Γραφική με Υπολογιστές
Τομέας Ηλεκτρονικής και Υπολογιστών
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Α.Π.Θ.

8ο Εξάμηνο
Άνοιξη 2018

Γραφική με Υπολογιστές *2018*
Εργασία *#1 :* Πλήρωση τριγώνων

Κοσμάς Τσιάκας
ΑΕΜ: 8255

4/4/2018

1. Περιγραφή της λειτουργίας και του τρόπου κλήσης των προγραμμάτων

1.1 Συναρτήσεις demo

Αρχικά, οι δύο συναρτήσεις demo φορτώνουν τα δεδομένα από το αρχείο *cat.mat* και στη συνέχεια καλούν την συνάρτηση objectPainter με το κατάλληλο όρισμα, *Flat* ή *Gouraud*, για την διαδικασία του χρωματισμού των εικόνων. Μετά την εκτέλεση της συνάρτησης, εμφανίζεται το αποτέλεσμα με την χρήση της συνάρτησης *imshow*.

1.2 Συναρτήση objectPainter

Στην συνάρτηση *objectPainter* αρχικοποιούμε τον πίνακα στον οποίο θα σχεδιαστεί η εικόνα μας, μεγέθους 1150x1300 και αρχική τιμή 1 σε κάθε κελί του. Στην συνέχεια, δημιουργούμε τον πίνακα *triangleD*, ο οποίος περιέχει για κάθε ένα από τα K τρίγωνα, το μέσο βάθος των κορυφών του. Ταξινομούμε τον πίνακα αυτό με αύξουσα σειρά και κρατάμε τα *indexes*, με βάση τα οποία ταξινομούμε στην συνέχεια τον πίνακα *F* και με αυτόν τον τρόπο θα μπορέσουμε να χρωματίσουμε αρχικά τα τρίγωνα που βρίσκονται σε μεγαλύτερο βάθος και στη συνέχεια τα κοντινότερα.

Στη συνέχεια, για κάθε ένα από τα K τρίγωνα, με την σειρά που έχει οριστεί από την ταξινόμηση, καλούμε την συνάρτηση χρωματισμού τριγώνων, *triPaintFlat* ή *triPaintGouraud* ανάλογα με την τιμή που έχει το όρισμα *painterType*. Ως ορίσματα στην συνάρτηση αυτή, δίνονται η εικόνα *I*, οι συντεταγμένες των τριών κορυφών του τριγώνου και το χρώμα στις κορυφές αυτές.

1.3 Συναρτήσεις triPaintFlat & triPaintGouraud

Στις συναρτήσεις αυτές εκτελείται η διαδικασία χρωματισμού των τριγώνων. Πιο συγκεκριμένα, κάθε μία εκτέλεση της, προσθέτει και ένα χρωματισμένο τρίγωνο στην συνολική εικόνα. Στην περίπτωση της *triPaintFlat*, απλά χρησιμοποιείται ο μέσος όρος των χρωμάτων των κορυφών, ενώ στην *triPaintGouraud* καλείται μια επιπλέον συνάρτηση, η *findColor*, η οποία εκτελεί την γραμμική παρεμβολή για τον υπολογισμό του χρώματος κάθε pixel.

2. Περιγραφή της διαδικασίας χρωματισμού τριγώνων

2.1 Συνάρτηση triPaintFlat

Στην συνάρτηση αυτή μας δίνονται ως ορίσματα οι συντεταγμένες των τριών κορυφών, το χρώμα των κορυφών αυτών, καθώς και η υπόλοιπη εικόνα. Για την εκτέλεση του *scanline* αλγορίθμου, δημιουργείται μια δομή με το όνομα *bucket*, η οποία περιέχει κάποια συγκεκριμένα στοιχεία για τις τρεις πλευρές του τριγώνου.

Πιο συγκεκριμένα, περιέχει τα εξής:

- *yMax, yMin,* δηλαδή την μέγιστη και την ελάχιστη τιμή του y για κάθε πλευρά
- x, δηλαδή την τιμή της τετμημένης για την κορυφή η οποία έχει το ελάχιστο y
- *sign,* δηλαδή την κλίση της πλευράς (0, 1 ή -1)

- *dX, dY,* δηλαδή την απόλυτη διαφορά των τετμημένων και τεταγμένων των κορυφών
- *sum*, μια μεταβλητή η οποία αρχικοποιείται στο 0 και χρησιμοποιείται για τον μετ' έπειτα υπολογισμό του x, καθώς αλλάζει το scanline

Για την συμπλήρωση των δεδομένων της δομής, ελέγχουμε τις πλευρές του τριγώνου ανά δύο και τις συγκρίνουμε μεταξύ τους.

Υπάρχει ακόμη μια ίδια δομή, η *activeList*, η οποία περιέχει ακριβώς τα ίδια πεδία με την προηγούμενη, απλά θα χρησιμοποιηθεί για την αποθήκευση των ενεργών πλευρών, καθώς μεταβάλλεται το scanline.

Ορίζουμε τα όρια του y στα οποία θα πραγματοποιηθεί η σάρωση, ανάμεσα στο ελάχιστο και το μέγιστο y από όλες τις πλευρές (y*min* και *ymax).*

Όσον αφορά το χρώμα με το οποίο θα γίνει η πλήρωση του τριγώνου, στην συγκεκριμένη περίπτωση, απλώς υπολογίζουμε το μέσο όρο του χρώματος των κορυφών, με χρήση της συνάρτησης *mean*.

Στη συνέχεια, εκτελείται μια επανάληψη με το *y* να παίρνει τιμές από το *ymin* έως το *ymax*, όπου και πραγματοποιείται ο χρωματισμός του τριγώνου με τα εξής βήματα.

- Ελέγχεται αν πρέπει να αφαιρέσουμε κάποια πλευρά από την δομή *activeList*, κάτι το οποίο συμβαίνει όταν η πλευρά αυτή έχει τιμή *yMax* μικρότερη ή ίση από την τρέχουσα τιμή του *y*.
- Για κάθε μία πλευρά που περιέχεται στην δομή *bucket*, ελέγχουμε αν το *yMin* της είναι ίσο με την τρέχουσα τιμή του *y*. Σε περίπτωση που αυτό ισχύει, προσθέτουμε την πλευρά αυτή στην δομή με τις ενεργές πλευρές.
- Ταξινομούμε τις ενεργές πλευρές με βάση την κλίση και το *x* τους.
- Ελέγχουμε αν η δομή με τις ενεργές πλευρές είναι άδεια ή περιέχει μόνο ένα στοιχείο, κατάσταση κατά την οποία πρέπει να συνεχίσουμε στο επόμενο *y* και δεν ζωγραφίζουμε κάποιο pixel.
- Εδώ θεωρούμε ότι πάντα θα έχουμε δύο ενεργές πλευρές στην δομή μας. Συνεπώς, χρωματίζουμε τα pixel τα οποία βρίσκονται ανάμεσα στο *x* των δύο ενεργών πλευρών με το χρώμα που υπολογίστηκε προηγουμένως.
- Για κάθε μία από τις ενεργές πλευρές, αυξάνουμε το *x* τους, με βάση την κλίση της πλευράς. Ακόμα, ελέγχουμε αν πρόκειται για κάθετη πλευρά, οπότε το x δεν αυξάνεται. Στην συνέχεια, στην μεταβλητή *sum* προσθέτουμε την τιμή του *dX* και εάν το *dY* είναι διάφορο του μηδενός, το αφαιρούμε από το *sum*.
- Με τον τρόπο αυτό, υπολογίζουμε την τιμή που θα έχει το *x* στην επόμενη επανάληψη, το οποίο θα είναι και η θέση από την οποία θα αρχίσει ο χρωματισμός της επόμενης γραμμής του τριγώνου.

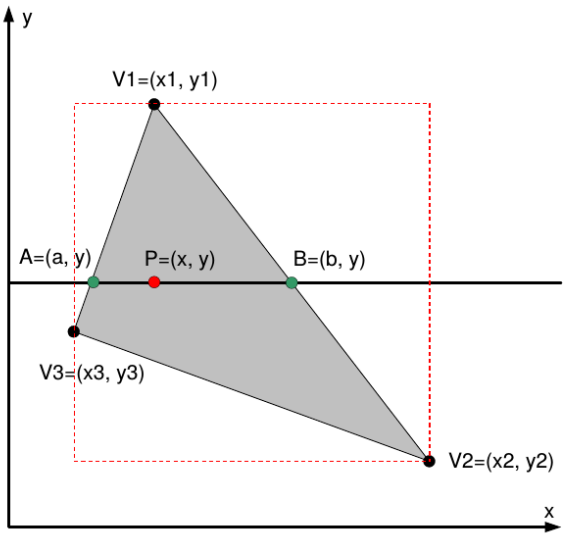
2.1 Συνάρτηση triPaintGouraud

Η βασική ιδέα της συνάρτησης αυτής ταυτίζεται με την *triPaintFlat*. Υπάρχουν όμως βασικές διαφορές στον υπολογισμό του χρώματος. Αρχικά, για να χρησιμοποιήσουμε τον αλγόριθμο πλήρωσης του Gouraud, χρειαζόμαστε περισσότερα δεδομένα στην δομή μας. Επομένως, εκτός από όσα υπήρχαν πριν, θα προσθέσουμε τα εξής στοιχεία:

- *normColor,* το οποίο πρόκειται για την κανονικοποιημένη τιμή του χρώματος, δηλαδή η τιμή κατά την οποία θα μεταβάλλεται το χρώμα ενός pixel, καθώς κινούμαστε πάνω σε μία πλευρά. Προκύπτει από το πηλίκο της διαφοράς των χρωμάτων των δύο κορυφών μιας πλευράς προς το μήκος της πλευράς.
- *totLength,* το οποίο πρόκειται για το μήκος της πλευράς.
- *startingColor,* το οποίο πρόκειται για το χρώμα της κορυφής με την μικρότερη τεταγμένη της πλευράς.
- *xOfymin,* το οποίο αρχικά ταυτίζεται με το *x* και η τιμή του αυτή παραμένει σταθερή, σε αντίθεση με το *x*, το οποίο μεταβάλλεται συνεχώς.

Η συνάρτηση αυτή διαφοροποιείται εντελώς από την *triPaintFlat* στο σημείο υπολογισμού του χρώματος για τον χρωματισμό των pixel.

Χρησιμοποιείται το παρακάτω σχήμα για αναφορά.



Αρχικά, χρησιμοποιούμε γραμμική παρεμβολή ανάμεσα στα σημεία V1 και V3 για τον υπολογισμό του χρώματος στο σημείο A και ανάμεσα στα σημεία V1 και V2 για τον υπολογισμό του χρώματος στο B. Η τετμημένη του σημείου A, δίνεται από την τιμή x, η οποία βρίσκεται αποθηκευμένη για την πλευρά αυτή στην δομή *activeList*. Υπολογίζεται η ευκλείδια απόσταση από το V3 έως το A και το χρώμα αυτό ως το χρώμα που έχει η κορυφή V3 (το αντίστοιχο *startingColor* της δομής) προσθέτοντας το κανονικοποιημένο χρώμα (*normColor*) επί την απόσταση V3-A.

Αντίστοιχη διαδικασία συμβαίνει και για την δεύτερη ενεργή πλευρά.

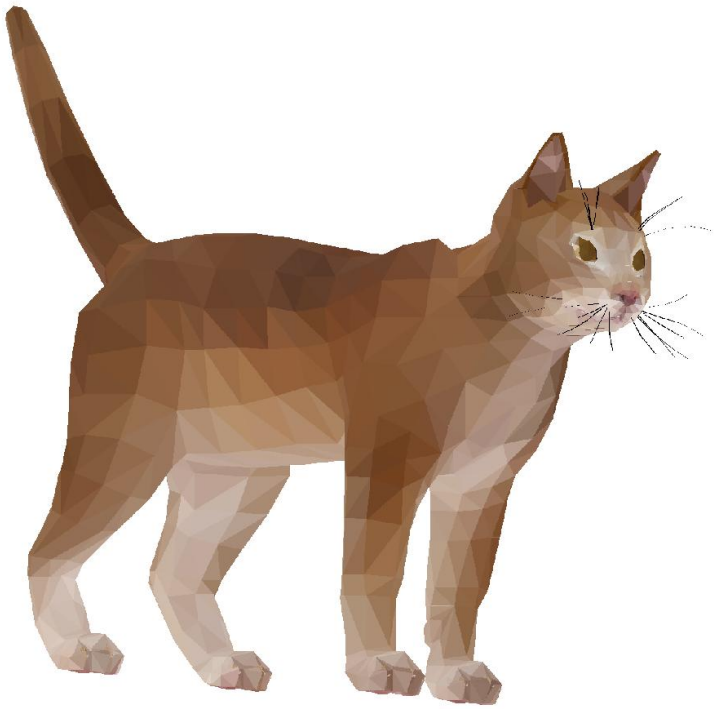
Στη συνέχεια, καλείται η συνάρτηση *findColor* για κάθε pixel που βρίσκεται από το a έως και το b, με ορίσματα την τιμή του *x* το οποίο θέλουμε να χρωματίσουμε, τα άκρα του τμήματος AB, και το χρώμα στα σημεία A και B, που υπολογίστηκαν προηγουμένως. Μέσα στην συνάρτηση *findColor*, για να υπολογίσουμε το χρώμα στο σημείο P, πρέπει να γνωρίζουμε κατά πόσο μεταβάλλεται το χρώμα για κάθε pixel στο τμήμα αυτό. Επομένως, υπολογίζουμε την διαφορά των χρωμάτων στα A και B, την διαιρούμε με την απόσταση AB, την πολλαπλασιάζουμε επί την απόσταση AP και στο τέλος προσθέτουμε το αρχικό χρώμα του σημείου A για να προκύψει η τελική τιμή του.

Στην περίπτωση που τα άκρα A και B ταυτίζονται για δύο ενεργές πλευρές, τότε ως χρώμα δίνουμε τον μέσο όρο του χρώματος των κορυφών, καθώς διαφορετικά η διαφορά τους θα οδηγούσε στην δημιουργία μαύρων pixel στην εικόνα.

3. Ενδεικτικά αποτελέσματα & σχόλια

Ο χρόνος εκτέλεσης των script είναι περίπου στα 90-100 δευτερόλεπτα. Εκτός από την σχετικά μεγάλη πολυπλοκότητα του αλγορίθμου και τους πολλούς ελέγχους, θεωρείται ότι η χρήση struct στο MATLAB επιβάρυνε ακόμη περισσότερη την εκτέλεση του προγράμματος, απ' ότι αν είχαν χρησιμοποιηθεί arrays, και γι' αυτό το λόγο η εκτέλεση απαιτεί περισσότερο χρόνο από το επιθυμητό.

3.1 demoFlat



Κατά την εκτέλεση του demoFlat script, το αποτέλεσμα προκύπτει πολύ ικανοποιητικό, καθώς διακρίνονται όλες οι λεπτομέρειες στην γάτα, το βάθος της εικόνας και είναι εμφανές ότι κάθε τρίγωνο έχει ένα διαφορετικό χρώμα το οποίο παραμένει σταθερό σε όλη την έκταση του.

3.2 demoGouraud

Κατά την εκτέλεση του demoGouraud script, το αποτέλεσμα που προκύπτει είναι πολύ βελτιωμένο σε σχέση με το προηγούμενο, καθώς φαίνεται ότι το χρώμα μεταβάλλεται με πολύ ομαλό τρόπο και προκύπτουν οι λεπτομέρειες του βάθους στην εικόνα.

3.3 Bonus

Αν η ταξινόμηση των στοιχείων στον αρχικό πίνακα των τριγώνων γίνει με τον ακριβώς αντίστροφο τρόπο, μπορούμε να δούμε την γάτα από την ακριβώς ανάποδη πλευρά.