Γραφική με Υπολογιστές
Εργασία 2: Μετασχηματισμοί και Προβολές

Φίλιππος Ρωσσίδης ΑΕΜ:10379

9 Μαΐου 2024

1 Εισαγωγή

Στην εργασία ζητούμενο είναι να δημιουργηθούν τεχνικές για μετασχηματισμούς περιστροφής και μετατόπισης σημείων, και τεχικές προβολής αντικειμένων δοσμένων των συντεταγμένων, πλευρών και χρωμάτων του, μέσω μιας κάμερας δοσμένων των συντεταγμένων της.

2 Κλάση Transform

Η κλάση Transform περιγράφει έναν affine μετασχηματισμό. Περιέχει ένα attribute $self.mat$ που δηλώνει των 4×4 πίνακα που περιγράφει τον μετασχηματισμό. Περιέχει επίσης 4 συναρτήσεις:

- Η συνάρτηση $\_\_init\_\_(self)$ καλέιται όταν δημιουργείται ένα αντικείμενο της κλάσης και αρχικοποιεί τη τιμή του $self.mat$ στον μοναδιαίο πίνακα $I_{4\times4}$.
- Η συνάρτηση $rotate(self, theta, u)$ ενημερώνει τον πίνακα μετασχηματισμού ώστε να πραγματοποιεί τον affine μετασχηματισμό που ήδη πραγματοποιούσε και έπειτα μια περιστροφή κατά θ ακτίνια με άξονα περιστροφής αυτόν που ορίζει το μοναδιαίο διάνυσμα $\hat{\mathbf{u}}$. Αυτό επιτυχγάνεται με πολλαπλασιαμό του υπάρχοντος πίνακα με τον πίνακα περιστοφής R_h σε ομογενής συντεταγμένες ως εξίς: $self.mat = R_h \cdot self.mat$. Ο πίνακας R_h υπολογίζεται μέσω του τύπου του Rodrigues.
- Η συνάρτηση $translate(self, t)$ ενημερώνει τον πίνακα ώστε να εφαρμόζει μετασχηματισμό μετατόπισης κατά $\mathbf{t}$ έπειτα των προηγούμενων μετασχηματισμών. Αυτό γίνεται με τη πράξη

$$self.mat = T_h \cdot self.mat, \qquad T_h = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Η συνάρτηση $transform\_pts(self, pts)$ επιστρέφει ένα πίνακα που περιέχει τα σημεία pts μετασχηματισμένα κατά τον πίνακα μετασχηματισμού. Πρώτα τα μετατρέπει σε ομογενείς συντεταγμένες, έπειτα πολλαπλασιάζει με τον πίνακα μετασχηματισμού και τέλος επιστρέφει τις καρτεσιανές συντεταγμένες.

3 Γενική περιγραφή λειτουργίας

Θα υλοποιηθούν 5 συναρτήσεις που θα πραγματοποιούν τη σκίαση (προβολή) αντικειμένων από την οπτική γωνία μιας κάμερας. Η επισκόπηση της λειτουργίας έχει ως εξίς:

- Από τα σημεία και διάνυσμα που ορίζουν την κάμερα παράγονται οι πίνακες περιστροφής και μετατόπισης R, d που μετατρέπουν σημεία από τις παγκόσμιες συντεταγμένες WCS στο σύστημα της κάμερας, μέσω της συνάρτησης $lookat(eye, up, target) \to (R, d)$.

- Τα σημεία που ορίζουν τα αντικείμενα μετατρέπονται στις συντεταγμένες της κάμερας με τους παραπάνω πίνακες, μέχω της συνάρτησης $world2view(pts, R, c0) \rightarrow points\_transf$.
- Τα σημεία από τις συντεταγμένες της κάμερας προβάλονται στο επίπεδο προβολής, με το μοντέλο της προοπτικής προβολής. Πλέον τα σημεία είναι δισδιάστατα. Αυτό γίνεται με τη συνάρτηση $perspective\_project(pts, focal, R, t) \rightarrow (pts\_2d, depth)$. Επιστρέφονται οι συντεταγμένες σε δύο διαστάσεις και το βάθος τους (απόσταση από την κάμερα).
- Με τη συνάρτηση $rasterize(pts\_2d, plane\_w, plane\_h, res\_w, res\_h) \rightarrow pixels$ τα σημεία κβαντίζονται σε ακέραιες τιμές που αναλογούν σε pixel σε πλαίσιο $plane\_h \times plane\_w$ με αριθμό pixel $res_h \times res_w$.
- Τέλος στη συνάρτηση $render\_object$ υπολογίζονται τα βάθη των πλευρών ως το κέντρο βάρους των κορυφών τους, διαγράφονται όσες έχουν αρνητικό βάθος (είναι πίσω από την κάμερα) και οι υπόλοιπες προβάλλονται με σειρά από την πιο μακρυνή στην πιο κοντινή στην κάμερα, με τη χρήση της συνάρτησης $g\_shading$ και της μεθόδου αποκοπής (clipping) όπου χρειάζεται.

3.1 Η μέθοδος αποκοπής

Εάν ένα τρίγωνο έχει κάποιες κορυφές μέσα στο πλαίσιο προβολής και κάποιες έξω από αυτό, τότε με τη διαγραφή των κορυφών που βρίσκονται έξω, εξαφανίζεται το τρίγωνο και δε μπορεί να προβληθεί το εσωτερικό κομμάτι του. Έτσι χρειάζεται ειδική επεξεργασία ώστε να αναχθεί σε τρίγωνα τα οποία βρίσκονται εξ ολοκλήρου μέσα στο πλαίσιο.

Στον αλγόριθμο που περιγράφεται παρακάτω (Algorithm 1 στις επόμενες σελίδες) έχει γίνει η παραδοχή ότι θα καλυφθούν μόνο οι περιπτώσεις όπου το τρίγωνο τέμνει είτε μία ακμή του πλαισίου, είτε δύο γειτονικές έτσι ώστε η κορυφή στο σημείο τομής τους να βρίσκεται μέσα στο τρίγωνο. Οι περιπτώσεις αυτές φαίνονται καλύτερα στο σχήμα 1. Αυτή η παραδοχή έγινε για απλούστευση του αλγορίθμου, εφόσον οι υπόλοιπες περιπτώσεις κρίθηκαν περιττές διότι για να βρεθεί σε αυτές ένα τρίγωνο θα έπρεπε να βρίσκεται πολύ κοντά στην κάμερα, τα οποία τρίγωνα όμως αποκλείονται από το render. Ο κώδικας της συνάρτησης αυτής βρίσκεται στο αρχείο Additional_Functions.py μαζί με άλλες αναγκαίες συναρτήσεις.

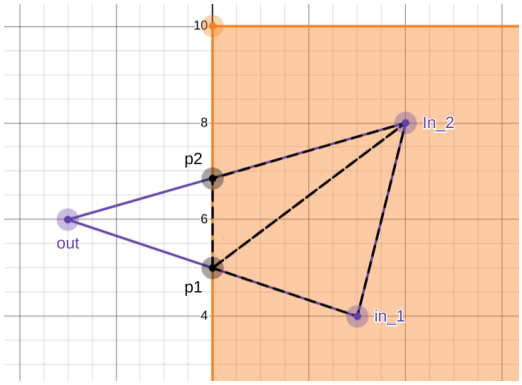
4 Συναρτήσεις

Η λειτουργία των συναρτήσεων περιγράφτηκε στην ενότητα 3, εδώ περιγράφεται η υλοποίηση τους.

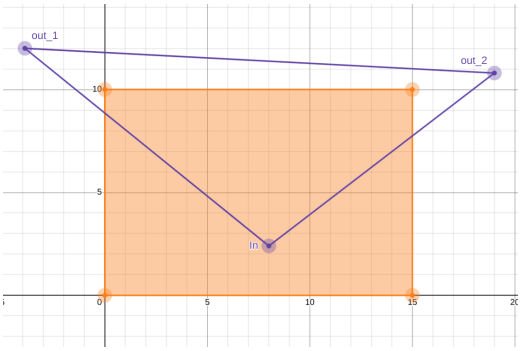
4.1 Η συνάρτηση world2view(pts, R, c0) → points_transf

Μετατρέπει τα σημεία εισόδου σε ομογενής συντεταγμένες, εφαρμόζει τον παρακάτω τύπο (1) και επιστρέφει τα μετασχηματισμένα σημεία σε καρτεσιανές συντεταγμένες (x_c, y_c, z_c).

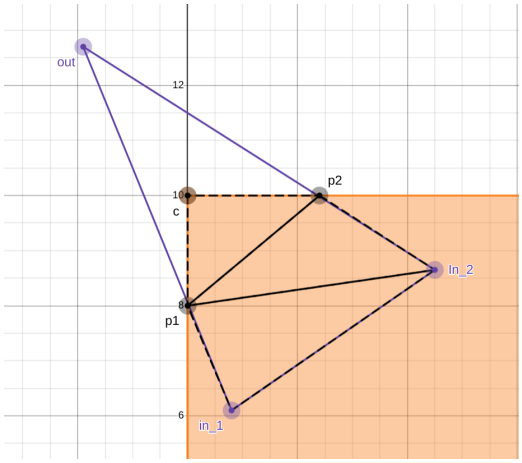
$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \left[\begin{array}{c|c} R^T & -R^T c0 \\ \hline 0_{1\times 3} & 1 \end{array}\right] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$



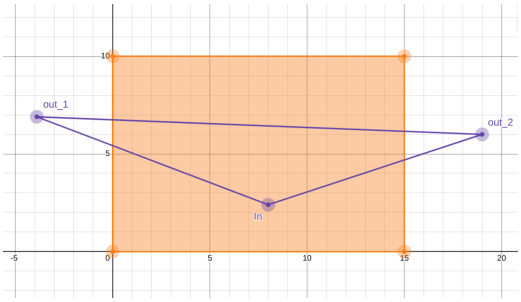
(α΄) Δύο σημεία μέσα, ίδια πλευρά



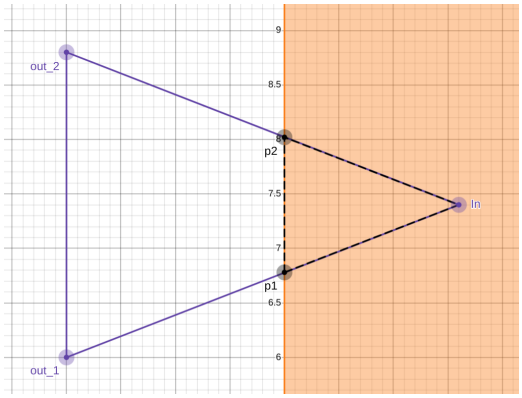
(β΄) Παράδειγμα που δεν συμπεριλαμβάνεται



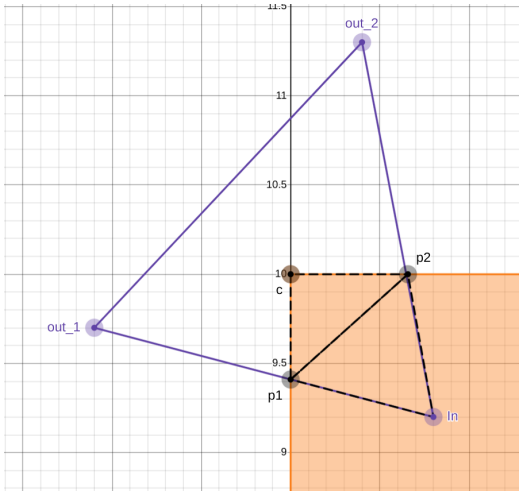
(γ΄) Δύο σημεία μέσα, στην κορυφή



(δ΄) Παράδειγμα που δεν συμπεριλαμβάνεται



(ε΄) Ένα σημείο μέσα, στην ίδια πλευρά



(ϛ΄) Ένα σημείο μέσα, στην κορυφή

Σχήμα 1: Παραδείγματα περιπτώσεων αποκοπής. Η αριστερή στύλη δείχνει περιπτώσεις που λαμβάνονται υπόψη και η δεξιά περιπτώσεις που δεν λαμβάνονται, διότι τα τρίγωνα είναι πολύ κοντά στην κάμερα(πιάνουν σχεδόν όλο το πλαίσιο).

Algorithm 1 Ψευδοκώδικας αποκοπής (clipping)

```
Βρες ποιές πλευρές είναι μέσα στο πλαίσιο προβολής και ποιές έξω από αυτό
if Όλες μέσα then
    g_shading(τρίγωνο)
else if δύο μέσα then
    Βρες τα σημεία τομής με το πλαίσιο, p1,p2
    Βρες τα χρώματα των p1,p2 με vector_interp
    if p1,p2 στην ίδια ακμή then                          ▷ περίπτωση (α) στο Σχήμα 1
        g_shading(p1,p2,in_1)
        g_shading(in_1,in_2,p2)
    else                                                  ▷ περίπτωση (γ) στο Σχήμα 1
        Βρες την κορυφή του πλαισίου corner που βρίσκεται μέσα στο τρίγωνο
        Βρες τα χρώματα της corner με vector_interp
        g_shading(corner,p1,p2)
        g_shading(p1,p2,in_1)
        g_shading(p2,in_1,in_2)
    end if
else if ένα σημείο μέσα then
    Βρες σημεία τομής με το πλαίσιο και χρώματα τους,p1,p2
    if p1,p2 στην ίδια ακμή του πλαισίου then             ▷ περίπτωση (ε) στο Σχήμα 1
        g_shading(in,p1,p2)
    else                                                  ▷ περίπτωση (στ) στο Σχήμα 1
        Βρες την κορυφή corner και το χρώμα της
        g_shading(corner,p1,p2)
        g_shading(p1,p2,in)
    end if
else
    continue (βρίσκεται όλο το τρίγωνο έξω)
end if
```

4.2 Η συνάρτηση lookat(eye, up, target) → (R, d)

Αν ονομάσουμε τις εισόδους ως εξίς: $eye \to C, up \to \hat{\mathbf{u}}, target \to K$, τότε εφαρμόζει τις παρακάτω εξισώσεις για να βρει τους πίνακες $\mathbf{R}, \mathbf{d}$.

$$\hat{\mathbf{z_c}} = \frac{\vec{CK}}{|CK|}, \quad \mathbf{t} \equiv \mathbf{u} - \langle \mathbf{u}, \hat{\mathbf{z_c}} \rangle \hat{\mathbf{z_c}}, \quad \hat{\mathbf{y_c}} = \frac{\mathbf{t}}{|\mathbf{t}|}, \quad \hat{\mathbf{x_c}} = \hat{\mathbf{y_c}} \times \hat{\mathbf{z_c}}$$

$$\mathbf{R} = [\hat{\mathbf{x_c}}, \hat{\mathbf{y_c}}, \hat{\mathbf{z_c}}], \quad \mathbf{d} = C = eye$$

Συγκεκριμένα το εξωτερικό γινόμενο για τον υπολογισμό του $\hat{\mathbf{x_c}}$ γίνεται αντίστροφα ώστε τα θετικά x να δείχνουν προς τα δεξιά και όχι ανάποδα, δηλαδή: $\hat{\mathbf{x_c}} = \hat{\mathbf{z_c}} \times \hat{\mathbf{y_c}}$. Στο σχήμα 2 φαίνονται και οι δύο περιπτώσεις συστημάτων συντεταγμένων μαζί με την κάμερα και είναι εμφανές ότι στο δεύτερο τα y δείχνουν προς τα πάνω και τα x προσ τα δεξιά.

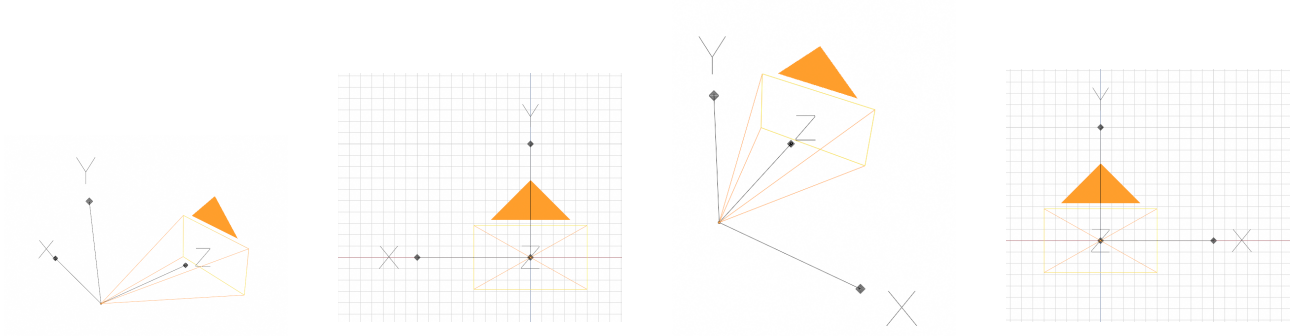
Επιστρέφεται το tuple $(\mathbf{R}, \mathbf{d})$.

4.3 Η συνάρτηση perspective_project(pts, focal, R, t) → (pts_2d, depth)

Πρώτα μετατρέπει τα σημεία pts στις συντεταγμένες της κάμερας μέσω της συνάρτησης $world2view$ με παραμέτρους R, t. Έπειτα, αν (x_p, y_p, z_p) είναι τα σημεία στις συντεταγμένες της κάμερας και (x_q, y_q) τα σημεία μετά την προβολή ($zq = 0$, διότι τα σημεία είναι σε δύο διαστάσεις), ο μετασχηματισμός γίνεται με τις παρακάτω εξισώσεις:

$$x_q = \frac{x_p \cdot focal}{z_p}, \quad y_q = \frac{y_p \cdot focal}{z_p}$$

Οι εξισώσεις αυτές εφαρμόζονται σε κάθε σημείο στον πίνακα pts.



(α′) Αρχικό

(β′) Αρχικό από την οπτική της κάμερας

(γ′) Μετά την αντιστροφή

(δ′) Μετά την αντιστροφή, από την οπική της κάμερας

Σχήμα 2: Κάμερα και βάσεις συστάματος συντεταγμένων της

Το βάθος κάθε σημείου είναι η συντεταγμένη z_p, δηλαδή η απόστασή του από την κάμερα. Έτσι, δημιουργείται ένας πίνακας $depths : 1 \times N$ που περιέχει για κάθε σημείο στον πίνακα $pts : 3 \times N$ το βάθος του.

Επιστρέφεται το tuple των μετασχηματισμένων σημείων και του βάθους τους.

4.4 Η συνάρτηση rasterize(pts_2d, plane_w, plane_h, res_w, res_h) → pixels

Τα μήκη κατά x και κατά y των $pixel$ δίνονται από την διαίρεση:

$$pixel\_len\_h = \frac{plane\_h}{res\_h}, \qquad pixel\_len\_w = \frac{plane\_w}{res\_w}$$

Αν οι διαστάσεις έχουν επιλεχθεί έτσι ώστε τα $pixel$ να είναι τετράγωνα, πρέπει αυτές οι δύο τιμές να είναι ίσες.

Αν το σημείο (x_{point}, y_{point}) αντιστοιχίζεται στο (x_{pixel}, y_{pixel}) τότε θα γίνει με την παρακάτω εξίσωση:

$$x_{pixel} = \frac{x_{point} + \frac{plane\_w}{2}}{pixel\_len\_w}, \qquad y_{pixel} = \frac{y_{point} + \frac{plane\_h}{2}}{pixel\_len\_h}$$

Η ποσότητες $\frac{plane\_w}{2}$ και $\frac{plane\_h}{2}$ προστίθονται ώστε το $pixel$ $(0, 0)$ να βρίσκεται κάτω αριστερά και όχι στο κέντρο.

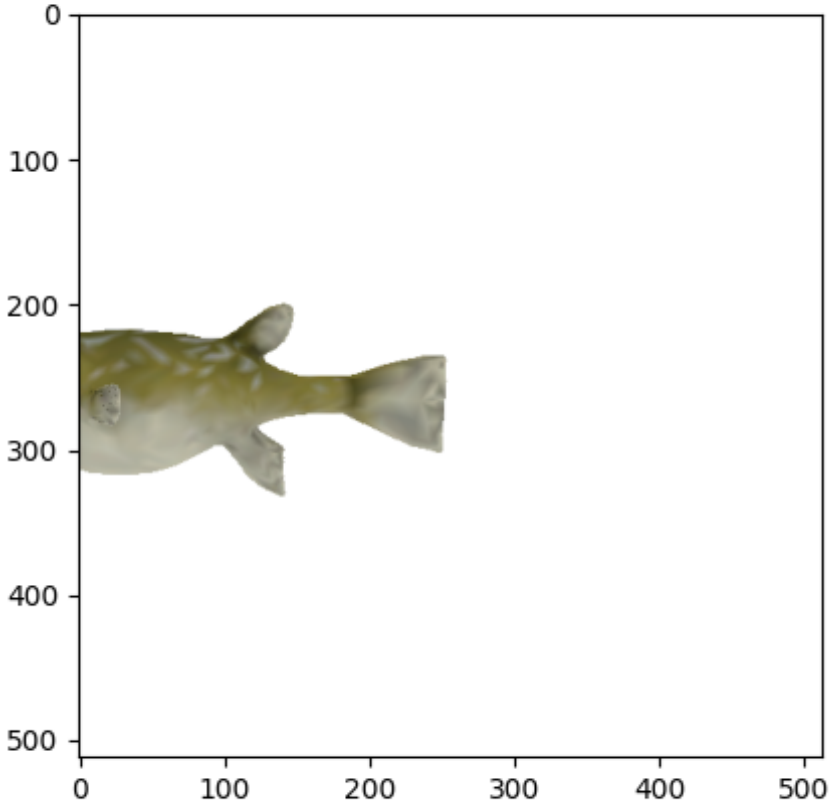
Οι παραπάνω εξισώσεις εφαρμόζονται για κάθε σημείο στον πίνακα pts.

4.5 Η συνάρτηση render_object(v_pos, v_clr, t_pos_idx, plane_h, plane_w, res_h, res_w, focal, eye, up, target) → image

Είναι η συνάρτηση που θα κληθεί με ορίσματα τα σημεία και τα χρώματα των αντικειμένων, τις διαστάσεις του πλαισίου, τα χαρακτηριστικά της κάμερας και εμφανίζει την εικόνα. Θα κληθούν οι παραπάνω συναρτήσεις με τη σειρά.

Υπολογίζεται ο πίνακας περιστροφής για τη μετατροπή του συστήματος συντεταγμένων με τη συνάρτηση $lookat$. Προβάλλονται τα σημεία στο πέτασμα της κάμερας με την $perspective\_project$ (η συνάρτηση $world2view$ καλείται για τη μετατροπή συντεταγμένων μέσα στη $perspective\_project$). Υπολογίζονται τα βάθη των τριγώνων ως το κέντρο βάρους των κορυφών τους, ταξινομούνται κατά φθήνον αριθμό βάθους, δηλαδή από το πιο μακρινό από την κάμερα στο πιο κοντινό. Για κάθε τρίγωνο ελέγχεται αν έχει αρνητικό ή μηδενικό βάθος, αν έχει (δηλαδή το τρίγωνο βρίσκεται πίσω από την κάμερα) συνεχίζει με το επόμενο, αλλιώς το σκιάζει. Η σκίαση γίνεται με τη συνάρτηση $triangle\_clipping(image, triangle\_vertices, triangle\_vcolors, res\_h, res\_w)$ η οποία υλοποιεί την λειτουργία αποκοπής που περιγράφθηκε στην ενότητα 3.1.

Αφού τελειώσει αυτή η διαδικασία επιστρέφεται η εικόνα.



Σχήμα 3: Αποτέλεσμα δοκιμής αποκοπής

5 Το αρχείο demo.py

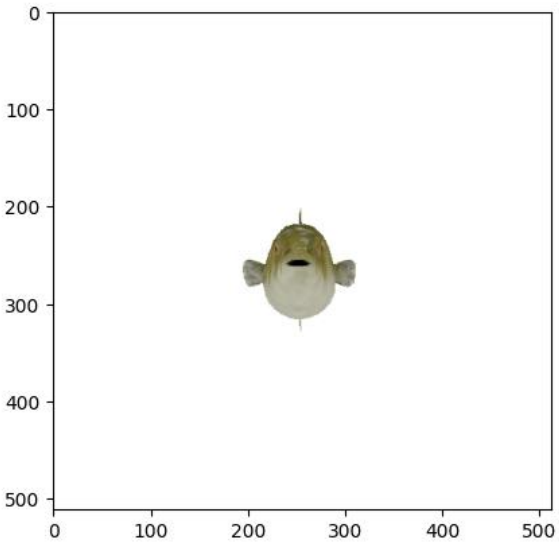
Στο αρχείο αυτό επιδεικνύεται η λειτουργία των παραπάνω αλγορίθμων. Αρχικά φοτρώνονται τα δεδομένα από το αρχείο hw2.npy και έπειτα εφαρμόζονται οι εξίς μετασχηματισμοί στο δωσμένο αντικέιμενο:

- Περιστροφή δεξιόστροφα κατά 30 μοίρες.
- Μετατόπιση κατά 1 μονάδα του WCS προς την κατεύθυνση x
- Μετατόπιση κατά 1 μονάδα του WCS προς την κατεύθυνση $-z$

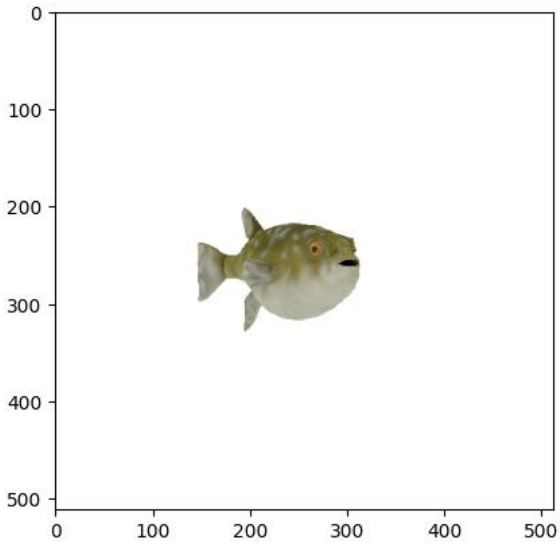
Τα αρχικά και μετασχηματισμένα σχήματα προβάλλονται σε εικόνες με τη συνάρτηση *render_object* και οι εικόνες προβάλλονται με τη χρήση της βιβλιοθήκης matplotlib.pyplot και αποθηκεύονται σε τέσσερεις εικόνες με ονόματα 0.jpg, 1.jpg, 2.jpg, 3.jpg.

Οι εικόνες αυτές φαίνονται στο σχήμα 4.

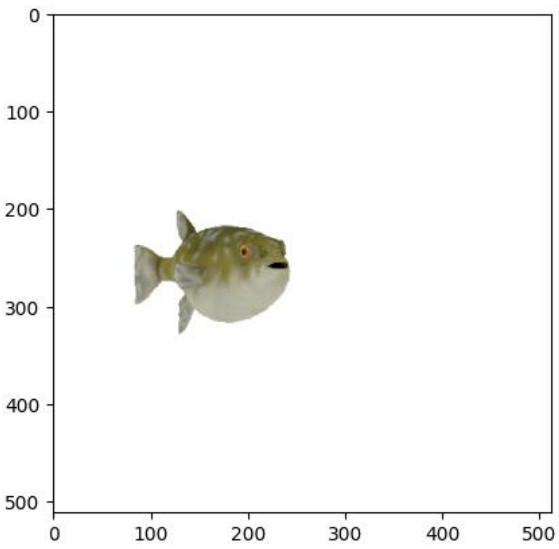
Επιπλέον, για τον έλεγχο της λειτουργίας της αποκοπής το αρχικό αντικείμενο περιστράφηκε κατά 90 μοίρες και μετατοπίστηκε κατά τον άξονα x κατά 3.5 μονάδες του WCS ώστε να βρίσκεται μισό μέσα στο πλαίσιο και μισό έξω από αυτό. Στο σχήμα 3 φαίνεται το αποτέλεσμα όπου φαίνεται ότι η αποκοπή λειτούργησε ως προοριζόταν.



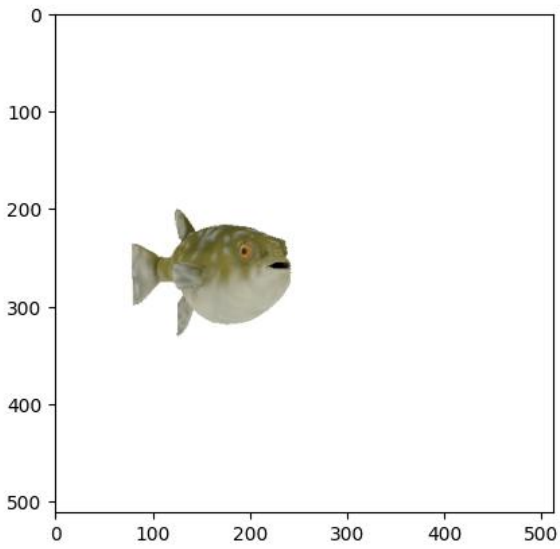
(α′) Αρχική θέση



(β′) Μετά την περιστροφή



(γ′) Μετά τη μετατόπιση κατά x



(δ′) Μετά τη μετατόπιση κατά z

Σχήμα 4: Αποτελέσματα λειτουργίας αρχείου demo.py