ECE433: ΓΡΑΦΙΚΑ ΗΥ

Χειμερινό Εξάμηνο 2021-2022

Εργασία 1

Ομάδα φοιτητών #8

Κουτσούκης Νικόλαος - 2907

## Άσκηση 1

**Bresenham algorithm**

Ο αλγόριθμος γραμμής του Bresenham είναι ένας αλγόριθμος γραμμικής σχεδίασης που καθορίζει τα σημεία ενός n-διαστάσεων πίνακα που πρέπει να επιλεγούν προκειμένου να σχηματιστεί μια στενή προσέγγιση σε μια ευθεία γραμμή μεταξύ δύο σημείων. Χρησιμοποιείται συνήθως για τη σχεδίαση πρωτόγονων γραμμών σε μια εικόνα bitmap (π.χ. σε μια οθόνη υπολογιστή), καθώς χρησιμοποιεί μόνο πρόσθεση, αφαίρεση και μετατόπιση ακέραιων αριθμών, όλα αυτά είναι πολύ φθηνές λειτουργίες σε κοινά χρησιμοποιούμενα σύνολα εντολών υπολογιστή όπως το x86\_64. Είναι ένας αλγόριθμος αυξητικού σφάλματος. Είναι ένας από τους πρώτους αλγόριθμους που αναπτύχθηκαν στον τομέα των γραφικών υπολογιστών. Παρακάτω θα δούμε την υλοποίηση του σε ψευδό κώδικα για -1 < m < 1. Αλλά και με μια μικρή τροποποίηση ώστε με αρχικό κόκκινο και τελικό μπλε χρώμα στα άκρα της γραμμής, υπολογίζεται το σωστό χρώμα σε κάθε ενδιάμεσο pixel της μεταβάλλοντας το ομαλά.

BresenhamLine() {

Μ = (yEnd – y0) / (xEnd – x0)

if(x0 > xEnd)

x0 = xEnd

y0 = yEnd

xEnd = x0

yEnd = y0

dx = |xEnd – x0|

dy = |yEnd – y0|

p = 2\*dy – dx

x = x0

y = y0

drawPoint(x, y)

while(x <=xEnd) {

x++

if(p < 0)

if( M < 1 && y0 < yEnd)

y++

else

y--

p += 2\*dy

else

y++

p += 2\*(dy – dx)

drawPoint(x, y)

}

}

Για να υλοποιηθεί η ομαλή μετάβαση χρώματος τροποποιήθηκε η drawPoint.

Αρχικά τα χρώματα RGB είναι σε συνάρτηση του t οπού t είναι μια μεταβαλλόμενη τιμή η οποία αυξάνεται από 0 σε 1 με βήμα 1/dx, όπου dx είναι η μετατόπιση του x και ισχύει R = 1 – t, G = 0, B = t.

drawPoint(x, y) {

R = 1 – t, G = 0, B = t

glcolor3d(R,G,B)

t += 1/dx

colorPixel(x, y)

}

Επίσης αν θέλουμε να λειτουργεί προς όλες τις κατευθύνσεις, δηλαδή και για |m| >= 1, απλά ανταλλάσσουμε τα x, y για |m| >= 1.

## Άσκηση 2

Η υλοποίηση της άσκησης έγινε σε περιβάλλον ubuntu με version 20.04.3, ενώ τα χαρακτηριστικά του μηχανήματος είναι: CPU AMD RYZEN 7 3800X(8 πυρήνες 16 threads), GPU NVIDIA 710 GT και 16 GB RAM.

Η δομή του προγράμματος αποτελείται από τις παρακάτω συναρτήσεις:

1. **Main**: Η κύρια συνάρτηση του προγράμματος.
2. **myInit**: Αρχικοποίησή τις μεταβλητές με το άνοιγμα του προγράμματος.
3. **myMenu**: Φροντίζει την σωστή επιλογή δυνατότητας στον χρήστη.
4. **myMouse**: Οποιαδήποτε επαφή με το ποντίκι λειτουργεί λόγω αυτής της συνάρτησης.
5. **keyEvent:** Κλείσιμο προγράμματος με το πλήκτρο Q.
6. **drawPoint**: Ζωγραφίζει το επιλεγμένο pixel.
7. **drawLinePoint**: Ζωγραφίζει το επιλεγμένο pixel αλλά και δίνει το σωστό χρώμα για την υλοποίηση της γραμμής, χρησιμοποιείται στην drawLine.
8. **DrawLine**: Σχηματίζει την γραμμή που έχει επιλέξει ο χρήστης με τον αλγόριθμο του bresenham.
9. **DrawEllipse**: Σχηματίζει την έλλειψη που έχει επιλέξει ο χρήστης με τον αλγόριθμο του bresenham.

Αρχικά με την έναρξη του προγράμματος είναι στην επιλογή που απλά εμφανίζει το pixel του αριστερού κλικ του χρήστη, ενώ με το δεξί κλικ εμφανίζεται το μενού του προγράμματος:

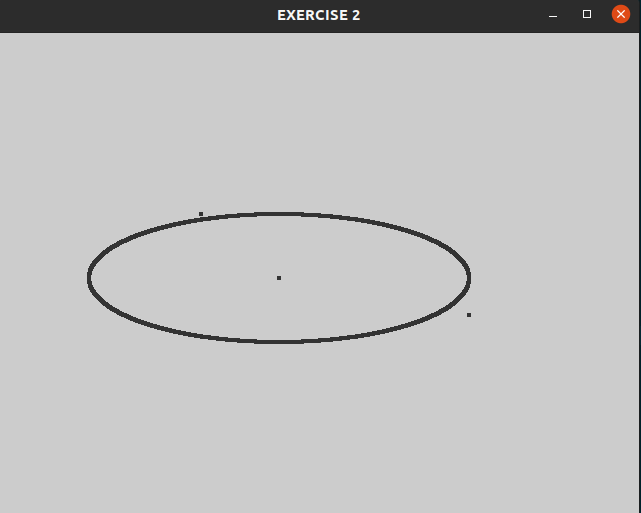
Εικόνα που περιέχει κείμενο, ρολόι

Περιγραφή που δημιουργήθηκε αυτόματα

Η πρώτη επιλογή**, Line**, δίνει την δυνατότητα στον χρήστη να δημιουργήσει μια γραμμή, δίνοντας την αρχική και τελική θέση με το αριστερό κλικ. Πχ ο χρήστης δίνει αρχική θέση την (100, 240) και τελική θέση (500, 240) το αποτέλεσμα θα είναι:



Η δεύτερη επιλογή, **Ellipse**, ο χρήστης μπορεί να σχεδιάσει μια έλλειψη, δίνοντας αρχικά το κέντρο και στην συνέχεια το πλάτος και ύψος της έλλειψης, με το αριστερό κλικ.



Ενώ με την τρίτη επιλογή, **Clear**, καθαρίζεται η οθόνη και επιστρέφει στην default δυνατότητα, η οποία είναι να εμφανίζει το pixel που επιλεγεί ο χρήστης.

## Άσκηση 3

**Scanline Polygon filling Algorithm**

Ο αλγόριθμος scanline είναι βασικά η κάλυψη πολυγώνων χρησιμοποιώντας οριζόντιες γραμμές ή γραμμές σάρωσης. Ο σκοπός του αλγόριθμου SLPF είναι να γεμίσει (χρωματίσει) τα εσωτερικά εικονoστοιχεία ενός πολυγώνου με δεδομένες μόνο τις κορυφές του σχήματος.

Αυτός ο αλγόριθμος λειτουργεί τέμνοντας τη γραμμή σάρωσης με άκρες πολυγώνου και γεμίζει το πολύγωνο μεταξύ ζευγών τομών.

Για την υλοποίηση του έχει τρία βασικά στοιχεία:

* **Edge Buckets**: Περιέχει πληροφορίες μιας άκρης. Υπάρχουν τρεις μεταβλητές σε κάθε Edge Buckets οι οποίες είναι:
  + **Ymax:** Είναι η μέγιστη τιμή του y για το συγκεκριμένο bucket.
  + **Xofymin**: Η τιμή του x για την στην γραμμή που βρίσκεται η scanline.
  + **Slopeinverse**: Η αντιστροφή τιμή της κλίσης της γραμμής.
* **Edge Table**: Αποτελείται από πολλές λίστες ακμών -> κρατά όλες τις ακμές που συνθέτουν το σχήμα. Κατά τη δημιουργία ακμών, οι κορυφές της ακμής πρέπει να ταξινομηθούν από αριστερά προς τα δεξιά και οι άκρες να διατηρούνται με αύξουσα σειρά yMin. Το γέμισμα ολοκληρώνεται μόλις αφαιρεθούν όλες οι άκρες από το ET.
* **Active List**: Το IT διατηρεί τις τρέχουσες άκρες που χρησιμοποιούνται για τη συμπλήρωση του πολυγώνου. Οι άκρες ωθούνται στο AL από τον πίνακα άκρων όταν το yMin μιας άκρης είναι ίσο με την τρέχουσα γραμμή σάρωσης που υποβάλλεται σε επεξεργασία. Η Active List θα ταξινομείται εκ νέου μετά από κάθε πέρασμα.

Ο ψευδοκωδικας της Scanline:

Scanline(){

storeEdgesToTable() //Αποθήκευση των ακρών του πολυγώνου στον Edge Table.

while(y < yMax) { //Σάρωση όλο το παράθυρο από κάτω προς τα επάνω.

CopyEdgesToActiveTuple() //Αντιγραφή ακρών στο Active Tuple.

sortActiveTuple(xofymin) //ταξινόμηση κατά xofymin στο Active Tuple.

removeEdges() //Αφαίρεση ακρών στο Active Tuple διότι

//βρίσκονται στην κορυφή τους.

If() //Γέμισμα της γραμμής συμπληρώνοντας ζευγάρι ακμών.

drawline() //Γέμισμα στη γραμμή που ορίζει η if().

updatexbyslope() //Ενημέρωση την μεταβλητή xofymin προσθέτοντας την slopeinverse στον κάθε bucket.

}

}

Οι τιμές που περιέχονται στο Active Edge Table και Active Edge List κατά την διαρκεια εκτελεσης της ScanLine για το συγκεκριμένο πολύγωνο:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Active Edge Table:   |  |  |  |  | | --- | --- | --- | --- | | 8 |  |  |  | | 7 | CD | FE |  | | 6 | CD | FE |  | | 5 | BC | GF |  | | 4 | AB | HG |  | | 3 | AB | HG |  | | 2 | AB | JI | HI,HG | | 1 | AB | JI |  | |

Active Edge List: line 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **AB** | **JI** |  |  |
| maxY | 5 | 3 |  |  |
| currentX | 4 | 7 |  |  |
| xIncr | -3/4 | 1/2 |  |  |

Active Edge List: line 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **AB** | **JI** | **HI** | **HG** |
| maxY | 5 | 3 | 3 | 5 |
| currentX | 13/4 | 15/2 | 10 | 10 |
| xIncr | -3/4 | 1/2 | -2 | 2/3 |

Active Edge List: line 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **AB** | **HG** |  |  |
| maxY | 5 | 5 |  |  |
| currentX | 10/4 | 32/3 |  |  |
| xIncr | -3/4 | 2/3 |  |  |

Active Edge List: line 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **AB** | **HG** |  |  |
| maxY | 5 | 5 |  |  |
| currentX | 7/4 | 34/3 |  |  |
| xIncr | -3/4 | 2/3 |  |  |

Active Edge List: line 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **BC** | **GF** |  |  |
| maxY | 6 | 6 |  |  |
| currentX | 1 | 12 |  |  |
| xIncr | 3 | -2 |  |  |

Active Edge List: line 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **CD** | **FE** |  |  |
| maxY | 8 | 8 |  |  |
| currentX | 4 | 10 |  |  |
| xIncr | 1/2 | 1/2 |  |  |

Active Edge List: line 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **CD** | **FE** |  |  |
| maxY | 8 | 8 |  |  |
| currentX | 9/2 | 21/2 |  |  |
| xIncr | 1/2 | 1/2 |  |  |

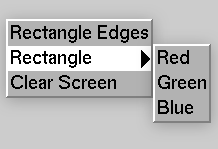
## **Άσκηση 4**

Η υλοποίηση της άσκησης έγινε σε περιβάλλον ubuntu με version 20.04.3, ενώ τα χαρακτηριστικά του μηχανήματος είναι: CPU AMD RYZEN 7 3800X(8 πυρήνες 16 threads), GPU NVIDIA 710 GT και 16 GB RAM.

Η δομή του προγράμματος αποτελείται από τις παρακάτω συναρτήσεις:

1. **Main**: Η κύρια συνάρτηση του προγράμματος.
2. **myInit**: Αρχικοποίησή τις μεταβλητές με το άνοιγμα του προγράμματος.
3. **myMenu**: Φροντίζει την σωστή επιλογή δυνατότητας στον χρήστη.
4. **myMouse**: Οποιαδήποτε επαφή με το ποντίκι λειτουργεί λόγω αυτής της συνάρτησης.
5. **keyEvent**: Κλείσιμο προγράμματος με το πλήκτρο Q.
6. **drawPoint**: Ζωγραφίζει το επιλεγμένο pixel.
7. **DrawLine**: Ζωγραφίζει την επιλεγμένη γραμμή.
8. **DrawRectangle**: Αποθηκεύει τις άκρες στο edge table και καλεί την scanLinefill.
9. **InsertionSort**: Κάνει sorting τις άκρες για οποιαδήποτε συνάρτηση την καλέσει.
10. **StoreEdgeInTuple**: Αποθηκεύει τις τιμές του bucket στο bucket.
11. **RemoveEdgeByYmax**: Αφαιρεί τις άκρες τις οποίες δεν χρησιμοποιούνται πλέον.
12. **Updatebyslopeinv**: κάνει update την αναστροφή τιμή της κλίσης.
13. **ScanlineFill**: Η βασική συνάρτηση γεμίσματος πολυγώνου.

Αρχικά με την έναρξη του προγράμματος είναι στην επιλογή που απλά εμφανίζει το pixel του αριστερού κλικ του χρήστη, ενώ με το δεξί κλικ εμφανίζεται το μενού του προγράμματος:

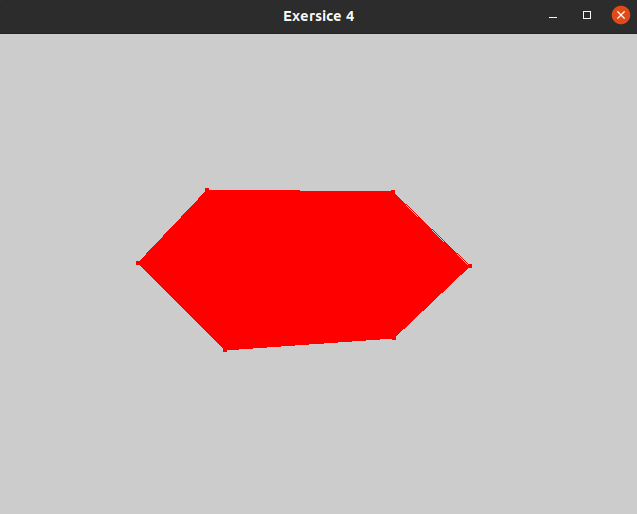


Η πρώτη επιλογή, **Rectangle Edges**, δίνει την δυνατότητα στον χρήστη να επιλέξει τις κορυφές που επιθυμεί για την δημιουργία ενός πολυγώνου, με όριο τις 10 κορυφές.

Εικόνα που περιέχει κείμενο, ουρανός, υπαίθριος, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματα

Η δεύτερη επιλογή**, Rectangle**, ο χρήστης μπορεί να χρωματίσει το πολύγωνο που σχεδίασε από την πρώτη επιλογή, διαλέγοντας και χρώμα, κόκκινο, πράσινο ή μπλε, από το υπομενου που εμφανίζεται.



Ενώ με την τρίτη επιλογή, **Clear Screen**, καθαρίζεται η οθόνη και επιστρέφει στην default δυνατότητα, η οποία είναι να εμφανίζει το pixel που επιλεγεί ο χρήστης.