ECE433: ΓΡΑΦΙΚΑ ΗΥ

Χειμερινό Εξάμηνο 2021-2022

Εργασία 1

Ομάδα φοιτητή #4

Χωροπανίτης Πασχάλης – 2453

# Γενικές πληροφορίες για την εργασία.

Συνολικός χρόνος ολοκλήρωσης της εργασίας: 1 εβδομάδα (3 ημέρες για τις πρώτες 2 ασκήσεις και άλλες 3 για τις υπόλοιπες 2).

Στο παραδοτέο περιέχονται τα εξής:

* bresenham\_exercise.c άσκηση 2.
* scan\_polygon\_exercise.c άσκηση 4.
* Makefile
* template\_gr.docx
* \*freeglut φάκελος για την μεταγλώττιση στα Windows
* \*freeglut.dll .dll αρχείο για την εκτέλεση στα Windows

\*(Προαιρετικά) Απαραίτητα μόνο σε περιβάλλον Windows.

Γενικές πληροφορίες για τις προγραμματιστικές ασκήσεις:

* Οι ασκήσεις είναι προγραμματισμένες σε C, σε περιβάλλον Windows Subsystem for Linux (WSL 2) με εγκατεστημένο OS το Ubuntu 20.04 (LTS).
* Οι δύο ασκήσεις έχουν δοκιμαστεί και σε native linux install (Ubuntu 20.04 (LTS)), καθώς και στα Windows 10 Home, και δουλεύουν χωρίς κανένα πρόβλημα.

Χρήσιμα χαρακτηριστικά του μηχανήματος στο οποίο δούλεψα:

* Laptop Lenovo Ideapad Gaming 3
* Renderer: NVIDIA GeForce GTX 1650 Ti (4096 MB vram)
* OpenGL Version: 4.6 (4.6.0 NVIDIA 497.29)

Οδηγίες για compile/link/execution των ασκήσεων σε περιβάλλον Linux:

* Στην παρακάτω εντολή βρίσκονται όλα τα πακέτα που χρειάστηκα για την ανάπτυξη των ασκήσεων:
  + sudo apt-get install mesa-utils freeglut3 freeglut3-dev
* Στο .zip αρχείο παράδοσης περιέχεται και ένα Makefile για την μεταγλώττιση των αρχείων μας.
* Για την μεταγλώττιση και των δύο ασκήσεων εκτελέστε την παρακάτω εντολή:
  + make ή make all
* Για την μεταγλώττιση μόνο μίας άσκησης, εκτελέστε την παρακάτω εντολή:
  + make bresenham\_exercise για την άσκηση 2.
  + make scan\_polygon\_exercise για την άσκηση 4.
* Για την εκτέλεση των ασκήσεων, εκτελέστε την παρακάτω εντολή:
  + ./bresenham\_exercise για την άσκηση 2.
  + ./scan\_polygon\_exercise για την άσκηση 4.
* Για την εκκαθάριση του φακέλου από τα εκτελέσιμα και τα object αρχεία της μεταγλώττισης, εκτελέστε την παρακάτω εντολή:
  + make clean

(Προαιρετικό) Οδηγίες για compile/link/execution των ασκήσεων σε περιβάλλον Windows:

* Η μεταγλώττιση των ασκήσεων στα Windows γίνεται πάλι με το ίδιο Makefile, αλλά προϋποθέτει την εγκατάσταση κάποιου third-party μεταγλωττιστή (gcc). Στην περίπτωσή μου χρησιμοποιώ το MinGW.
* Στο .zip αρχείο παράδοσης περιέχεται επίσης ένας φάκελος με όνομα freeglut και ένα αρχείο freeglut.dll, τα οποία είναι απαραίτητα για την μεταγλώττιση και εκτέλεση των ασκήσεων στα Windows.
* Για την μεταγλώττιση και των δύο ασκήσεων εκτελέστε την παρακάτω εντολή:
  + make ή make all
* Για την μεταγλώττιση μόνο μίας άσκησης, εκτελέστε την παρακάτω εντολή:
  + make bresenham\_exercise.exe για την άσκηση 2.
  + make scan\_polygon\_exercise.exe για την άσκηση 4.
* Για την εκτέλεση των ασκήσεων, απλά τρέξτε τα εκτελέσιμα .exe αρχεία.
* Για την εκκαθάριση του φακέλου από τα εκτελέσιμα και τα object αρχεία της μεταγλώττισης, εκτελέστε την παρακάτω εντολή:
  + make clean

# ΑΣΚΗΣΕΙΣ

## Άσκηση 1

Αλγόριθμος Bresenham για όλες τις κατευθύνσεις, ελαφρώς τροποποιημένος από το παράδειγμα στις διαλέξεις του μαθήματος.

Η κύρια ιδέα του αλγόριθμου είναι η εξής:

Έστω ότι η ευθεία που θέλουμε να σχεδιάσουμε είναι η y=mx με m 0<= m <= 1, δηλαδή στο πρώτο τεταρτημόριο, και αρχίζει από το pixel (0, 0).  
Τότε, πάντα αυξάνουμε το x κατά 1 καθώς κινούμαστε πάνω στην ευθεία και διαλέγουμε για το επόμενο y, αν θα χρειαστεί να πάμε στο y + 1 ή να μείνουμε στο παρόν y.  
Δηλαδή, αν το pixel (xi, yi) επιλέγει στο προηγούμενο βήμα τότε, το παρόν pixel της ακμής θα είναι το (xi + 1, yi) ή το (xi + 1, yi + 1).  
Η επιλογή του παρόντος pixel γίνεται εξετάζοντας την απόσταση της ευθείας από τα κέντρα των 2 υποψηφίων pixels, δηλαδή το πρόσημο της παράστασης:

e = m(xi + 1) – (yi + 0.5) με m – 1 <= e <= m.

Ένα πρόβλημα που συναντάμε εδώ, είναι ότι στον αλγόριθμο του Bresenham, θέλουμε να αποφύγουμε υπολογισμούς με float μεταβλητές, κάτι το οποίο είναι δύσκολο καθώς η εξίσωση μια ευθείας είναι y = mx + c, όπου m = (y2 – y1) / (x2 – x1).  
Για να λύσουμε λοιπόν αυτό το πρόβλημα, πολλαπλασιάζουμε και στις δυο πλευρές στην εξίσωση m = (y2 – y1) / (x2 – x1) με (x2 – x1).   
Επίσης τροποποιούμε το e να είναι e \* (x2 – x1). Για να αποφύγουμε τη σύγκριση με το 0.5, τροποποιούμε το e ακόμα μια φορά σε e \* (x2 – x1) \* 2.

Με κάποιους επιπλέον υπολογισμούς για να βρούμε την αρχική τιμή του e, φτάνουμε στον παρακάτω αλγόριθμο:

Bresenham’s Line Algorithm (x1, y1, x2, y2)

{

x = x1

y = y1

dx = x2 - x1

dy = y2 - y1

c0 = (r0, g0, b0)

c1 = (r1, g1, b1)

// some modifications for the algorithm to work also on the negative axes

if (dy < 0)

dy = -dy

stepy = -1

else stepy = 1

if (dx < 0)

dx = -dx

stepx = -1

else stepx = 1

if (dx > dy) // if slope is less than 1

p = 2dy – dx // initial value of the decision parameter

c = c0\*(1 - p) + c1\*p

while (x <= x2) // loop at each x along the line

drawPixel(x, y)

x += stepx

if (p < 0)

p = p + 2dy // 2dy is the incremental value of the decision parameter when p < 0

c = c + 2dy

else

p = p + 2dy – 2dx // 2dy – 2dx is the incremental value of the decision parameter when p >= 0

c = c + 2dy – 2dx

y += stepy

end

else // if slope is greater or equal to 1

p = 2dx – dy // initial value of the decision parameter

c = c1\*(1 - p) + c0\*p

while (y <= y2) // loop at each y along the line

drawPixel(x, y, c)

y += stepy

if (p < 0)

p = p + 2dx // 2dx is the incremental value of the decision parameter when p < 0

c = c + 2dx

else

p = p + 2dx – 2dy // 2dx – 2dy is the incremental value of the decision parameter when p < 0

c = c + 2dx – 2dy

x += stepx

end

}

## Άσκηση 2

Αρχικά, η άσκηση χωρίζεται σε δύο κομμάτια:

1. Σχεδίαση γραμμών προς όλες τις κατευθύνσεις χρησιμοποιώντας τον αλγόριθμο Bresenham.
2. Σχεδίαση ελλείψεων χρησιμοποιώντας πάλι τον αλγόριθμο Bresenham.

Ο χρήστης έχει την δυνατότητα να επιλέξει τη σχεδίαση γραμμής ή έλλειψης ή τον καθαρισμό της οθόνης από μενού που ενεργοποιείται με το δεξί πλήκτρο του ποντικιού.  
Πιο συγκεκριμένα, το μενού περιέχει τις παρακάτω επιλογές:

1. Line επιλογή αλγορίθμου σχεδίασης γραμμών
2. Ellipse επιλογή αλγορίθμου σχεδίασης έλλειψης
3. Color υπό-μενού που επιλέγει το χρώμα που επιθυμούμε για την σχεδίαση (red, green, blue)
4. Clear Screen καθαρισμός οθόνης
5. Quit τερματισμός προγράμματος (η αλλιώς με το κουμπί «q»)

Diagram

Description automatically generated with medium confidence

Το πρόγραμμα ξεκινάει έχοντας ως defaults την επιλογή 1 (σχεδίαση γραμμών) και το χρώμα κόκκινο.

Για την διευκόλυνση του χρήστη, ειδικά στην επιλογή των σημείων, το πρόγραμμα έχει ως φόντο τους άξονες x, y σε καρτεσιανό σύστημα συντεταγμένων. Επίσης, όταν επιλεχτεί ένα σημείο, εμφανίζονται στην οθόνη οι x και y συντεταγμένες του.

### Επιλογή 1) Σχεδίαση γραμμών προς όλες τις κατευθύνσεις χρησιμοποιώντας τον αλγόριθμο Bresenham.

Ο χρήστης, αφού έχει επιλέξει την επιλογή 1, πρέπει να επιλέξει τα δύο σημεία που χρειάζεται για να σχηματιστεί η ευθεία. Η επιλογή των σημείων γίνεται με το αριστερό πλήκτρο του ποντικιού, σημαδεύοντας πάνω στο σημείο που επιθυμούμε. Με την επιλογή του δεύτερου σημείου, σχηματίζεται και η ευθεία.

Για να σχηματιστεί η ευθεία στην οθόνη χρησιμοποιώ τον αλγόριθμο Bresenham από την άσκηση 1.  
O αλγόριθμος Bresenham υλοποιείται στην συνάρτηση draw\_line(x1, y1, x2, y2).

Όσο αφορά το φαινόμενο της ταύτισης, στην αρχή του προγράμματος γίνεται #define το macro ANTIALIASING\_ON. Για να σχηματιστεί η ευθεία χωρίς antialiasing, βάζουμε την παραπάνω σειρά σε σχόλια.

Το φαινόμενο της ταύτισης ήταν το σημείο που με δυσκόλεψε περισσότερο στην εργασία. Μετά από εκτεταμένη έρευνα, κατέληξα στο συμπέρασμα πως για να υλοποιήσω τον Αλγόριθμο από το Antialiasing\_Notes.pdf που μας επισυνάπτετε, έπρεπε να τροποποιήσω τον Αλγόριθμο του Bresenham να δουλεύει με subpixel συντεταγμένες για να λειτουργεί σωστά. Έτσι, ακολούθησα μια ελαφρώς διαφορετική υλοποίηση που βρήκα στο βιβλίο «Foundations of 3D Graphics Programming: Using JOGL and Java3D».

A picture containing shape

Description automatically generated

### Επιλογή 2) Σχεδίαση ελλείψεων χρησιμοποιώντας τον αλγόριθμο Bresenham.

Ο χρήστης, αφού έχει επιλέξει την επιλογή 2, πρέπει να επιλέξει με το ποντίκι (βλ. επιλογή 1) το κέντρο της έλλειψης και τα δύο σημεία (a, b) της περιφέρειας της. Με την επιλογή του τελευταίου σημείου (b), σχηματίζεται και η έλλειψη.

Για να σχηματιστεί η έλλειψη στην οθόνη χρησιμοποιώ πάλι τον αλγόριθμο Bresenham από την άσκηση 1, ελαφρώς τροποποιημένο για να σχηματίζει ελλείψεις αντί για ευθείες.  
O αλγόριθμος Bresenham για ελλείψεις υλοποιείται στην συνάρτηση draw\_ellipse(cx, cy, xradius, yradius), όπου cx, cy είναι το κέντρο της έλλειψης και xradius, yradius είναι το Α, Β της έλλειψης αντίστοιχα.  
Για την σχεδίαση της έλλειψης εκμεταλλευόμαστε την συμμετρία τεσσάρων σημείων που εμφανίζειται.

Diagram

Description automatically generated

Τέλος, η άσκηση δουλεύει και όταν αλλάζουμε το μέγεθος του παραθύρου (resize, fullscreen). Απλά, κάθε φορά που ο χρήστης αλλάζει το μέγεθος του παραθύρου, γίνεται καθαρισμός της οθόνης για να γίνει συγχρονισμός των καινούργιων συντεταγμένων του παραθύρου.

Chart

Description automatically generated

## Άσκηση 3

Γενικός αλγόριθμος γεμίσματος πολυγώνων με γραμμές σάρωσης.

// process the polygon edge after edge, and store in the edge table

// insert edges into table at scan-line associated with lowest end-point  
// after addition of any edge in an edge tuple, the tuple is sorted  
// using insertion sort, according to its currentX value.

// start from scanline 0  
// repeat until last scanline  
for (int i = 0; i < windowHeight; i++) // increment y by 1 (next scan line)  
 // move from ET to AET those edges whose ymin = y (entering edges)

for (int j = 0; j < EdgeTable[i].count; i++)  
 storeEdgeInAET(activeEdge)  
 end  
 // remove from AET those edges for which y = ymax (not involved in next scan line)  
 removeEdgeByYmax(activeEdge)  
 // sort AET  
 insertionSort(activeEdge)  
 // fill lines on scan line y by using pairs of x-coords from AET  
 j = 0  
 while (j < activeEdge.count)  
 // if both lines intersecting at the vertex are on the same side of the scanline,  
 // consider it as two points  
 // if lines intersecting at the vertex are at opposite sides of the scanline,  
 // consider it as only one point  
 if (coordCount mod 2 != 0)  
 x2 = activeEdge.currentX  
 ymax2 = activeEdge.ymax  
 // checking for intersection  
 if (x1 == x2)  
 // lines are towards top of the intersection or lines are towards bottom  
 if ((x1 == ymax1 && x2 != ymax2) || (x2 == ymax2 && x1 != ymax1))  
 x1 = x2  
 ymax1 = ymax2

// one line is towards top and other is towards botttom  
 else  
 coordCount++  
 else  
 coordCount++  
 // draw actual lines  
 draw(x1, i)  
 draw(x2, i)  
 else  
 x1 = activeEdge.currentX  
 ymax1 = activeEdge.ymax

// checking for intersection  
 if (x1 == x2)  
 // lines are towards top of the intersection or lines are towards bottom  
 if ((x1 == ymax1 && x2 != ymax2) || (x2 == ymax2 && x1 != ymax1))  
 x2 = x1  
 ymax2 = ymax1

// one line is towards top and other is towards botttom  
 else  
 coordCount++  
 else  
 coordCount++  
 j++  
 end  
// for each nonvertical edge remaining in AET, update x for new y  
updatexbyslopeinverse(activeEdge)  
end

Αρχικά γεμίζουμε τον πίνακα Edge Table του πολυγώνου.

Edge Table:

|  |  |  |
| --- | --- | --- |
| 8 |  |  |
| 7 |  |  |
| 6 | CD | FE |
| 5 | BC | GF |
| 4 |  |  |
| 3 |  |  |
| 2 | HI | HG |
| 1 | AB | JI |

Παρακάτω δίνονται αναλυτικά τα βήματα της επανάληψης ξεκινώντας από το scanline = 0 μέχρι το τέλος του σχήματός μας, δηλαδή για scanline = 8.

Βήμα 0:

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

Active Edge Tuple: 🡪 Empty.

|  |
| --- |
|  |
| maxY |
| currentX |
| xIncr |

Βήμα 1:

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

Active Edge Tuple: 🡪 Insert AB, JI

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

Βήμα 2:

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

Active Edge Tuple: Insert HI, HG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | AB | JI | HI | HG |
| maxY | 5 | 3 | 3 | 5 |
| currentX | 4-3/4 | 7+1/2 | 10 | 10 |
| xIncr | -3/4 | 1/2 | -2 | 2/3 |

Βήμα 3:

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

Active Edge Tuple: 🡪 Delete JI, HI

|  |  |  |
| --- | --- | --- |
|  | AB | HG |
| maxY | 5 | 5 |
| currentX | 2+1/2 | 10+2/3 |
| xIncr | -3/4 | 2/3 |

Βήμα 4:

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

Active Edge Tuple:

|  |  |  |
| --- | --- | --- |
|  | AB | HG |
| maxY | 5 | 5 |
| currentX | 2-1/4 | 11+1/3 |
| xIncr | -3/4 | 2/3 |

Βήμα 5:

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

Active Edge Tuple: 🡪 Delete AB, HG and Insert BC, GF

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

Βήμα 6:

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

Active Edge Tuple: Delete BC, GF and Insert CD, FE

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

Βήμα 7:

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

Active Edge Tuple:

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

Βήμα 8:

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

Active Edge Tuple: Delete CD, FE 🡪 Empty.

|  |
| --- |
|  |
| maxY |
| currentX |
| xIncr |

## Άσκηση 4

Αρχικά, η άσκηση χωρίζεται σε δύο κομμάτια:

1. Σχεδίαση του πολυγώνου.
2. Χρωματισμός – γέμισμα του πολυγώνου χρησιμοποιώντας τον αλγόριθμο YX για σάρωση πολυγώνων.

Ο χρήστης έχει την δυνατότητα να επιλέξει τη σχεδίαση ή γέμισμα του πολυγώνου και τον καθαρισμό της οθόνης από μενού που ενεργοποιείται με το δεξί πλήκτρο του ποντικιού.  
Πιο συγκεκριμένα, το μενού περιέχει τις παρακάτω επιλογές:

1. Draw Polygon επιλογή σχεδίασης πολυγώνου
2. Fill Polygon επιλογή γεμίσματος πολυγώνου
3. Color υπό-μενού που επιλέγει το χρώμα που επιθυμούμε για την σχεδίαση (red, green, blue)
4. Clear Screen καθαρισμός οθόνης
5. Quit τερματισμός προγράμματος (η αλλιώς με το κουμπί «q»)

Timeline

Description automatically generated with medium confidence

Το πρόγραμμα ξεκινάει έχοντας ως default το χρώμα κόκκινο.

Για την διευκόλυνση του χρήστη, ειδικά στην επιλογή των σημείων, το πρόγραμμα έχει ως φόντο τους άξονες x, y σε καρτεσιανό σύστημα συντεταγμένων. Επίσης, όταν επιλεχτεί ένα σημείο, εμφανίζονται στην οθόνη οι x και y συντεταγμένες του.

Το πρόγραμμα ξεκινάει περιμένοντας από τον χρήστη να επιλέξει τις κορυφές του πολυγώνου. Η επιλογή των κορυφών γίνεται με το αριστερό πλήκτρο του ποντικιού, σημαδεύοντας πάνω στο σημείο που επιθυμούμε.  
Ο χρήστης έχει την δυνατότητα να επιλέξει μέχρι 10 κορυφές για τον σχηματισμό του πολυγώνου. Αν θέλουμε περισσότερες κορυφές μπορούμε να αυξήσουμε την τιμή του MAXEDGES που γίνεται define στην αρχή του προγράμματος. Επίσης, να σημειωθεί πως ο ελάχιστος αριθμός κορυφών για να σχηματιστεί ένα πολύγωνο είναι 3. Οπότε ο χρήστης πρέπει να επιλέξει τουλάχιστον 3 κορυφές.

Ο χρήστης, αφού επιλέξει τις κορυφές που επιθυμεί, διαλέγει την επιλογή «Draw Polygon» από το μενού για να σχηματιστεί το πολύγωνο στην οθόνη.

Chart, radar chart

Description automatically generated  
Πολύγωνο με 5 κορυφές.

Chart, line chart

Description automatically generated  
Πολύγωνο με 10 κορυφές.

Η σχεδίαση του πολυγώνου γίνεται στην συνάρτηση drawPolygon() όπου αρχικοποιείται και γεμίζει ο πίνακας Edge Table.

Έπειτα, διαλέγοντας την επιλογή «Fill Polygon», γίνεται το γέμισμα του πολυγώνου.

Το γέμισμα του πολύγωνου γίνεται στην συνάρτηση scanlineFill().

Στην συνάρτηση scanlineFill, για κάθε γραμμή σάρωσης y, από την ελάχιστη ως τη μέγιστη y τιμή του πολυγώνου επαναλαμβάνεται η εξής διαδικασία:

1. Ενημερώνουμε την λίστα ενεργών πλευρών Active Edge Tuple (AET) από το y-bucket του πίνακα πλευρών, που αντιστοιχεί στην παρούσα γραμμή σάρωσης.
2. Διαγράφουμε από την λίστα AET τις πλευρές που το ymax είναι μεγαλύτερο ή ίσο από την γραμμή σάρωσης.
3. Ταξινομούμε την λίστα AET με τον αλγόριθμο insertion sort.
4. Γεμίζουμε γραμμές στην γραμμή σάρωσης y, χρησιμοποιώντας τα ζευγάρια πλευρών τις λίστας AET.
5. Για κάθε μη κάθετη πλευρά που μένει στην λίστα AET, ενημερώνουμε το x για το καινούργιo y.

Chart

Description automatically generated

Chart

Description automatically generated

Για να προχωρήσουμε σε νέα επιλογή κορυφών, πρέπει να έχει γίνει το γέμισμα του τρέχον πολυγώνου ή να γίνει καθαρισμός την οθόνης, διαλέγοντας την επιλογή «Clear Screen».

Τέλος, η άσκηση δουλεύει και όταν αλλάζουμε το μέγεθος του παραθύρου (resize, fullscreen). Απλά, κάθε φορά που ο χρήστης αλλάζει το μέγεθος του παραθύρου, γίνεται καθαρισμός της οθόνης για να γίνει συγχρονισμός των καινούργιων συντεταγμένων του παραθύρου.

A screenshot of a computer

Description automatically generated with medium confidence

# ΒΙΒΛΙΟΓΡΑΦΙΑ

1. «https://en.wikipedia.org/wiki/Ellipse», τελευταία προσπέλαση 10.01.2022
2. «https://dai.fmph.uniba.sk/upload/0/01/Ellipse.pdf», τελευταία προσπέλαση 10.01.2022
3. Jim X. Chen, Chunyang Chen, «Foundations of 3D Graphics Programming: Using JOGL and Java3D»
4. «https://www.geeksforgeeks.org/scan-line-polygon-filling-using-opengl-c/», τελευταία προσπέλαση 12.01.2022

# ΠΑΡΑΡΤΗΜΑ

## Κώδικας Ερώτησης 2

Ακολουθεί ο κώδικας για το ερώτημα 2. Όνομα Αρχείου «bresenham\_exercise.c»

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* Ergasia 1 – Askhsh 2 – 10.01.2022

 \* Choropanitis Paschalis - 2453

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <GL/glut.h>

#include <stdio.h>

#include <string.h>

#include <math.h>

#define WIDTH 800

#define HEIGHT 600

// comment the line below for switching antialiasing off

#define ANTIALIASING\_ON

enum *MENU\_TYPE* {

    LINE,

    ELLIPSE,

    CLEAR\_SCREEN,

    QUIT,

};

enum *COLOR\_TYPE* {

    RED\_COLOR,

    GREEN\_COLOR,

    BLUE\_COLOR,

};

int winID; // windows ID for destroying window

/\* odd is the variable that measures the clicks of the user,

  depending at the mode selected by the menu,

  and do stuff accordingly in the mouse function \*/

int odd = 1, mode = 1;

int width = WIDTH;

int height = HEIGHT;

// x, y mouse coordinates of the clicked locations

int xm, ym, xm1, ym1, xm2, ym2;

int a, b;

float red = 1.0, green = 0.0, blue = 0.0;

// draw picel with intensity by its distance to the line

void draw\_pixel\_antialiased(int *x*, int *y*, float *R*, float *G*, float *B*, float *A*) {

    float r, g, b;

    r = (*R* \* (1 - *A* / 1.5));

    g = (*G* \* (1 - *A* / 1.5));

    b = (*B* \* (1 - *A* / 1.5));

    glBegin(GL\_POINTS);

    glColor3f(r, g, b);

    glVertex2i(*x*, *y*);

    glEnd();

}

void draw\_pixel(int *x*, int *y*, float *R*, float *G*, float *B*, float *A*) {

    glBegin(GL\_POINTS);

    glColor3f(*A*\**R*, *A*\**G*, *A*\**B*);

    glVertex2i(*x*, *y*);

    glEnd();

}

void draw\_line(int *x1*, int *x2*, int *y1*, int *y2*) {

    int i, p;

    int stepx, stepy, inc1, inc2;

    float d = 0, sin\_a, cos\_a, sin\_cos\_a, denom;

    int x = *x1*;

    int y = *y1*;

    int dx = *x2*-*x1*;

    int dy = *y2*-*y1*;

    if (dy < 0) {

        dy = -dy;

        stepy = -1;

    } else stepy = 1;

    if (dx < 0) {

        dx = -dx;

        stepx = -1;

    } else stepx = 1;

    if (dx > dy) {

        draw\_pixel(x, y, red, green, blue, 1.0); // start pixel

        p = 2\*dy - dx; // initial value of the decision parameter

        inc1 = 2\*(dy - dx);

        inc2 = 2\*dy;

        denom = (float) sqrt((double) (dx \* dx + dy \* dy));

        sin\_a = dy / denom;

        cos\_a = dx / denom;

        sin\_cos\_a = sin\_a - cos\_a;

        for (i=0; i < dx; i++) {

            if (p >= 0) {

                y += stepy;

                p += inc1;

                d += sin\_cos\_a;

            }

            else {

                p += inc2;

                d += sin\_a;

            }

            x += stepx;

            #ifdef ANTIALIASING\_ON

            draw\_pixel\_antialiased(x, y, red, green, blue, fabs(d));

            draw\_pixel\_antialiased(x, y + 1, red, green, blue, fabs(d - cos\_a));

            draw\_pixel\_antialiased(x, y - 1, red, green, blue, fabs(d + cos\_a));

            #else

            draw\_pixel(x, y, red, green, blue, 1.0);

            #endif

        }

    } else {

        draw\_pixel(x, y, red, green, blue, 1.0); // start pixel

        p = 2\*dx - dy;

        inc1 = 2\*(dx - dy);

        inc2 = 2\*dx;

        denom = (float) sqrt((double) (dx \* dx + dy \* dy));

        sin\_a = dx / denom;

        cos\_a = dy / denom;

        sin\_cos\_a = sin\_a - cos\_a;

        for (i=0; i < dy; i++) {

            if (p >= 0) {

                x += stepx;

                p += inc1;

                d += sin\_cos\_a;

            }

            else {

                p += inc2;

                d += sin\_a;

            }

            y += stepy;

            #ifdef ANTIALIASING\_ON

            draw\_pixel\_antialiased(x, y, red, green, blue, fabs(d));

            draw\_pixel\_antialiased(x + 1, y, red, green, blue, fabs(d - cos\_a));

            draw\_pixel\_antialiased(x - 1, y, red, green, blue, fabs(d + cos\_a));

            #else

            draw\_pixel(x, y, red, green, blue, 1.0);

            #endif

        }

    }

}

void draw\_ellipse(int *cx*, int *cy*, int *xradius*, int *yradius*) {

    int x, y;

    int xchange, ychange;

    int error;

    int twoasquare, twobsquare;

    int stopx, stopy;

    twoasquare = 2\**xradius*\**xradius*;

    twobsquare = 2\**yradius*\**yradius*;

    x = *xradius*;

    y = 0;

    xchange = *yradius*\**yradius*\*(1 - 2\**xradius*);

    ychange = *xradius*\**xradius*;

    error = 0;

    stopx = twobsquare\**xradius*;

    stopy = 0;

    while (stopx >= stopy) { // first set of points, y' > -1

        draw\_pixel(*cx*+x, *cy*+y, red, green, blue, 1.0);

        draw\_pixel(*cx*-x, *cy*+y, red, green, blue, 1.0);

        draw\_pixel(*cx*-x, *cy*-y, red, green, blue, 1.0);

        draw\_pixel(*cx*+x, *cy*-y, red, green, blue, 1.0);

        y++;

        stopy = stopy + twoasquare;

        error = error + ychange;

        ychange = ychange + twoasquare;

        if ((2\*error + xchange) > 0) {

            x--;

            stopx = stopx - twobsquare;

            error = error + xchange;

            xchange = xchange + twobsquare;

        }

    }

    y = *yradius*;

    x = 0;

    ychange = *xradius*\**xradius*\*(1 - 2\**yradius*);

    xchange = *yradius*\**yradius*;

    error = 0;

    stopy = twoasquare\**yradius*;

    stopx = 0;

    while (stopy >= stopx) { // second set of points, y' < -1

        draw\_pixel(*cx*+x, *cy*+y, red, green, blue, 1.0);

        draw\_pixel(*cx*-x, *cy*+y, red, green, blue, 1.0);

        draw\_pixel(*cx*-x, *cy*-y, red, green, blue, 1.0);

        draw\_pixel(*cx*+x, *cy*-y, red, green, blue, 1.0);

        x++;

        stopx = stopx + twobsquare;

        error = error + xchange;

        xchange = xchange + twobsquare;

        if ((2\*error + ychange) > 0) {

            y--;

            stopy = stopy - twoasquare;

            error = error + ychange;

            ychange = ychange + twoasquare;

        }

    }

}

void \*font = GLUT\_BITMAP\_TIMES\_ROMAN\_10;

void show(int *x*, int *y*, char \**string*) {

    glColor3f(1.0, 1.0, 1.0);

    int len, i;

    glRasterPos2f(*x*, *y*);

    len = (int) strlen(*string*);

    for (i = 0; i < len; i++)

        glutBitmapCharacter(font, *string*[i]);

}

// reset the display and draw the axis

void resetDisplay() {

    odd = 1;

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glLoadIdentity();

    glViewport(0, 0, width, height);

    gluOrtho2D(-(width/2), (width/2), -(height/2), (height/2));

    // draw axis

    for (int i =- (width/2); i < (width/2); i=i+10)

    {

        draw\_pixel(i, 0, 1.0, 1.0, 1.0, 1.0);

        draw\_pixel(0, i, 1.0, 1.0, 1.0, 1.0);

    }

    show(-(width/2),0,"- X-AXIS");

    show((width/2)-50,0,"X-AXIS");

    show(0,(height/2)-10,"Y-AXIS");

    show(0,-(height/2)+5,"- Y-AXIS");

    glutSwapBuffers();

    glFlush();

}

// update width and height values when on reshape

void myReshape(int *w*, int *h*) {

    width = *w*;

    height = *h*;

    resetDisplay();

}

void myDisplay() {

}

void keyEvent(unsigned char *key*, int *x*, int *y*) {

    switch (*key*) {

    // Quit

    case 'Q':

    case 'q':

        glutDestroyWindow(winID);

        break;

    default:

        break;

    }

}

void myMouse(int *button*, int *state*, *GLint* *x*, *GLint* *y*) {

    char coords[14];

    if(*button*==GLUT\_LEFT\_BUTTON && *state*==GLUT\_DOWN) {

        if (mode == 1) {

            // first mode is the line

            // if we are at the second click of the user we draw the line

            if (odd % 2 == 0) {

                xm1 = *x* - (width/2);

                ym1 = (height/2) - *y*;

                sprintf(coords, "X=%d,Y=%d", xm1, ym1);

                show(xm1+5, ym1, coords);

                draw\_pixel(xm, ym, red, green, blue, 1.0);

                draw\_line(xm, xm1, ym, ym1);

                glFlush();

            }

            else {

                xm = *x* - (width/2);

                ym = (height/2) - *y*;

                sprintf(coords, "X=%d,Y=%d", xm, ym);

                show(xm+5, ym, coords);

                draw\_pixel(xm, ym, red, green, blue, 1.0);

                glFlush();

            }

            odd++;

        }

        else {

            // second mode is the ellipse

            // if we are at the third click of the user we draw the ellipse

            if (odd == 3) {

                // get b location (y-axis of the ellipse)

                xm2 = *x* - (width/2);

                ym2 = (height/2) - *y*;

                // calculate b disance from the center

                b = sqrt(pow(xm2 - xm, 2) + pow(ym2 - ym, 2) \* 1.0);

                sprintf(coords, "B=%d", b);

                show(xm2+5, ym2, coords);

                draw\_pixel(xm2, ym2, red, green, blue, 1.0);

                draw\_ellipse(xm, ym, a, b);

                glFlush();

                odd = 1;

                return;

            }

            else if (odd == 2) {

                // get a location (x-axis of the ellipse)

                xm1 = *x* - (width/2);

                ym1 = (height/2) - *y*;

                // calculate a disance from the center

                a = sqrt(pow(xm1 - xm, 2) + pow(ym1 - ym, 2) \* 1.0);

                sprintf(coords, "A=%d", a);

                show(xm1+5, ym1, coords);

                draw\_pixel(xm1, ym1, red, green, blue, 1.0);

                glFlush();

            }

            else {

                // get the center of the eclipse

                xm = *x* - (width/2);

                ym = (height/2) - *y*;

                sprintf(coords, "X=%d,Y=%d", xm, ym);

                show(xm+5, ym, coords);

                draw\_pixel(xm, ym, red, green, blue, 1.0);

                glFlush();

            }

            odd++;

        }

    }

    else if (*button* == GLUT\_MIDDLE\_BUTTON && *state* == GLUT\_DOWN) {

        resetDisplay();

    }

}

void mySubMenu(int *selection*) {

    switch (*selection*) {

        case RED\_COLOR:

            red = 1.0;

            green = 0.0;

            blue = 0.0;

            break;

        case GREEN\_COLOR:

            red = 0.0;

            green = 1.0;

            blue = 0.0;

            break;

        case BLUE\_COLOR:

            red = 0.0;

            green = 0.0;

            blue = 1.0;

            break;

        default:

            printf("Invalid menu selection\n");

            break;

    }

}

void myMenu(int *selection*) {

    switch(*selection*) {

        case LINE: // drawing  line

            mode = 1;

            odd = 1;

            break;

        case ELLIPSE: // drawing ellipse

            mode = 2;

            odd = 1;

            break;

        case CLEAR\_SCREEN:

            resetDisplay();

            break;

        case QUIT:

            glutDestroyWindow(winID);

            break;

        default:

            printf("Invalid menu selection\n");

            break;

    }

}

void createMenu() {

    int lineColorSubmenu;

    // Create a two-level pop-up menu

    lineColorSubmenu = glutCreateMenu(mySubMenu);

    glutAddMenuEntry("Red", RED\_COLOR);

    glutAddMenuEntry("Green", GREEN\_COLOR);

    glutAddMenuEntry("Blue", BLUE\_COLOR);

    // menu creation

    glutCreateMenu(myMenu);

    glutAddMenuEntry("Line", LINE);

    glutAddMenuEntry("Ellipse", ELLIPSE);

    glutAddSubMenu("Color", lineColorSubmenu);

    glutAddMenuEntry("Clear Screen", CLEAR\_SCREEN);

    glutAddMenuEntry("Quit", QUIT);

    // register menu to right mouse button

    glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

void init() {

    glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

    glutInitWindowPosition(100, 100);

    glutInitWindowSize(width, height);

    winID = glutCreateWindow("Bresenham Exercise");

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glMatrixMode(GL\_PROJECTION);

}

int main(int *argc*, char \*\**argv*) {

    glutInit(&*argc*, *argv*);

    init();

    createMenu();

    glutDisplayFunc(myDisplay);

    glutReshapeFunc(myReshape);

    glutMouseFunc(myMouse);

    glutKeyboardFunc(keyEvent);

    glutMainLoop();

    return 0;

}

## Κώδικας Ερώτησης 4

Ακολουθεί ο κώδικας για το ερώτημα 4. Όνομα Αρχείου «scan\_polygon\_exercise.c»

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* Ergasia 1 – Askhsh 4 – 12.01.2022

 \* Choropanitis Paschalis - 2453

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <GL/glut.h>

#include <stdio.h>

#include <string.h>

#include <math.h>

#define WIDTH 800

#define HEIGHT 600

#define MAXEDGES 10

#define MAXVER 10000

enum *MENU\_TYPE* {

    POLYGON,

    FILL,

    CLEAR\_SCREEN,

    QUIT,

};

enum *COLOR\_TYPE* {

    RED\_COLOR,

    GREEN\_COLOR,

    BLUE\_COLOR,

};

typedef struct *edgelist* {

    int ymax;       // max y-coord of edge

    float currentx; // x-coord of lowest edge point

    float slopeinverse;

} *EdgeList*;

typedef struct *edgetabletup* {

    int countEdgeList; // count of edgelists

*EdgeList* lists[MAXVER];

} *EdgeTableTuple*;

*EdgeTableTuple* \*EdgeTable, ActiveEdgeTuple;

typedef struct *edgecoords* {

    int x;

    int y;

} *EdgeCoord*;

*EdgeCoord* EdgeCoordinates[MAXEDGES];

int winID; // windows ID for destroying window

/\* odd is the variable that measures the clicks of the user,

  depending at the mode selected by the menu,

  and do stuff accordingly in the mouse function \*/

int odd = 1, mode = 0;

int width = WIDTH;

int height = HEIGHT;

// x, y mouse coordinates of the clicked locations

int xm, ym;

float red = 1.0, green = 0.0, blue = 0.0;

void draw\_pixel(int *x*, int *y*, float *R*, float *G*, float *B*, float *e*) {

    glBegin(GL\_POINTS);

    glColor3f(*e*\**R*, *e*\**G*, *e*\**B*);

    glVertex2i(*x*, *y*);

    glEnd();

}

void initEdgeTable() {

    for (int i = 0; i < height; i++) {

        EdgeTable[i].countEdgeList = 0;

    }

    ActiveEdgeTuple.countEdgeList = 0;

}

// Function to sort an array using insertion sort

void insertionSort(*EdgeTableTuple* \**ett*) {

    int i,j;

*EdgeList* temp;

    for (i = 1; i < *ett*->countEdgeList; i++)

    {

        temp.ymax = *ett*->lists[i].ymax;

        temp.currentx = *ett*->lists[i].currentx;

        temp.slopeinverse = *ett*->lists[i].slopeinverse;

        j = i - 1;

        while ((temp.currentx < *ett*->lists[j].currentx) && (j >= 0))

        {

*ett*->lists[j + 1].ymax = *ett*->lists[j].ymax;

*ett*->lists[j + 1].currentx = *ett*->lists[j].currentx;

*ett*->lists[j + 1].slopeinverse = *ett*->lists[j].slopeinverse;

            j = j - 1;

        }

*ett*->lists[j + 1].ymax = temp.ymax;

*ett*->lists[j + 1].currentx = temp.currentx;

*ett*->lists[j + 1].slopeinverse = temp.slopeinverse;

    }

}

void storeEdgeInTuple (*EdgeTableTuple* \**receiver*,int *y*,int *x*,float *slopInv*) {

    // both used for edgetable and active edge table..

    // The edge tuple sorted in increasing ymax and x of the lower end.

    (*receiver*->lists[(*receiver*)->countEdgeList]).ymax = *y*;

    (*receiver*->lists[(*receiver*)->countEdgeList]).currentx = (float)*x*;

    (*receiver*->lists[(*receiver*)->countEdgeList]).slopeinverse = *slopInv*;

    // sort the buckets

    insertionSort(*receiver*);

    (*receiver*->countEdgeList)++;

}

void storeEdgeInTable (int *x1*, int *y1*, int *x2*, int *y2*) {

    float m, minv;

    int ymaxTS, xwithyminTS, scanline; //ts stands for to store

    if (*x2* == *x1*) {

        minv=0.000000;

    }

    else {

        m = ((float)(*y2* - *y1*)) / ((float)(*x2* - *x1*));

        // horizontal lines are not stored in edge table

        if (*y2* == *y1*)

            return;

        minv = (float)1.0 / m;

    }

    if (*y1* > *y2*) {

        scanline = *y2*;

        ymaxTS = *y1*;

        xwithyminTS = *x2*;

    }

    else {

        scanline = *y1*;

        ymaxTS = *y2*;

        xwithyminTS = *x1*;

    }

    // the assignment part is done..now storage..

    storeEdgeInTuple(&EdgeTable[scanline],ymaxTS,xwithyminTS,minv);

}

void removeEdgeByYmax(*EdgeTableTuple* \**Tup*, int *yy*) {

    for (int i = 0; i < *Tup*->countEdgeList; i++) {

        if (*Tup*->lists[i].ymax == *yy*) {

            for (int j = i; j < *Tup*->countEdgeList - 1; j++) {

*Tup*->lists[j].ymax =*Tup*->lists[j+1].ymax;

*Tup*->lists[j].currentx =*Tup*->lists[j+1].currentx;

*Tup*->lists[j].slopeinverse = *Tup*->lists[j+1].slopeinverse;

            }

*Tup*->countEdgeList--;

            i--;

        }

    }

}

void updatexbyslopeinv(*EdgeTableTuple* \**Tup*) {

    for (int i=0; i < *Tup*->countEdgeList; i++) {

        (*Tup*->lists[i]).currentx =(*Tup*->lists[i]).currentx + (*Tup*->lists[i]).slopeinverse;

    }

}

void scanlineFill() {

    int j, x1, ymax1, x2, ymax2, FillFlag = 0, coordCount;

    // we will start from scanline 0;

    // Repeat until last scanline:

    for (int i = 0; i < height; i++) { // Increment y by 1 (next scan line)

        // Move from ET bucket y to the

        // AET those edges whose ymin = y (entering edges)

        for (j=0; j<EdgeTable[i].countEdgeList; j++)

        {

            storeEdgeInTuple(&ActiveEdgeTuple,EdgeTable[i].lists[j].

                     ymax,EdgeTable[i].lists[j].currentx,

                    EdgeTable[i].lists[j].slopeinverse);

        }

        // Remove from AET those edges for

        // which y=ymax (not involved in next scan line)

        removeEdgeByYmax(&ActiveEdgeTuple, i);

        //sort AET (remember: ET is presorted)

        insertionSort(&ActiveEdgeTuple);

        // Fill lines on scan line y by using pairs of x-coords from AET

        j = 0;

        FillFlag = 0;

        coordCount = 0;

        x1 = 0;

        x2 = 0;

        ymax1 = 0;

        ymax2 = 0;

        while (j < ActiveEdgeTuple.countEdgeList) {

            if (coordCount%2==0) {

                x1 = (int)(ActiveEdgeTuple.lists[j].currentx);

                ymax1 = ActiveEdgeTuple.lists[j].ymax;

                if (x1==x2) {

                /\* three cases can arrive-

                    1. lines are towards top of the intersection

                    2. lines are towards bottom

                    3. one line is towards top and other is towards bottom

                \*/

                    if (((x1 == ymax1) && (x2 != ymax2))||

                        ((x1 != ymax1) && (x2 == ymax2))) {

                        x2 = x1;

                        ymax2 = ymax1;

                    }

                    else {

                        coordCount++;

                    }

                }

                else {

                    coordCount++;

                }

            }

            else {

                x2 = (int)ActiveEdgeTuple.lists[j].currentx;

                ymax2 = ActiveEdgeTuple.lists[j].ymax;

                FillFlag = 0;

                // checking for intersection...

                if (x1 == x2) {

                /\*three cases can arrive-

                    1. lines are towards top of the intersection

                    2. lines are towards bottom

                    3. one line is towards top and other is towards bottom

                \*/

                    if (((x1 == ymax1) && (x2 != ymax2))||

                        ((x1 != ymax1) && (x2 == ymax2))) {

                        x1 = x2;

                        ymax1 = ymax2;

                    }

                    else {

                        coordCount++;

                        FillFlag = 1;

                    }

                }

                else {

                    coordCount++;

                    FillFlag = 1;

                }

                if(FillFlag)

                {

                    //drawing actual lines...

                    glColor3f(red, green, blue);

                    glBegin(GL\_LINES);

                    glVertex2i(x1,i);

                    glVertex2i(x2,i);

                    glEnd();

                    glFlush();

                }

            }

            j++;

        }

        // For each nonvertical edge remaining in AET, update x for new y

        updatexbyslopeinv(&ActiveEdgeTuple);

    }

    printf("\nScanline filling complete\n");

    free(EdgeTable);

}

void drawPolygon() {

    EdgeTable = (*EdgeTableTuple* \*) malloc (height \* sizeof(*EdgeTableTuple*));

    initEdgeTable();

    glColor3f(red, green, blue);

    int count = 0, x0, y0, x1, y1, x2, y2;

    for (int i = 0; i < odd; i++) {

        if (i == 0) {

            x0 = EdgeCoordinates[i].x;

            y0 = EdgeCoordinates[i].y;

        }

        else if (i == odd - 1) {

            x1 = x2;

            y1 = y2;

            x2 = x0;

            y2 = y0;

            storeEdgeInTable(x1, y1, x2, y2); //storage of edges in edge table.

            break;

        }

        if (count > 1) {

            x1 = x2;

            y1 = y2;

            count = 1;

        }

        if (count == 0) {

            x1 = EdgeCoordinates[i].x;

            y1 = EdgeCoordinates[i].y;

        }

        else {

            x2 = EdgeCoordinates[i].x;

            y2 = EdgeCoordinates[i].y;

            glBegin(GL\_LINES);

            glVertex2i(x1, y1);

            glVertex2i(x2, y2);

            glEnd();

            if (i == odd - 2) {

                glBegin(GL\_LINES);

                glVertex2i(x2, y2);

                glVertex2i(x0, y0);

                glEnd();

            }

            storeEdgeInTable(x1, y1, x2, y2); //storage of edges in edge table.

            glFlush();

        }

        count++;

    }

}

void \*font = GLUT\_BITMAP\_TIMES\_ROMAN\_10;

void show(int *x*, int *y*, char \**string*) {

    glColor3f(1.0, 1.0, 1.0);

    int len, i;

    glRasterPos2f(*x*, *y*);

    len = (int) strlen(*string*);

    for (i = 0; i < len; i++)

        glutBitmapCharacter(font, *string*[i]);

}

// reset the display and draw the axis

void resetDisplay() {

    odd = 1;

    mode = 0;

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glLoadIdentity();

    glViewport(0, 0, width, height);

    gluOrtho2D(0, width, 0, height);

    // draw axis

    for (int i = 0; i < width; i=i+10)

    {

        draw\_pixel(i, height/2, 1.0, 1.0, 1.0, 1.0);

        draw\_pixel(width/2, i, 1.0, 1.0, 1.0, 1.0);

    }

    show(0,(height/2),"- X-AXIS");

    show(width-40,(height/2),"X-AXIS");

    show((width/2),height - 10,"Y-AXIS");

    show((width/2),0,"- Y-AXIS");

    glutSwapBuffers();

    glFlush();

}

// update width and height values when on reshape

void myReshape(int *w*, int *h*) {

    width = *w*;

    height = *h*;

    resetDisplay();

}

void myDisplay() {

}

void keyEvent(unsigned char *key*, int *x*, int *y*) {

    switch (*key*) {

    // Quit

    case 'Q':

    case 'q':

        EdgeTable = NULL;

        free(EdgeTable);

        glutDestroyWindow(winID);

        break;

    default:

        break;

    }

}

void myMouse(int *button*, int *state*, *GLint* *x*, *GLint* *y*) {

    char coords[14];

    if(*button*==GLUT\_LEFT\_BUTTON && *state*==GLUT\_DOWN) {

        if (mode == 0) {

            if (odd <= MAXEDGES) {

                xm = *x*;

                ym = height - *y*;

                sprintf(coords, "X=%d,Y=%d", xm - (width/2), ym - (height/2));

                show(xm+5, ym, coords);

                draw\_pixel(xm, ym, red, green, blue, 1.0);

                glFlush();

                EdgeCoordinates[odd - 1].x = xm;

                EdgeCoordinates[odd - 1].y = ym;

                odd++;

            }

        }

    }

    else if (*button* == GLUT\_MIDDLE\_BUTTON && *state* == GLUT\_DOWN) {

        resetDisplay();

    }

}

void mySubMenu(int *selection*) {

    switch (*selection*) {

        case RED\_COLOR:

            red = 1.0;

            green = 0.0;

            blue = 0.0;

            break;

        case GREEN\_COLOR:

            red = 0.0;

            green = 1.0;

            blue = 0.0;

            break;

        case BLUE\_COLOR:

            red = 0.0;

            green = 0.0;

            blue = 1.0;

            break;

        default:

            printf("Invalid menu selection\n");

            break;

    }

}

void myMenu(int *selection*) {

    switch(*selection*) {

        case POLYGON: // draw polygon if there are at least 3 edges

            if (odd > 3) {

                drawPolygon();

                mode = 1;

            }

            break;

        case FILL: // fill polygon if there are at least 3 edges drawed

            if (odd > 3 && mode == 1) {

                scanlineFill(); // actual calling of scanline filling..

                odd = 1;

                mode = 0;

            }

            break;

        case CLEAR\_SCREEN:

            resetDisplay();

            break;

        case QUIT:

            EdgeTable = NULL;

            free(EdgeTable);

            glutDestroyWindow(winID);

            break;

        default:

            printf("Invalid menu selection\n");

            break;

    }

}

void createMenu() {

    int lineColorSubmenu;

    // Create a two-level pop-up menu

    lineColorSubmenu = glutCreateMenu(mySubMenu);

    glutAddMenuEntry("Red", RED\_COLOR);

    glutAddMenuEntry("Green", GREEN\_COLOR);

    glutAddMenuEntry("Blue", BLUE\_COLOR);

    // menu creation

    glutCreateMenu(myMenu);

    glutAddMenuEntry("Draw Polygon", POLYGON);

    glutAddMenuEntry("Fill Polygon", FILL);

    glutAddSubMenu("Color", lineColorSubmenu);

    glutAddMenuEntry("Clear Screen", CLEAR\_SCREEN);

    glutAddMenuEntry("Quit", QUIT);

    // register menu to right mouse button

    glutAttachMenu(GLUT\_RIGHT\_BUTTON);

}

void init() {

    glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

    glutInitWindowPosition(100, 100);

    glutInitWindowSize(width, height);

    winID = glutCreateWindow("Polygon Exercise");

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glClearColor(0.0, 0.0, 0.0, 1.0);

    glMatrixMode(GL\_PROJECTION);

}

int main(int *argc*, char \*\**argv*) {

    glutInit(&*argc*, *argv*);

    init();

    createMenu();

    glutDisplayFunc(myDisplay);

    glutReshapeFunc(myReshape);

    glutMouseFunc(myMouse);

    glutKeyboardFunc(keyEvent);

    glutMainLoop();

    return 0;

}