# Γραφική με Υπολογιστές 2025

# Εργασία #2: Μετασχηματισμοί και Προβολές

## Ζητούμενα

### Α. Συναρτήσεις Μετασχηματισμών

Να υλοποιηθεί η συνάρτηση:

```
def translate(t_vec: np.ndarray) -> np.ndarray:
    """
    Create an affine transformation matrix w.r.t. the
    specified translation vector.
    """
    return xform
```

η οποία δημιουργεί έναν πίνακα affine μετασχηματισμού λόγω μετατόπισης, όπου:

- `t_vec` ένα διάνυσμα μετατόπισης $1 \times 3$.
- `xform` ο affine μετασχηματισμός, πίνακας $4 \times 4$.

Να υλοποιηθεί η συνάρτηση:

```
def rotate(axis: np.ndarray, angle: float, center: np.ndarray) -> np.ndarray:
    """
    Create an affine transformation matrix w.r.t. the
    specified rotation.
    """
    return xform
```

η οποία δημιουργεί έναν πίνακα affine μετασχηματισμού λόγω περιστροφής, που αντιστοιχεί σε δεξιόστροφη περιστροφή κατά γωνία `angle` σε rads, περί άξονα με κατεύθυνση που δίνεται από το διάνυσμα `axis`. Όπου:

- `axis` Ο άξονας περιστροφής, διάνυσμα $1 \times 3$.
- `angle` Η γωνία περιστροφής.
- `center` Το σημείο από το οποίο διέρχεται ο άξονας περιστροφής.
- `xform` Ο affine μετασχηματισμός, πίνακας $4 \times 4$.

Να υλοποιηθεί η συνάρτηση:

```
    def compose(mat1: np.ndarray, mat2: np.ndarray) -> np.ndarray:
        return mat
```

η οποία πραγματοποιεί τη σύνθεση δύο πινάκανων affine μετασχηματισμών.

## Β. Συνάρτηση αλλαγής συστήματος συντεταγμένων

Να υλοποιηθεί η συνάρτηση:

```
    def world2view(pts: np.ndarray, R: np.ndarray, c0: np.ndarray) -> np.ndarray:
        # Implements a world-to-view transform, i.e. transforms the specified
        # points to the coordinate frame of a camera. The camera coordinate frame
        # is specified rotation (w.r.t. the world frame) and its point of reference
        # (w.r.t. to the world frame).
```

η οποία μετασχηματίζει τα σημεία εισόδου `pts` στο σύστημα συντ/νων της κάμερας. Πιο συγκεκριμένα:

- `pts` $\in \mathbb{R}^{3\times N}$ είναι ο $3 \times N$ πίνακας με τις συντ/νες των αρχικών σημείων ως προς ένα σύστημα συντ/νων.
- `R` $\in \mathbb{R}^{3\times 3}$ είναι ο πίνακας περιστοφής το νέου συστήματος ως προς το αρχικό.
- `c0` $\in \mathbb{R}^3$ το σημείο αναφοράς του νέου συστήματος συντ/νων ως προς το αρχικό.

Η συνάρτηση θα επιστρέφει έναν πίνακα $N \times 3$ με τα σημεία `pts` μετασχηματισμένα ως προς το σύστημα συντ/νων της κάμερας.

## Γ. Συνάρτηση προσανατολισμού κάμερας

Να υλοποιηθεί η συνάρτηση:

```
    def lookat(eye: np.ndarray, up: np.ndarray, target: np.ndarray) -> Tuple[np.ndarray,
    np.ndarray]:
        # Calculate the camera's view matrix (i.e., its coordinate frame transformation
    specified
        # by a rotation matrix R, and a translation vector t).
        # :return a tuple containing the rotation matrix R (3 x 3) and a translation
    vector
        #   t (1 x 3)
```

Η συνάρτηση θα επιστρέφει τις παραμέτρους που χρειάζονται για το μετασχηματισμό σημείων από το WCS στο σύστημα συντ/νων της κάμερας. Αυτές θα είναι ο πίνακας περιστροφής $R \in \mathbb{R}^{3\times 3}$, και το διάνυσμα μετατόπισης $t \in \mathbb{R}^3$. Πιο συγκεκριμένα:

- `eye` $\in \mathbb{R}^3$ είναι το σημείο του κέντρου της κάμερας.
- `up` $\in \mathbb{R}^3$ είναι το μοναδιαίο up διάνυσμα της κάμερας.
- `target` $\in \mathbb{R}^3$ είναι το σημείο στόχος.

## Δ. Συνάρτηση προοπτικής προβολής με pinhole κάμερα

Να υλοποιηθεί η συνάρτηση:

```
25  def perspective_project(pts: np.ndarray, focal: float, R: np.ndarray, t: np.ndarray)
    -> Tuple[np.ndarray, np.ndarray]:
26      # Project the specified 3d points pts on the image plane, according to a pinhole
    perspective projection model.
```

η οποία παράγει τις προοπτικές προβολές και το βάθος των σημείων εισόδου `pts`. Πιο συγκεκριμένα:

- `pts` $\in \mathbb{R}^{3\times N}$ είναι ο πίνακας των σημείων εισόδου.
- `focal` είναι η απόσταση του πετάσματος της κάμερας από το κέντρο (μετρημένη στις μονάδες που χρησιμοποιεί το σύστημα συντ/νων της κάμερας).
- `R` $\in \mathbb{R}^{3\times 3}$ είναι ο πίνακας περιστροφής προς το σύστημα συντ/νων της κάμερας.
- `t` $\in \mathbb{R}^3$ είναι το διάνυσμα μετατόπισης προς το σύστημα συντ/νων της κάμερας.

Η συνάρτηση θα επιστρέφει τις 2D συντ/νες των σημείων εισόδου στο πέτασμα της κάμερας.

## Ε. Συνάρτηση απεικόνισης

Να υλοποιήσετε τη συνάρτηση:

```
27  def rasterize(pts_2d: np.ndarray, plane_w: int, plane_h: int, res_w: int, res_h: int)
     -> np.ndarray:
28      # Rasterize the incoming 2d points from the camera plane to image pixel
    coordinates
```

η οποία απεικονίζει τις συντ/νες των σημείων από το σύστημα συντ/νων του πετάσματος της κάμερας, με πέτασμα `plane_h` $\times$ `plane_w`, σε ακέραιες θέσεις (pixels) της εικόνας διάστασης `res_h` $\times$ `res_w`.

**Σημείωση**: Ο άξονας της κάμερας περνά από το κέντρο ορθογώνιου διάστασης `plane_h` $\times$ `plane_w`, ενώ η αρίθμιση του `res_h` $\times$ `res_w` πίνακα της εικόνας ξεκινά από τα κάτω προς τα πάνω και από αριστερά προς δεξιά και έχει τιμές $[0, \ldots,$ `res_w`$]$ οριζοντίως και $[0, \ldots,$ `res_h`$]$ κατακορύφως.

## ΣΤ. Συνάρτηση φωτογράφισης

Να υλοποιήσετε τη συνάρτηση:

```
29  def render_object(v_pos, v_clr, t_pos_idx, plane_h, plane_w, res_h, res_w, focal, eye
    , up, target) -> np.ndarray:
30      # render the specified object from the specified camera.
```

η οποία φωτογραφίζει μία 3D σκηνή ενός αντικειμένου από μία κάμερα. Πιο συγκεκριμένα:

- `v_pos` $\in \mathbb{R}^{N\times 3}$ είναι οι τρισδιάστατες συντ/νες των σημείων του αντικειμένου.
- `v_clr` $\in \mathbb{R}^{N\times 3}$ είναι ο πίνακας με τα χρώματα των κορυφών.
- `t_pos_idx` είναι ο πίνακας που περιέχει δείκτες σε σημεία του πίνακα `v_pos` που αποτελούν τις κορυφές των τριγώνων. Ο πίνακας είναι διάστασης $F \times 3$. Η $i$-οστή γραμμή του πίνακα, δηλώνει τις τρεις κορυφές που σχηματίζουν το τρίγωνο (με αναφορά σε κορυφές του πίνακα `v_pos` και η αρίθμησή του ξεκινάει από το 0).

- `plane_h` και `plane_w` είναι το ύψος και το πλάτος του πετάσματος της κάμερας.
- `res_h` και `res_w` είναι το ύψος και το πλάτος του καμβά σε pixel.
- `focal` είναι η απόσταση του πετάσματος από το κέντρο της κάμερας.
- `eye` είναι το κέντρο της κάμερας ως προς το WCS.
- `up` είναι το up διάνυσμα της κάμερας.
- `target` είναι το σημείο στόχος της κάμερας.

Ολά τα σημεία, και οι πίνακες που αφορούν 3D συντ/νες δίνονται σε μη ομογενή μορφή. Η συνάρτηση θα επιστρέφει έναν πίνακα $res_h \times res_w \times 3$ (τη φωτογραφία του αντικειμένου) και χρησιμοποιώντας κατάλληλα τις παραπάνω συναρτήσεις θα υλοποιεί όλο το pipeline της απεικόνισης ενός αντικειμένου. Επίσης θα πρέπει να χρησιμοποιεί τη συνάρτηση πλήρωσης της προηγούμενης εργασίας για να χρωματίσει το αντικείμενο με τη μέθοδο Gouraud.

# Παραδοτέα

Έστω ότι ένα αυτοκίνητο κινείται σε έναν κυκλικό δρόμο με κέντρο `k_circle_center`, ακτίνας `k_circle_radius`, με ταχύτητα `k_car_velocity` $\frac{m}{sec}$. Έστω ακόμα κάμερα στη θέση του συνοδηγού, με σχετική θέση με το αυτοκίνητο `k_cam_rel_pos`. Τα demos που θα παραδώσετε θα πρέπει να προσομοιώνουν τα παρακάτω προβλήματα:

- Καθώς το αυτοκίνητο κινείται, η κάμερα είναι στατική και κοιτάει πάντα μπροστά, δλδ. το διάνυσμα $\hat{z}_c$ είναι πάντα παράλληλο με το διάνυσμα ταχύτητας του αυτοκινήτου.
- Καθώς το αυτοκίνητο κινείται, η κάμερα μπορεί να περιστρέφεται, και κοιτάει πάντα το σημείο `k_cam_target`, που σας δίνεται στο αρχείο της εργασίας.

Θα πρέπει να παραδώσετε:

1. Οι παραπάνω συναρτήσεις σε μορφή **σχολιασμένου** πηγαίου κώδικα python (μέχρι $v3.10$) με σχόλια γραμμένα στα **αγγλικά** ή **greeklish**, (κοινώς, **μη γράφετε σχόλια με ελληνικούς χαρακτήρες)**.
2. Δύο demos που θα υλοποιούν τις δύο παραπάνω προσομοιώσεις. Οι προσομοιώσεις θα πρέπει να έχουν διάρκεια 5 δευτερόλεπτα, και να αποθηκεύονται 25 εικόνες ανά δευτερόλεπτο ($25fps$). Το αντικείμενο που σας δίνεται είναι μία πυραμίδα με κορυφές `v_pos`, τρίγωνα `t_pos_idx`, συντ/νες uv `v_uvs`, και εικόνα υφής `diffuse.jpg`. Θα χρησιμοποιήσετε τη συνάρτηση πλήρωσης με υφή από την προηγούμενη εργασία.

   **Σημείωση 1**: Αν δεν είχατε υλοποιήσει τη συνάρτηση `render`, ή αν είχατε κάποιο λάθος στην υλοποίησή της, μπορείτε να χρησιμοποιήσετε τη `render` κάποιου/κάποιας συναδέλφου σας, **αρκεί να το δηλώσετε το όνομά του στην αναφορά σας**.

   **Σημείωση 2**: Το αρχείο hw2.npy που σας δίνεται, περιέχει τις παραμέτρους του αντικειμένου `v_pos, v_uvs, t_pos_idx, diffuse.jpg` καθώς και όλες τις παραμέτρους που θεωρούνται γνωστές, όπως το κέντρο και η ακτίνα του κύκλου κατά τον οποίο κινείται το αυτοκίνητο, η ταχύτητα του αυτοκινήτου, και οι παράμετροι της κάμερας.

3. Αναφορά με:
    - Περιγραφή της λειτουργίας και του τρόπου κλήσης των προγραμμάτων.
    - Περιγραφή των συναρτήσεων.
    - Τα ενδεικτικά αποτελέσματα που παράγονται από το demo.

# Παρατηρήσεις

- Θεωρείστε ότι οι διαστάσεις της εικόνας είναι: `res_h = res_w = 512`.
- Οι εργασίες αξιολογούνται με χρήση Python(ως v3.10).
- Οι εργασίες είναι **αυστηρά** ατομικές.
- Το background του καμβά είναι λευκό `rgb = np.array([1.0, 1.0, 1.0])`.
- Υποβάλετε ένα και μόνο αρχείο, τύπου .zip.
- Το όνομα του αρχείου πρέπει να είναι AEM.zip, όπου AEM είναι τα τέσσερα ψηφία του Α.Ε.Μ. του φοιτητή της ομάδας.
- Το προς υποβολή αρχείο πρέπει να περιέχει τα αρχεία κώδικα python και το αρχείο report.pdf το οποίο θα είναι η αναφορά της εργασίας.
- Η αναφορά πρέπει να είναι ένα αρχείο τύπου PDF, και να έχει όνομα report.pdf.
- Όλα τα αρχεία κώδικα πρέπει να είναι αρχεία κειμένου τύπου UTF-8, και να έχουν κατάληξη .py .
- Το αρχείο τύπου zip που θα υποβάλετε δεν πρέπει να περιέχει κανένα φάκελο.
- Για την ονομασία των αρχείων που περιέχονται στο προς υποβολή αρχείο, χρησιμοποιείτε μόνο αγγλικούς χαρακτήρες, και όχι ελληνικούς ή άλλα σύμβολα, πχ “#”, “$”, “%” κλπ.

**Προσοχή: Θα αξιολογηθούν μόνο όσες εργασίες έχουν demos που τρέχουν!**