Γραφικά Εργασία 1η

Κόρο Έρικα, 9707

Απρίλιος 8, 2022

Εισαγωγή

Σκοπός της εργασίας είναι η υλοποίηση ενός προγράμματος πλήρωσης τριγώνων που σχηματίζουν την προβολή ενός τρισδιάστατου αντικειμένου σε δύο διαστάσεις. Η σειρά με την οποία πρέπει να χρωματιστούν τα τρίγωνα προκύπτει από τον πίνακα βάθους που δίνεται στο numpy αρχείο. Ο χρωματισμός θα ξεκινάει από τα μακρινότερα τρίγωνα και θα συνεχίζει με τα κοντινότερα.

Σχετικά με τον κώδικα

Για την υλοποίηση του προγράμματος χρησιμοποιήθηκε ο package manager pip και οι βιβλιοθήκες numpy και matplotlib. Για να τρέξει επιτυχώς ο κώδικας χρειάζεται μέσα στο ίδιο directory με το project να βρίσκεται και το αρχείο hw1.npy που μας δίνεται. Το αρχείο αυτό περιέχει τα απαραίτητα δεδομένα για τις κορυφές των τριγώνων, το βάθος των κορυφών και τα χρώματα των κορυφών. Μέσα στα αρχεία demo_gouraud.py και demo_flat.py δίνεται ο κώδικας για την φόρτωση των δεδομένων. Το render της εικόνας παίρνει κάποια δευτερόλεπτα και όταν τελειώσει η εκτέλεση του προγράμματος η είκόνα αποθηκεύεται στο αντίστοιχο αρχείο.

Δομή του κώδικα

Αρχικά, υλοποιείται η συνάρτηση interpolate color για να υπολογίζει τις χρωματικές συνιστώσες ενός σημείου που βρίσκεται ανάμεσα σε δύο σημεία με γνωστές οριζόντιες ή κάθετες συντεταγμένες και γνωστές χρωματικές συνιστώσες. Ο τύπος που εφαρμόζεται είναι ο ακόλουθος:

$$m = \frac{x_2 - x}{x_2 - x_1} \tag{1}$$

$$value = m * C_1 + (1 - m) * C_2 \tag{2}$$

όπου C_1 είναι ο R, G, B πίνακας του πρώτου σημείου και C_2 είναι ο R, G, B πίνακας του δεύτερου σημείου.

Έπειτα, στο πλαίσιο πραγμάτωσης της συνάρτησης shade triangle , υλοποιήθηκε η συνάρτηση χάραξης ευθειών bresenham, όπως αυτή παρουσιάζεται στις σημειώσεις, με κάποιες τροποποιήσεις ώστε να συμπεριληφθεί και η περίπτωση όπου η κλίση της ευθείας είναι μεγαλύτερη της μονάδας. Η συνάρτηση αυτή καλείται τρεις φορές μέσα στην shade triangle με ορίσματα τις κορυφές του τριγώνου για να σχηματιστεί το αντίστοιχο τρίγωνο. Ανάλογα με τον επιθυμητό τρόπο σκίασης "flat" ή "gouraud" που δίνεται σαν είσοδος στην συνάρτηση, χρωματίζονται πρώτα τα pixels που βρίσκονται πάνω στις πλευρές του τριγώνου με βάση το χρώμα των κορυφών είτε καλώντας την interpolate color(gouraud) είτε χρωματίζοντας με το μέσο όρο χρώματος των κορυφών. Στη συνέχεια, ξεκινάει ένας βρόχος επανάληψης για την scanline η οποία ξεκινάει απ΄ το μικρότερο row των πλευρών του τριγώνου και καταλήγει στο μεγαλύτερο. Μέσα σ΄ αυτό τον βρόχο, θεωρούμε όλες τις πλευρές του τριγώνου ενεργές και σε κάθε επανάληψη βρίσκονται τα ενεργά σημεία της κάθε πλευράς. Τέλος, ανάμεσα στα ενεργά σημεία των πλευρών με το μικρότερο column χρωματίζονται καταλλήλως τα σημεία που βρίσκονται ανάμεσά τους.

Τελευταία υλοποιείται η συνάρτηση render . Μέσα σ΄ αυτή ταξινομείται ο πίνακας faces , ο οποίος ύστερα από τροποποίηση έχει πλέον τις συντεταγμένες των κορυφών κάθε τριγώνου,

με βάση το βάθος κάθε τριγώνου για να ξεκινήσει ο χρωματισμός των τριγώνων απ΄ τα μακρινότερα στα κοντινότερα. Αφού έχουν υλοποιηθεί όλα αυτά, καλείται η shade triangle για τον χρωματισμό όλων των τριγώνων και τον τελικό χρωματισμό του αντικειμένου.

```
def shade_triangle(img: list, verts2d: list, vcolors: list,
                        shade_t: str):

    # Call the bresenham algorithm 3 times to shape the triangle
    side1 = bresenham(verts2d[0], verts2d[1])
    side2 = bresenham(verts2d[1], verts2d[2])
    side3 = bresenham(verts2d[2], verts2d[0])

    # Shade edges
    if shade_t == "flat":
        # Color each point on the triangle's sides
        for pixel in side1:
            img[pixel[1]][pixel[0]] = list(average_color)
        for pixel in side2:
            img[pixel[1]][pixel[0]] = list(average_color)
        for pixel in side3:
            img[pixel[1]][pixel[0]] = list(average_color)
    elif shade_t == "gouraud":
        # Color the pixels in the edges
        for pixel in sides:
            # If the pixel is not a vertex, use interpolate_color
            if pixel not in verts2d:
                image[pixel] = interpolate_color()
            # else the color of the pixel is in the vcolors list
            else:
                image[pixel] = vcolors

    # Find the active points(the colored)
    for scan in range(scan_start, scan_end + 1):
        # Find the edge points that cross with the scanline
            # Hold those points in active_points list
            for i in range(min_col_active + 1, max_col_active):
                # shade the points between min_col_active
                # and max_active_col
                if shade_t == "flat":
                    img[i][scan] = average_color
                    elif shade_t == "gouraud":
                        img[i][scan] =
                            interpolate_color(min_col_active,
                                              max_col_active, i,
                                          img[min_col_active][scan],
                                          img[max_col_active][scan])
```

Έχοντας ολοκληρώσει τον αλγόριθμο για την πλήρωση τριγώνου, υλοποιείται η συνάρτηση για τον χρωματισμό του αντικειμένου. Μέσα στον πίνακα faces , ύστερα από τροποποιήσεις βρίσκονται πλέον οι συντεταγμένες των κορυφών κάθε τριγώνου και στον πίνακα depths Of Triangles αποθηκεύονται τα βάθη κάθε τριγώνου. Έπειτα, ταξινομείται ο πίνακας με τα βάθη και αντιστοίχως ταξινομείται και ο πίνακας faces για να ξεκινήσει ο χρωματισμός.

```python
def render(verts2d: list, faces: list, vcolors: list, depth: list,
               shade_t: str):
    # Initialize the image
    # Make appropriate changes for faces list
    # Calculate the depth of triangles

    for triangle in faces:
        shade_triangle(image, triangle,
                           triangle_colors[faces.index(triangle)], shade_t)


```

Παραδοχές και Παρατηρήσεις

Λόγω της δομής του αλγορίθμου για την πλήρωση τριγώνου κατά τον οποίο όλες οι πλευρές θεωρούνται ενεργές και λόγω της παραδοχής ότι κάποια pixels που ανήκουν σε περισσότερα από ένα τρίγωνα χρωματίζονται περισσότερες από μια φορές παίρνει αρκετά δευτερόλεπτα για το render της εικόνας. Είναι αρκετά δύσκολο ωστόσο να γνωρίζουμε εκ των προτέρων ποια pixels ανήκουν σε ποια τρίγωνα και να χρωματιστούν μόνο μια φορά. Επίσης, το render της εικόνας καθυστερεί και λόγω της διαχείρισης κάποιων δεδομένων και της ταξινόμησής τους.

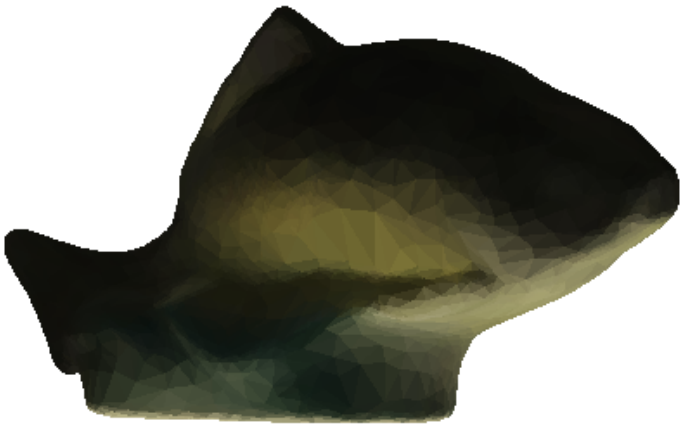

Figure 1: flat shading

Figure 2: gouraud shading