο αρχικό σύστημα συντεταγμένων
- Μετατόπιση των περιεστραμμένων σημείων κατά `t`

III. dp = change_coordinate_system(cp, R, c0)

Πρόκειται για μία συνάρτηση όπου αλλάζει το σύστημα συντεταγμένων ενός συνόλου N σημείων c_p, περιστρέφοντας τους άξονες σύμφωνα με έναν πίνακα περιστροφής R και μετατοπίζοντάς τους κατά ένα διάνυσμα $\overrightarrow{\mathrm{v}_0}$ με συντεταγμένες c_0. Ο μετασχηματισμός αυτός γίνεται σύμφωνα με τη σχέση:

$$d_p = R^T\left(c_p - c_0\right)$$

για κάθε σημείο c_p του συνόλου. Η σχέση αυτή προκύπτει καθώς η διαδικασία μετασχηματισμού των αξόνων είναι η ακριβώς αντίστροφη διαδικασία μετασχηματισμού των σημείων, και επιπλέον ο πίνακας R είναι πίνακας $SO(3)$, δηλαδή $R^T = R^{-1}$.

Στον κώδικα, υλοποιείται ακριβώς αυτή η διαδικασία, όπου `cp` ένας πίνακας $N \times 3$ με τις συντεταγμένες των σημείων, `R` ο πίνακας περιστροφής, και `c0` οι συντεταγμένες του $\overrightarrow{\mathrm{v}_0}$.

IV. p2d, depth = pinhole(f, cv, cx, cy, cz, p3d)

Πρόκειται για μία συνάρτηση όπου υλοποιεί τη διαδικασία της προοπτικής προβολής N τρισδιάστατων σημείων με συντεταγμένες c_p στο πέτασμα μιας κάμερας με μοναδιαία $\hat{x}_c$, $\hat{y}_c$, $\hat{z}_c$, και με κέντρο $\overrightarrow{\mathrm{v}_c}$, ως προς το σύστημα συντεταγμένων του κόσμου WCS. Το πέτασμα έχει εστιακή απόσταση f από τον φακό της κάμερας, και ο μετασχηματισμός της προοπτικής προβολής γίνεται σύμφωνα με τη σχέση:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = -f \cdot \begin{bmatrix} x/z \\ y/z \end{bmatrix}$$

Ωστόσο, πρώτα θα πρέπει να βρεθούν οι συντεταγμένες των σημείων c_p ως προς το σύστημα συντεταγμένων της κάμερας CCS, κάτι το οποίο υλοποιείται μέσω του μετασχηματισμού:

$$c_{p,CCS} = R^T\left(c_{p,WCS} - c_{\mathrm{v},WCS}\right)$$

όπου $R = [\hat{x}_c\ \hat{y}_c\ \hat{z}_c]$ ο πίνακας περιστροφής του συστήματος συντεταγμένων WCS.

Κατά την υλοποίηση της συνάρτησης, χρησιμοποιήθηκε η παραδοχή ότι το πέτασμα βρίσκεται μπροστά από τον φακό, με αποτέλεσμα να προκύπτουν αρνητικά πρόσημα στο πρόγραμμα κατά τον υπολογισμό των συντεταγμένων.

Στον κώδικα, η υλοποίηση του μετασχηματισμού γίνεται μέσω της συνάρτησης `change_coordinate_system` χρησιμοποιώντας τον πίνακα `p3d` ως τα σημεία που θα μετασχηματιστούν οι συντεταγμένες τους, `R` ο πίνακας περιστροφής που ορίζεται από τα σημεία `cx`, `cy` και `cz` ως προς το σημείο `cv`. Τα σημεία λοιπόν, έχουν υπολογιστεί πλέον ως προς το CCS. Στη συνέχεια, υλοποιείται ο μετασχηματισμός προοπτικής προβολής, και δημιουργείται ένας νέος πίνακας $N \times 2$ δισδιάστατων σημείων με τις συντεταγμένες τους στο πέτασμα (ως προς το κέντρο του πετάσματος) `p2d`. Επιπλέον, δημιουργείται και ένας πίνακας `depth` διαστάσεων $N \times 1$ με το βάθος (z συντεταγμένη) των σημείων `p3d` ως προς την κάμερα.

V. p2d, depth = camera_looking_at(f, cv, ck, cup, p3d)

Πρόκειται για μία συνάρτηση όπου υλοποιεί παρομοίως τη διαδικασία προοπτικής προβολής N σημείων του τρισδιάστατου χώρου, αλλά αυτή τη φορά γνωρίζοντας τα διανύσματα θέσης της κάμερας C, του σημείου θέασης K, και του *up vector* $\vec{u}$ της κάμερας. Από αυτά τα διανύσματα, θα υπολογιστούν τα μοναδιαία της κάμερας μέσω των εξής σχέσεων:

$$\hat{z}_c = \frac{\overrightarrow{CK}}{\left|\overrightarrow{CK}\right|}$$

$$\hat{y}_c = \frac{\vec{u} - \langle \vec{u}, \hat{z}_c \rangle \hat{z}_c}{|\vec{u} - \langle \vec{u}, \hat{z}_c \rangle \hat{z}_c|}$$

$$\hat{x}_c = \frac{\hat{y}_c \times \hat{z}_c}{|\hat{y}_c \times \hat{z}_c|}$$

και θα γίνει υπολογισμός της προοπτικής προβολής των σημείων.

Στον κώδικα, υλοποιείται αυτή ακριβώς η διαδικασία, υπολογίζοντας τις συντεταγμένες των `cx`, `cy` και `cz` μέσω των `cv`, `ck` και `cup`. Στη συνέχεια, καλείται η συνάρτηση `pinhole` και υπολογίζονται και πάλι τα σημεία `p2d` και ο πίνακας `depth`.

VI. n2d = rasterize(p2d, rows, cols, h, w)

Πρόκειται για μία συνάρτηση που υπολογίζει τις ακέραιες συντεταγμένες (n, m) των σημείων πάνω στο πέτασμα. Θεωρώντας ότι οι ακέραιες συντεταγμένες έχουν σημείο αρχής το σημείο πάνω αριστερά στο πλέγμα των pixels, οι ακέραιες συντεταγμένες ενός pixel σε πλέγμα διαστάσεων $rows \times cols$, που αντιστοιχούν στις συντεταγμένες του σημείου στο πέτασμα της κάμερας διαστάσεων $H \times W$, θα είναι:

$$\begin{bmatrix} n \\ m \end{bmatrix} = \begin{bmatrix} cols/2 \\ rows/2 \end{bmatrix} + \begin{bmatrix} x \cdot cols/H \\ -y \cdot rows/W \end{bmatrix}$$

Ακριβώς με αυτόν τον τρόπο στον κώδικα μετασχηματίζονται τα σημεία `p2d`, κρατώντας το *ακέραιο μέρος* των μετασχηματισμένων σημείων. Εδώ είναι καλό να τονιστεί, οτί κατά τη μετατροπή των «αναλογικών» συντεταγμένων `p2d` σε «ψηφιακές» `n2d`, θα μπορούσε εναλλακτικά το αποτέλεσμα να στρογγυλοποιηθεί στον πλησιέστερο ακέραιο, μία διαδικασία που απορρίφθηκε, καθώς θα μπορούσε να οδηγήσει σε ενδεχόμενο σφάλμα. Το σφάλμα θα οφειλόταν στο γεγονός ότι η στρογγυλοποίηση ενδεχομένως να οδηγούσε σε ακέραιο μεγαλύτερο των διαστάσεων του πλέγματος κατά μία μονάδα (π.χ. στρογγυλοποίηση του 512.6 στο 513 σε πλέγμα διαστάσεων 512×512).

VII. img = render_object(p3d, faces, vcolors, h, w, rows, cols, f, cv, ck, cup)

Πρόκειται για μία συνάρτηση όπου χρωματίζει ένα αντικείμενο. Συγκεκριμένα, τα τρισδιάστατα σημεία γίνονται δισδιάστατα μέσω της συνάρτησης `camera_looking_at`, δημιουργώντας τον πίνακα `p2d`, καθώς και τον πίνακα `depth` που θα χρησιμοποιηθεί στη συνέχεια. Οι συντεταγμένες αυτές, που έχουν πραγματικές τιμές, μέσω της συνάρτησης `rasterize` μετατρέπονται σε ακέραιες. Τα τρίγωνα τέλος, χρωματίζονται μέσω της συνάρτησης `render` με τη μέθοδο gouraud, καλώντας την `gourauds` από την πρώτη εργασία.

Οι συναρτήσεις της πρώτης εργασίας τροποποιήθηκαν ελαφρώς ώστε να είναι υπολογιστηκά πιο απλές, και επιπλέον δημιουργήθηκαν ορισμένες συναρτήσεις βοηθητικού χαρακτήρα, οι οποίες δεν περιλαμβάνονταν στην πρώτη εργασία και θα παρουσιαστούν εδώ:

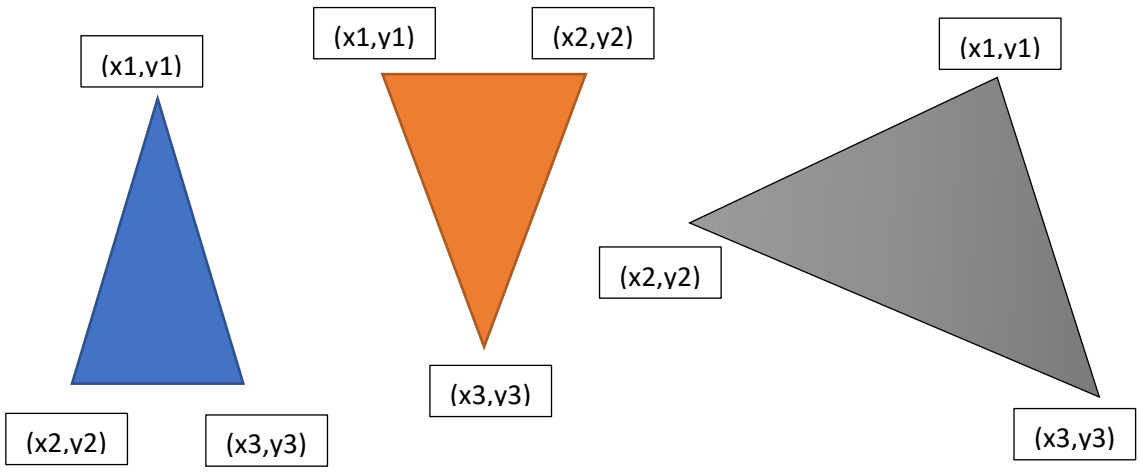
I. `sorted_points = sort_points(points)`

Πρόκειται για μία συνάρτηση η οποία ταξινομεί τις δισδιάστατες κορυφές ενός τριγώνου ως εξής:

- Ταξινόμηση με την τεταγμένη τους σε αύξουσα σειρά
- Εάν δύο σημεία έχουν ίδια τεταγμένη, τότε ταξινομούνται με την τετμημένη τους σε αύξουσα σειρά.

Με αυτόν τον τρόπο, εξασφαλίζεται η διάταξη των σημείων του τριγώνου με βάση τους δείκτες του.

Στον κώδικα, επικαλείται η συνάρτηση `numpy.argsort` ώστε να παραχθεί ένας πίνακας με τους δείκτες των ταξινομημένων τεταγμένων. Οι δείκτες αυτοί ταξινομούν και τις αντίστοιχες τετμημένες. Σε περίπτωση ισότητας τεταγμένων, αλλάζουν θέσεις οι τετμημένες μεταξύ τους εφόσον χρειάζεται. Γραφικά, το επιθυμητό αποτέλεσμα παρουσιάζεται παρακάτω:



Οι δείκτες των τριγώνων αντιστοιχούν στην ταξινόμηση των συντεταγμένων τους με αύξουσα σειρά, θεωρώντας ότι ο καμβάς έχει ως κεντρικό σημείο την πάνω αριστερά άκρη του (δηλαδή το y αυξάνεται προς τα κάτω).

II. `effective_points = calculate_effective_points(vertices)`

Πρόκειται για μία συνάρτηση η οποία υπολογίζει τα ενεργά σημεία ενός τριγώνου, και επιστρέφει έναν πίνακα $N \times 2$, όπου περιέχει τις τετμημένες των σημείων τομής των πλευρών του τριγώνου με την οριζόντια scanline y. Είναι δεδομένο ότι, λόγω του ότι το τρίγωνο είναι ένα κυρτό πολύγωνο, μεταξύ των σημείων y_{min} και y_{max} θα υπάρχουν πάντοτε δύο σημεία τομής.

Στην περίπτωση οριζόντιας πλευράς στις θέσεις y_{min} ή y_{max}, αποθηκεύονται στον πίνακα οι τετμημένες των κορυφών του τριγώνου που συνιστούν την πλευρά αυτή. Για να διευκολυνθεί η διαδικασία αυτή, οι κορυφές `vertices` ταξινομούνται με κλήση της συνάρτησης `sort_points`.

III. `updated_canvas = gourauds(canvas, vertices, vcolors)`

Πρόκειται για τη συνάρτηση χρωματισμού ενός τριγώνου με τη μέθοδο *gouraud*. Συγκεκριμένα, η συνάρτηση χρωματίζει ένα τρίγωνο με κορυφές `vertices`, κάθε μία από τις οποίες έχει χρώμα που αντιστοιχεί σε μία σειρά του πίνακα `vcolors`, ο οποίος περιέχει στις σειρές του τις RGB συνιστώσες της κάθε κορυφής. Το τρίγωνο του καμβά `canvas`, αφού οι κορυφές του ταξινομηθούν μέσω της `sort_points` και αντιστοιχηθούν τα χρώματα στις νέες κορυφές, υπολογίζονται τα effective points μέσω της `calculate_effective_points` και χρωματίζονται με βάση τη μέθοδο gouraud, λαμβάνοντας υπόψιν όλες τις πιθανές μορφές τριγώνων (οριζόντιες ακμές πάνω και κάτω). Στη συνάρτηση αυτή, καλείται και η `interpolate_vectors` της πρώτης εργασίας, η οποία παρέμεινε απαράλλακτη.

IV. `img = render(verts2d, faces, vcolors, depth)`

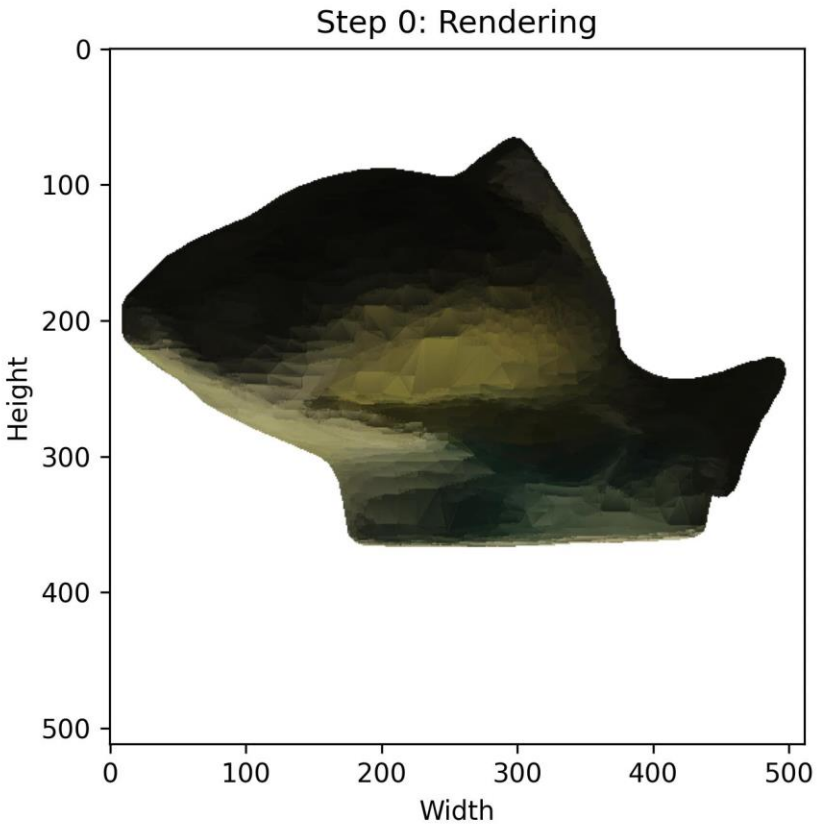
Η συνάρτηση `render` είναι ίδια με αυτήν της πρώτης εργασίας, με τη διαφορά ότι εδώ πέρα δεν υπάρχει όρισμα `shade_t` που να καλεί τη συνάρτηση να επιλέξει μέθοδο χρωματισμού (flat ή gouraud), καθώς στην παρούσα εργασία ζητείται αποκλειστικά η επίδειξη της gouraud. Επομένως, αφαιρέθηκε το block όπου ελέγχονταν η τιμή της μεταβλητής αυτής, για απλοποίηση και στοιχειώδη επιτάχυνση του κώδικα.

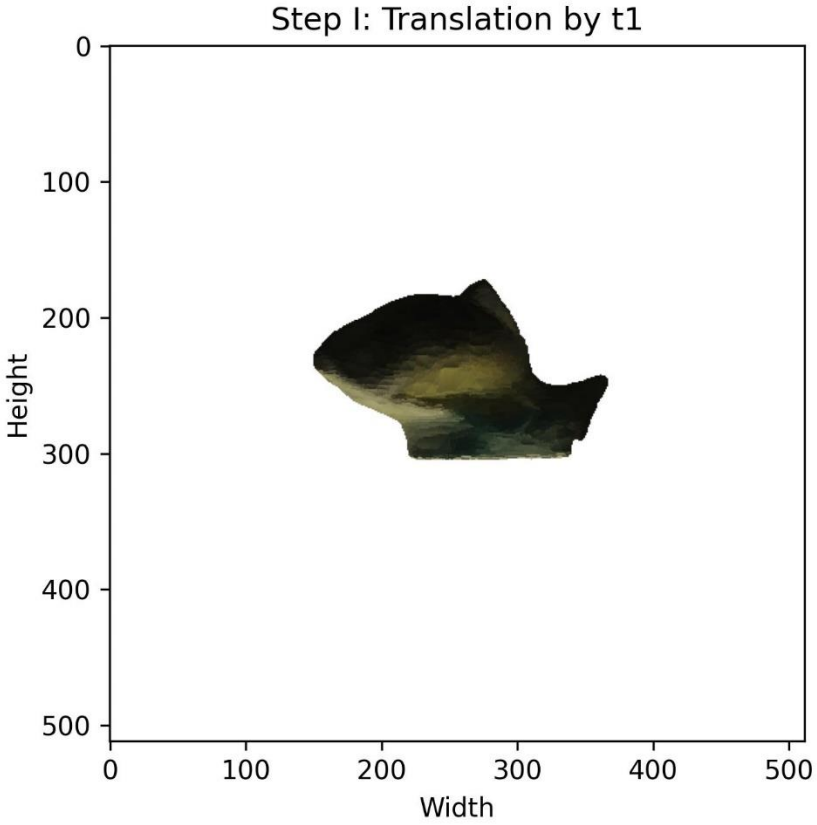
ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

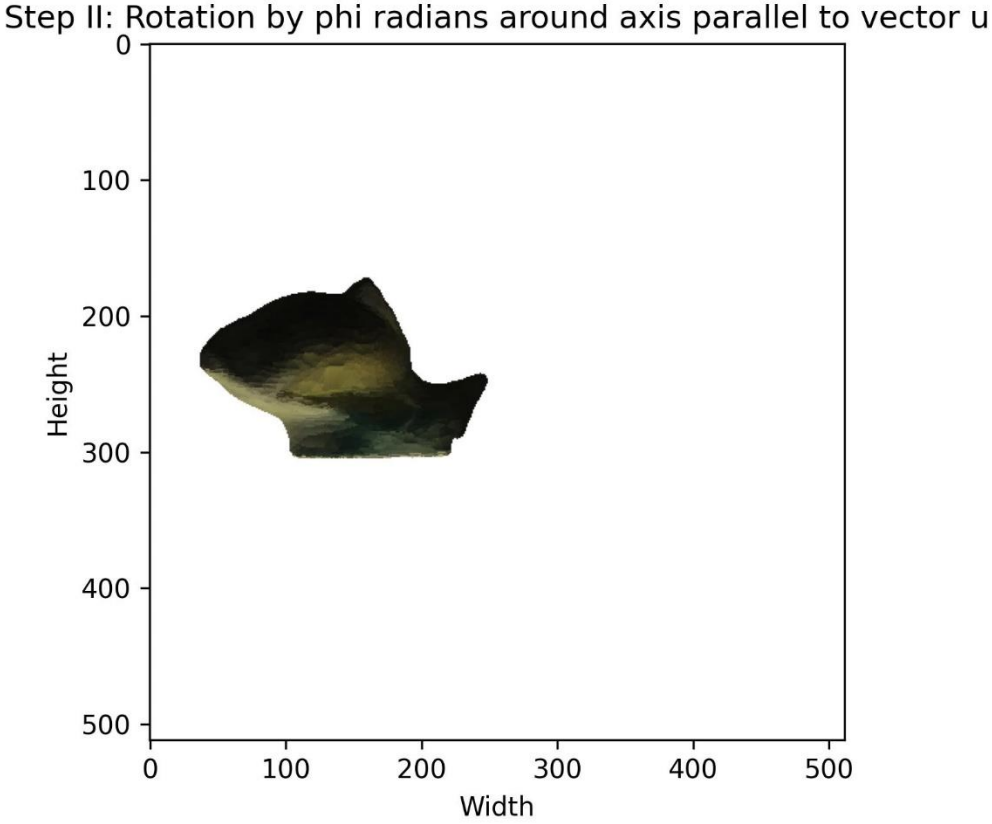
Για την επίδειξη λειτουργίας του προγράμματος, ζητείται μία σειριακή εκτέλεση ορισμένων μετασχηματισμών σε ένα αντικείμενο, και η φωτογράφιση του αντικειμένου και δισδιάστατη απεικόνισή του έπειτα από κάθε μετασχηματισμό. Συγκεκριμένα, ζητείται:

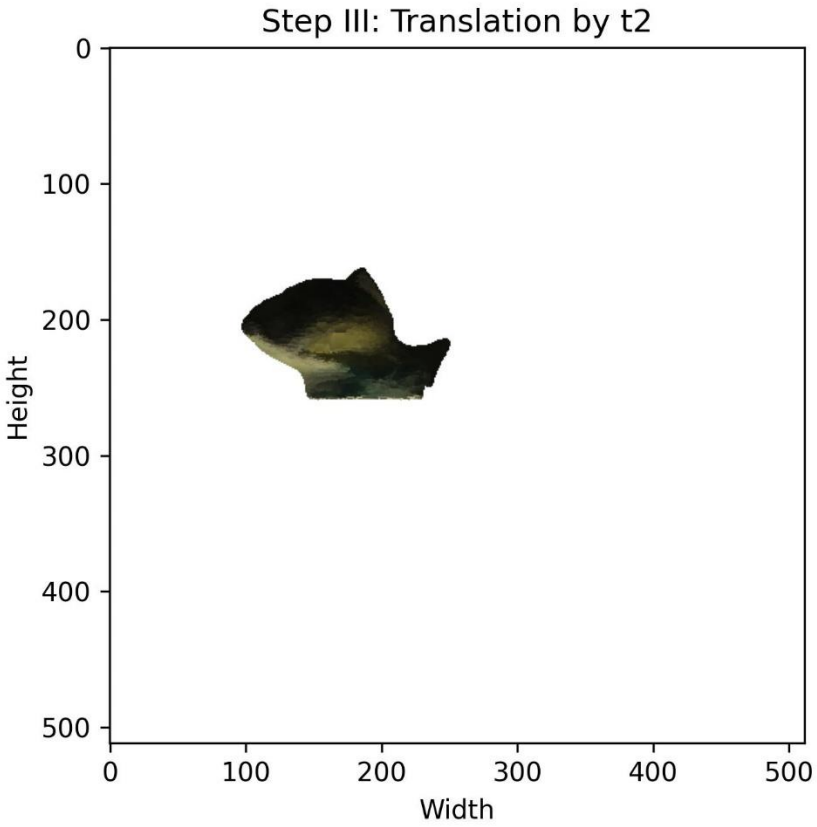
- Χρωματισμός του αντικειμένου με τη μέθοδο gouraud
- Μετατόπιση του αντικειμένου κατά ένα διάνυσμα $\overrightarrow{t_1}$
- Περιστροφή του μετατοπισμένου αντικειμένου κατά φ rads αριστερόστροφα, γύρω από άξονα παράλληλο σε διάνυσμα $\hat{u}$ που διέρχεται από το σημείο θέασης (ταυτίζεται με την αρχή του WCS) με συντεταγμένες $(0,0,0)$
- Μετατόπιση του περιεστραμμένου και μετατοπισμένου αντικειμένου κατά ένα διάνυσμα $\overrightarrow{t_2}$

Όλα τα διανύσματα, η γωνία, καθώς και οι παράμετροι της κάμερας διατίθενται από το αρχείο `h2.npy`. Οι φωτογραφίσεις του αντικειμένου σε κάθε βήμα αποθηκεύονται με όνομα `stepi.jpg` όπου i ο αριθμός του βήματος, και παρουσιάζονται παρακάτω:




Step I: Translation by t1
Height
Width
0
100
200
300
400
500


Step II: Rotation by phi radians around axis parallel to vector u
Height
Width
0
100
200
300
400
500



Οι μετασχηματισμοί πραγματοποιήθηκαν θεωρώντας διαστάσεις πετάσματος $H = W = 15\ in$, εστιακή απόσταση πετάσματος $f = 70\ in$ και μέγεθος καμβά $N = M = 512\ pixels$. Επιπλέον, με κάθε εκτέλεση του demo τυπώνεται ο χρόνος εκτέλεσης των μετασχηματισμών. Τέλος, να τονιστεί ότι οι εικόνες με τα αποτελέσματα αποθηκεύονται στον φάκελο του προγράμματος (μαζί με τα αρχεία .py).

ΤΕΛΟΣ ΑΝΑΦΟΡΑΣ