Εργασία 2: Μετασχηματισμοί και Προβολές

Παλάσκα Αιμιλία 10453 (aimiliapm@ece.auth.gr)
Επιβλέποντες: Αναστάσιος Ντελόπουλος, Αντώνης Καρακώττας

Λειτουργία και Τρόπος Κλήσης των Προγραμμάτων

Τα προγράμματα μπορούν να κληθούν είτε από κάποιο IDE π.χ. VSCode είτε με την εντολή τερματικού */bin/python3 demo.py* **από τον φάκελο που περιέχει όλα τα αρχεία κώδικα και το αρχείο δεδομένων** (π.χ. μετάβαση με την εντολή τερματικού cd {path}). Να σημειωθεί πως απαιτούνται μερικά λεπτά για να ολοκληρωθεί η αποθήκευση των τελικών εικόνων. Ο κώδικας μπορεί να βρεθεί και στο GitHub.

Ψευδοκώδικας Κλάσεων, Συναρτήσεων & Script

- Κλάση Μετασχηματισμών

```
class Transform:
    def __init__(self):
        Initialize a Transform object with a 4x4 identity matrix

    def rotate(self, theta: float, u: np.ndarray) -> None:
        Extract cosine, sine and coordinates of u
        Construct the rotation matrix
        Rotate the transformation matrix by multiplying it with the
rotation matrix

    def translate(self, t: np.ndarray) -> None:
        Update the translation part of the matrix (last column) by
adding the vector t to it

    def transform_pts(self, pts: np.ndarray) -> np.ndarray:
        Turn to homogeneous coordinates by adding a row of ones to the
points
        Transform the specified points by multiplying them with our
current matrix
        Return the first three rows of the multiplication result
```

- Συνάρτηση αλλαγής συστήματος συντεταγμένων

```
def world2view(pts: np.ndarray, R: np.ndarray, c0: np.ndarray) ->
np.ndarray:
    Subtract the camera coordinates vector c0 from the given points pts
    Multiply the subtracted points with the transposed rotation matrix R
    Return the transformed coordinates of the points
```

- Συνάρτηση προσανατολισμού κάμερας

```
def lookat(eye: np.ndarray, up: np.ndarray, target: np.ndarray) ->
tuple[np.ndarray, np.ndarray]:
    Calculate zc as target - eye and normalize it
    Calculate yc as u - (u * zc) zc and normalize it
    Calculate xc as yc x zc and normalize it
    Form them as columns to a rotation matrix R
    Return R and the translation vector, which is eye
```

- Συνάρτηση προοπτικής προβολής με pinhole κάμερα

```
def perspective_project(pts: np.ndarray, focal: float, R: np.ndarray, t:
np.ndarray) -> tuple[np.ndarray, np.ndarray]:
    Transform points to camera coordinates using the implemented
function world2view
    Handle exception of points being too close (z = 0) to the plane
    Extract depth as last row of pts (z coordinate)
    Project them to the image plane by dividing with z and multiplying
by focal
    Return the projected points and the depth vector
```

- Συνάρτηση απεικόνισης

```
def rasterize(pts_2d: np.ndarray, plane_w: int, plane_h: int, res_w:
int, res_h: int) -> np.ndarray:
    Remove any points outside of the camera plane
    Calculate rasterization ratio as res_w / plane_w and res_h / plane_h
    Calculate rasterization offset as plane_w / 2 and plane_h / 2
    Apply rasterization by adding offset and then multiplying by the
ratio
    Return points with pixel coordinates after rounding them to the
closer integer
```

- Συνάρτηση φωτογράφισης

```
def render_object(v_pos, v_clr, t_pos_idx, plane_h, plane_w, res_h,
res_w, focal, eye, up, target) -> np.ndarray:
    Get the rotation matrix and translation vector using the lookat
function
    Project the points onto the camera plane using the
perspective_project function
    Convert the plane to image pixels using the rasterize function
    Render the image with Gouraud shading using the render_img function
    Return the rendered image
```

- Script επίδειξης *demo.py*

```
Load 'hw2.npy' file
Extract data to vectors (v_pos, v_clr, t_pos_idx, eye, up, target,
focal, plane_w,
plane_h, res_w, res_h, theta_0, rot_axis_0, t_0, t_1)
Initialize three Transform instances
Capture the initial image
Rotate points and capture the first image
Translate points and capture the second image
Translate points and capture the third image
Save results locally with .jpg extension
```

- Διευκρινίζεται ότι τα αρχεία *vector_interp.py*, *f_shading.py*, *g_shading.py* και *render_img.py* ενσωματώθηκαν από την υλοποίηση της πρώτης εργασίας στην πλήρωση τριγώνων και **είναι απαραίτητο να βρίσκονται στον ίδιο φάκελο με τα υπόλοιπα αρχεία προκειμένου να παραχθεί η τελική εικόνα**.

Παραδοχές που χρησιμοποιήθηκαν

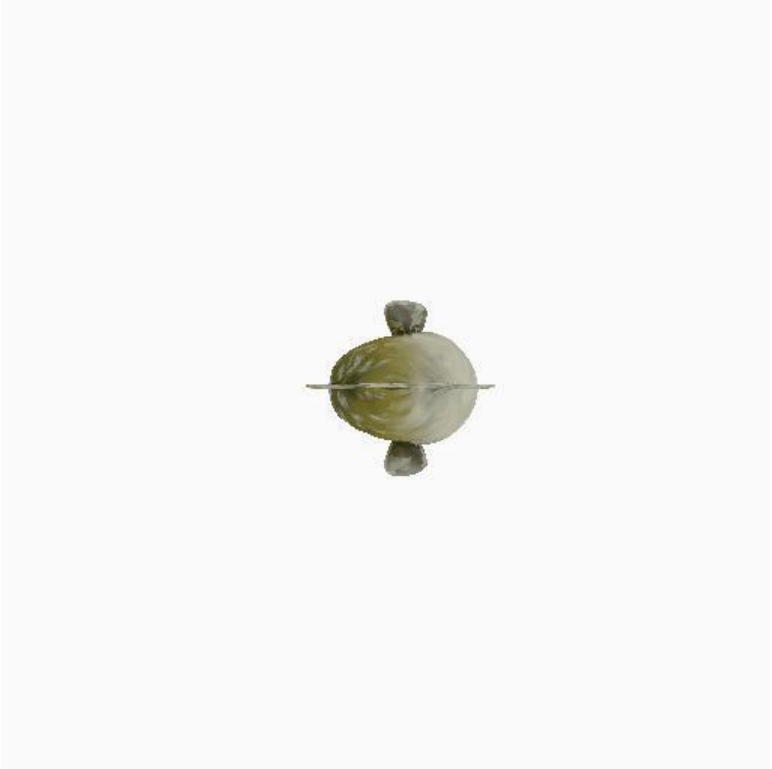
- Για την λειτουργία των scripts είναι απαραίτητο όλα τα αρχεία να βρίσκονται στον ίδιο φάκελο, όπως επίσης και το δοσμένο αρχείο δεδομένων *hw2.py*.
- Γίνονται όλες οι παραδοχές που αναφέρθηκαν στην πρώτη εργασία σχετικά με τις συναρτήσεις *g_shading*, *vector_interp* και *render_img*
- Στα σχόλια στην αρχή κάθε συνάρτησης, καθορίζονται οι διαστάσεις των ορισμάτων και των επιστρεφόμενων στοιχείων (μεταβλητές, διανύσματα, πίνακες, tuples) όπως επίσης και τι αντιπροσωπεύουν.

Πηγές

- Γλωσσικό μοντέλο ChatGPT 3.5
- Επέκταση κειμενογράφου Code Blocks
- Visual Studio Code - Community Edition
- Python- Version 3.10.12

Ενδεικτικά αποτελέσματα από το demo

Step 0: Αποτέλεσμα απεικόνισης από τα σημεία χωρίς εφαρμογή μετασχηματισμού



Step 1: Αποτέλεσμα απεικόνισης μετά από δεξιόστροφη περιστροφή των σημείων κατά γωνία $\theta = 0.5236$ *rads* ως προς τον άξονα των y

Step 2: Αποτέλεσμα απεικόνισης μετά από μετατόπιση των σημείων κατά διάνυσμα *t_0 = [1, 0, 0]* δηλαδή κάθετη μετατόπιση, δεδομένου ότι *eye = [0, 0, -35]* και *target = [0, 0, 0]* δηλαδή η κάμερα βρίσκεται πάνω στον άξονα z και ”στοχεύει” προς τα πάνω

Step 3: Αποτέλεσμα απεικόνισης μετά από μετατόπιση των σημείων κατά διάνυσμα *t_1 = [0, 0, -1]* δηλαδή μεγέθυνση (μετατόπιση του βάθους), δεδομένου ότι *eye = [0, 0, -35]* και *target = [0, 0, 0]* δηλαδή η κάμερα βρίσκεται πάνω στον άξονα z και ”στοχεύει” προς τα πάνω