ECE433: ΓΡΑΦΙΚΑ ΗΥ

Χειμερινό Εξάμηνο 2021-2022

Εργασία 2

Ομάδα φοιτητών #39

Πολυκάρπου Χριστόφορος - 2668

Πουλιάνου Χάιδω - 2613

## Άσκηση 1

1. Αρχικοποίησε τις μεταβλητές u1=0 και u2=1 και r1, r3 = 0, r2, r4 = 1.
2. Υπολόγισε το Δx = x2 – x1, Δy = y2- y1, q1 = x1 – xmin, q2 = xmax – x1, q3 = y1 – ymin, q4 = ymax – y1.
3. Αν Δx == 0 και q1 ή q2 ή q3 ή q4 < 0:
   1. Επιστρέφει, χωρίς να σχεδιάσει γραμμή
4. Αλλιώς,
   1. Αν Δx <0
      1. r1 = q2/Δx
      2. r2 = q1/Δx
   2. Αλλιώς
      1. r1 = q1/Δx
      2. r2 = q2/Δx
   3. Αν Δy < 0
      1. r3 = q4/Δy
      2. r4 = q3/Δy
   4. Αλλιώς
      1. r3 = q3/Δy
      2. r4 = q4/Δy
5. Θέσε u1 = max(r1, r3) και u2 = min(r2, r4)
6. Αν u1 > u2,
   1. Επιστρέφει, χωρίς να σχεδιάσει γραμμή
7. Αλλιώς, υπολόγισε τα νέα σημεία του τμήματος και σχεδίασέ το:
   1. x1 = x1 + Δx \* u1, yn1 = y1 + Δy \* u1
   2. x2 = x1 + Δx \* u2, y2 = y1 + Δy \* u2
8. Σχεδίασε το νέο ευθύγραμμο clipped τμήμα με P1(x1, y1)->P2(x2, y2).

## Άσκηση 2

1. Θέσε τα region codes c0, c1 για το P1 και P2 αντίστοιχα. Επίσης Δx = x2- x1, Δy = y2 – y1

και yinter1 = y1, xinter1 = x1, yinter2 = y2, xinter2 = x2 τα σημεία τομής.

1. Αν OR(c0,c1) == 0000, σχεδίασε το τμήμα P1->P2 χωρίς αλλαγή και επέστρεψε (trivial accept)
2. Αλλιώς
   1. Αν AND(c0, c1) != 0000, επέστρεψε χωρίς να σχεδιάσεις (trivial reject)
   2. Αλλιώς,
      1. Αν x1 < xmin
         1. Yinter1 = y1 + Δy/Δx \* (xmin – x1)
         2. Xinter1 = xmin
      2. Αν y1 < ymin
         1. Yinter1 = ymin
         2. Xinter1 = x1 + (ymin – y1)/(Δy/Δx)
      3. Αν x2 > xmax
         1. Yinter2 = y1 + Δy/Δx \* (xmax – x1)
         2. Xinter2 = xmax
      4. Αν y2 > ymax
         1. Yinter2 = ymax
         2. Xinter2 = x1 + (ymax – y1)/(Δy/Δx)
3. Σχεδίασε το νέο clipped ευθύγραμμο τμήμα P1(xinter1, yinter1)->P2(xinter2, yinter2).

## Άσκηση 3

Αρχικοποίηση των region codes

C0 = 0101

C1 = 1010

Και m = Δy/Δx = 1.

Ελέγχω αρχικά αν είναι όλο μέσα:

OR(C0, C1) != 0000, άρα δεν είναι μέσα.

Ελέγχω αν είναι όλο έξω:

AND(C0, C1) == 0000, άρα αναλύω και βρίσκω τα σημεία τομής.

Βήμα 1:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο C-S  Έλεγχος με αριστερή πλευρά.  X1 < xmin=>  Yinter1 = y1 + m\*(xmin – x1) = 0.5  Xinter1 = xmin = 1  P0 (1, 0.5) |

Βήμα 2:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο C-S  Έλεγχος κάτω πλευράς.  Y1 < ymin=>  Xinter1 = x1 + (ymin – y1)/ m = 1.5  Yinter1 = ymin = 1  P0(1.5, 1) |

Βήμα 3:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο C-S  Έλεγχος δεξιάς πλευράς.  X2 > xmax=>  yinter2 = y1 + m\*(xmax – x1) = 4.5  xinter2 = xmax = 5  P1(5, 4.5) |

Βήμα 4:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο C-S  Έλεγχος κάτω πλευράς.  Y2 > ymax=>  Xinter2 = x1 + (ymax – y1)/ m = 4.5  Yinter2 = ymax = 4  P1(4.5, 4) |

Βήμα 5:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο C-S  Τέλος πλευρών.  Σχεδίαση clipped τμήματος  P0(1.5, 1)->P1(4.5, 4) |

## Άσκηση 4

Αρχικοποίηση των u1 = 0, u2 = 1,

p1 = -(x2 – x1) = -(5.5 – 0.5 )=-5

p2 = x2 – x1 = 5

p3 = -(y2 – y1) = -(5 - 0 )= -5

p4 = y2 – y1 = 5

q1 = x1 - xmin = 0.5 – 1 = -0.5

q2 = xmax – x1 = 4.5

q3 = y1 - ymin = -1

q4 = ymax – y1 = 4

Βήμα 1:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο L-B  Για την αριστερή πλευρά:  p1 <0, άρα εισέρχεται  r1 = q1/p1 = 0.1  U1 = max (0, r1, u1) = max(0, 0.1, 0) = 0.1  Υπολόγισε το σημείο τομής:  X1’ = u1\*Δx + x1 = 0.1\*5 + 0.5 = 1  Y1’ = u1\*Δy + y1 = 0.1\*5 = 0.5  t1(1, 0.5)  Σχεδίαση νέου ευθ. τμήματος, από t1->P2  u1<u2, συνεχίζουμε |

Βήμα 2:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο L-B  Για τη δεξιά πλευρά:  p2 > 0, άρα εξέρχεται  r2 = q2/p2 = 0. 9  U2 = min (1, r2, u2) = min(1, 0. 9, 1) = 0.9  Υπολόγισε το σημείο τομής:  X2’ = u2\*Δx + x1 = 0.9\*5 + 0.5 = 5  Y2’ = u2\*Δy + y1 = 0.9\*5 = 4.5  t2(5, 4.5)  Σχεδίαση νέου ευθ. τμήματος, από t1->t2  u1<u2, συνεχίζουμε |

Βήμα 3:



|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο L-B  Για την κάτω πλευρά:  p3 < 0, άρα εισέρχεται  r1 = q3/p3 = 0.2  U1 = max (0, r1, u1) = max(0, 0.2, 0.1) = 0.2  Υπολόγισε το σημείο τομής:  X1’ = u1\*Δx + x1 = 0.2\*5 + 0.5 = 1.5  Y1’ = u1\*Δy + y1 = 0.2\*5 = 1  t1(1.5 , 1)  Σχεδίαση νέου ευθ. τμήματος, από t1->t2  u1<u2, συνεχίζουμε |

Βήμα 4:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο L-B  Για την πάνω πλευρά:  p4 > 0, άρα εξέρχεται  r4 = q4/p4 = 0.8  U2 = min (1, r2, u2) = min(1, 0. 8, 0.9) = 0.8  Υπολόγισε το σημείο τομής:  X2’ = u2\*Δx + x1 = 0.8\*5 + 0.5 = 4.5  Y2’ = u2\*Δy + y1 = 0.8\*5 = 4  t2(4.5, 4)  Σχεδίαση νέου ευθ. τμήματος, από t1->t2  u1<u2, συνεχίζουμε, αλλά τελειώσαμε απτις πλευρές, οπότε τερματίζει  Νέο clipped ευθ. τμήμα το t1->t2 |

## Άσκηση 5

Χρήση 6 bit αντί για 4.

Το 1ο και 2ο bit δειχνουν αν βρισκεται πάνω ή κάτω από το ορθογώνιο παράθυρο

Το 3ο και 4ο αν βρίσκεται δεξιά ή αριστερά.

Το 5ο και 6ο αν βρίσκεται μπροστά ή πίσω.

1. Θέσε τα region codes c0, c1 για το P1 και P2 αντίστοιχα. Επίσης Δx = x2- x1, Δy = y2 – y1, Δz = z2- z1.

και yinter1 = y1, xinter1 = x1, zinter1 = z1, yinter2 = y2, xinter2 = x2, zinter2 = z2 τα σημεία τομής.

1. Αν OR(c0,c1) == 000000, σχεδίασε το τμήμα P1->P2 χωρίς αλλαγή και επέστρεψε (trivial accept)
2. Αλλιώς
   1. Αν AND(c0, c1) != 000000, επέστρεψε χωρίς να σχεδιάσεις (trivial reject)
   2. Αλλιώς για **προοπτική προβολή**,
      1. Αν y1 > ytopPlane
         1. Yinter1= Ytop = y1 + t \* (y2 – y1) => t = (Ytop – y1)/(y2 – y1)
         2. Xinter1 = x1 + t\*(x2 – x1) = x1 + (Ytop – y1)/(y2 – y1)\*(x2 – x1)
         3. Zinter1 = z1 + t\*(z2 – z1) = z1 + (Ytop – y1)/(y2 – y1)\*(z2 – z1)
      2. Αν y1 < ybottomPlane
         1. Yinter1= ybottomPlane = y1 + t \* (y2 – y1) => t = (ybottomPlane – y1)/(y2 – y1)
         2. Xinter1 = x1 + t\*(x2 – x1) = x1 + (ybottomPlane – y1)/(y2 – y1)\*(x2 – x1)
         3. Zinter1 = z1 + t\*(z2 – z1) = z1 + (ybottomPlane – y1)/(y2 – y1)\*(z2 – z1)
      3. Αν x1 > xrightPlane
         1. Xinter1= Xright = x1 + t \* (x2 – x1) => t = (Xright– x1)/(x2 – x1)
         2. Yinter1 = y1 + t\*(y2 – y1) = y1 + (Xright – x1)/(x2 – x1)\*(y2 – y1)
         3. Zinter1 = z1 + t\*(z2 – z1) = z1 + (Xright – x1)/(x2 – x1)\*(z2 – z1)
      4. Αν x1 < XleftPlane
         1. Xinter1= Xleft= x1 + t \* (x2 – x1) => t = (Xleft– x1)/(x2 – x1)
         2. Yinter1 = y1 + t\*(y2 – y1) = y1 + (Xleft– x1)/(x2 – x1)\*(y2 – y1)
         3. Zinter1 = z1 + t\*(z2 – z1) = z1 + (Xleft– x1)/(x2 – x1)\*(z2 – z1)
      5. Αν z1 > ZFrontPlane
         1. Zinter1= ZFront = z1 + t \* (z2 – z1) => t = (ZFront– z1)/(z2 – z1)
         2. Xinter1 = x1 + t\*(x2 – x1) = x1 + (ZFront– z1)/(z2 – z1)\*(x2 – x1)
         3. Yinter1 = y1 + t\*(y2 – y1) = y1 + (ZFront– z1)/(z2 – z1)\*(y2 – y1)
      6. Αν z1 < ZBackPlane
         1. Zinter1= ZBack= z1 + t \* (z2 – z1) => t = (ZBack– z1)/(z2 – z1)
         2. Xinter1 = x1 + t\*(x2 – x1) = x1 + (ZBack– z1)/(z2 – z1)\*(x2 – x1)
         3. Yinter1 = y1 + t\*(y2 – y1) = y1 + (ZBack– z1)/(z2 – z1)\*(y2 – y1)
3. Σχεδίασε το νέο clipped ευθύγραμμο τμήμα P1(xinter1, yinter1)->P2(xinter2, yinter2).

## Άσκηση 6

Το πρόγραμμα δίνει τη δυνατότητα στο χρήστη αρχικά να μαρκάρει τα σημεία των κορυφών του πολυγώνου. Με το πάτημα του χαρακτήρα ‘p’ ή ‘P’ καλείται η keyEvent() και σχεδιάζεται το πολύγωνο με βάση τις δοθείσες κορυφές. Ο χρήστης έπειτα μπορεί να πατήσει το F1 , όπου καλείται η keyCut(), και να μεταβεί στη κατάσταση σχεδιασμού ορθογωνίου αποκοπής, όπου επιλέγει ένα σημείο στην οθόνη και σέρνει και απελευθερώνει το ποντίκι μέχρι την απέναντι επιθυμητή γωνία του ορθογωνίου. Η όλη διαδικασία χρησιμοποιεί μια συνάρτηση mouseMove(), όπου ελέγχει τη κίνηση του ποντικού και σχεδιάζει παράλληλα το ορθογώνιο. Για την αποκοπή του πολυγώνου με βάση το δημιουργούμενο ορθογώνιο, ο χρήστης απαιτείται να πατήσει το χαρακτήρα ‘c’ ή ‘C’. Με αυτό καλείται η μέθοδος αποκοπής Sutherland-Hodgeman, όπου ελέγχει για κάθε πλευρά του ορθογωνίου και για όλες τις κορυφές του πολυγώνου αν έχω περίπτωση outside-inside, inside-inside και inside-outside. Ανάλογα, υπολογίζει τα νέα σημεία τομής και τα τοποθετεί σε έναν πίνακα temp\_buffer[], όπου χρησιμοποιεί για τον έλεγχο με την επόμενη σε σειρά πλευρά του ορθογωνίου. Αφού βρεθούν όλα τα νέα σημεία τομής με όλες τις πλευρές, καλείται η glutRedisplay() για να σχεδιάσει το νέο πολύγωνο με βάση αυτές τις νέες κορυφές.

Με το πάτημα του πλήκτρου space, εμφανίζεται ή εξαφανίζεται το ορθογώνιο αποκοπής. Τέλος, ο χρήστης μπορεί με το πάτημα του δεξιού πλήκτρου να καθαρίσει πλήρως την οθόνη και μετέπειτα με επιλογή του αριστερού να επαναεπιλέξει νέα σημεία-κορυφές και να επαναλάβει την όλη διαδικασία.

Οι δυσκολίες που συναντήσαμε ήταν στην υλοποίηση του αλγορίθμου αποκοπής, συγκεκριμένα στην εύρεση της εξίσωσης σημείων τομής και στον έλεγχο για όλες τις κορυφές. Τα εύκολα σημεία ήταν η δημιουργία των display(), myMouse() και keyEvent() functions, καθώς τα έχουμε ξαναδεί.

Το πρόγραμμα πήρε περίπου 2-3 μέρες να υλοποιηθεί.

## Άσκηση 7

Αρχικοποιώ τις συντεταγμένες του παραθύρου για σύγκριση:

Xleft = 1, xright = 5, ytop = 4, ybottom = 1.

m = dy/dx = (yE – yS)/(xE- xS)

Και τις λίστες που θα περιέχουν τις ακμές R(), B(), T() αρχικά κενές. Η αρχική λίστα L() = {F,E,D,C,B,A}.

Βήμα 1:



|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Ελέγχω για την αριστερή πλευρά. Κάνω pop από τη λίστα το Α και ελέγχω αν xA < xleft.  xA > xleft => S inside  ελέγχω για το επόμενο στη λίστα, αν xB < xleft:  xB < xleft => E outside  Άρα έχω περίπτωση inside-outside, οπότε πρέπει να βρω το σημείο τομής I1:  I1( xleft, yA -m\*(xleft - xA)) = (1, 0.5 + 1.3) =>  I1(1, 1.83)  Οπότε προσθέτω στον R(I1)={I1}  L(A) = {F,E,D,C,B} |

Βήμα 2:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop από τη λίστα το B.  ελέγχω αν xB < xleft:  xB < xleft => S outside  Ενώ για το C, που είναι το επόμενο στη λίστα:  xC > xleft=> E inside  Άρα έχω περίπτωση outside-inside, οπότε πρέπει να βρω το σημείο τομής I2 και να το επιστρέψω μαζί με το Ε:  I2( xleft, yΒ +m\*(xleft - xΒ)) = (1, 2.5 + 1\*0.5) =>  I2(1, 3)  Οπότε προσθέτω στον R(I1)={C, I2, I1}  L(A) = {F,E,D,C} |

Βήμα 3:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop από τη λίστα το C.  xC > xleft => S inside  Ενώ για το D, που είναι το επόμενο στη λίστα:  xD > xleft=> E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  R(I1)={D, C, I2, I1}  L(A) = {F,E,D} |

Βήμα 4:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop από τη λίστα το D.  xD > xleft => S inside  Ενώ για το E, που είναι το επόμενο στη λίστα:  xE > xleft=> E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  R(I1)={E, D, C, I2, I1}  L(A) = {F,E} |

Βήμα 3:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop από τη λίστα το E.  xE> xleft => S inside  Ενώ για το F, που είναι το επόμενο στη λίστα:  xF > xleft=> E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  R(I1)={F, E, D, C, I2, I1}  L(A) = {F } |

Βήμα 5:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop από τη λίστα το F.  xF > xleft => S inside  Ενώ για το A, που είναι το αρχικό κρατημένο στη λίστα:  xΑ > xleft=> E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  R(I1)={A, F, E, D, C, I2, I1}  L() = { } |

Βήμα 6:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop από τη λίστα R() πλέον το I1.  XI1 < xright => S inside  Ενώ για το I2, που είναι το επόμενο στη λίστα:  xI2 < xright => E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  B(I2)={ I2 }  R(I1) = { A, F, E, D, C, I2} |

Βήμα 7:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop το I2.  XI2 < xright => S inside  Ενώ για το C, που είναι το επόμενο στη λίστα:  xC < xright => E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  B(I2)={ C, I2 }  R(I1) = { A, F, E, D, C} |

Βήμα 8:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop C.  XC < xright => S inside  Ενώ για το D, που είναι το επόμενο στη λίστα:  xD > xright => E outside  Άρα έχω περίπτωση inside-outside, οπότε υπολογίζω το I3:  I3( xright, yC +m\*(xright - xC)) = (5, 4.5 - 0.5\*2.5)  =>I3(5 , 3.5)  B(I2)={ I3, C, I2 }  R(I1) = { A, F, E, D} |

Βήμα 9:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop D.  XD > xright => S outside  Ενώ για το E, που είναι το επόμενο στη λίστα:  xE < xright => E inside  Άρα έχω περίπτωση outside-inside, οπότε επιστρέφω το I4 και το E:  I4( xright, yD +m\*(xright - xD)) = (5, 3 - 2.5\*0.5)  =>I4(5 , 1.75)  B(I2)={E, I4, I3, C, I2 }  R(I1) = { A, F, E } |



Βήμα 10:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop E.  XE < xright => S inside  Ενώ για το F, που είναι το επόμενο στη λίστα:  xF < xright => E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  B(I2)={F, E, I4, I3, C, I2 }  R(I1) = { A, F } |

Βήμα 11:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop F.  XF < xright => S inside  Ενώ για το A, που είναι το επόμενο στη λίστα:  xA < xright => E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  B(I2)={A, F, E, I4, I3, C, I2 }  R(I1) = { A } |

Βήμα 12:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop A.  XA < xright => S inside  Ενώ για το I1:  xI1 < xright => E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  B(I2)={I1, A, F, E, I4, I3, C, I2 }  R(I1) = { } |

Βήμα 13:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop από την B() το Ι2.  yΙ2 > ybottom => S inside  Ενώ για το C:  yC > ybottom => E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  T(C) = {C}  B(I2)={ I1, A, F, E, I4, I3, C } |

Βήμα 14:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop C.  yC > ybottom => S inside  Ενώ για το F, που είναι το επόμενο στη λίστα:  yI3 > ybottom => E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  T(C) = {I3, C}  B(I2)={ I1, A, F, E, I4, I3 } |

Βήμα 15:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop I3.  YI3 > ybottom => S inside  Ενώ για το I4, που είναι το επόμενο στη λίστα:  YI4 > ybottom => E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  T(C) = {I4, I3, C}  B(I2)={ I1, A, F, E, I4 } |

Βήμα 16:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop I4.  YI4 > ybottom => S inside  Ενώ για το E, που είναι το επόμενο στη λίστα:  yE < ybottom => E outside  Άρα έχω περίπτωση inside-outside, οπότε επιστρέφω το I5:  I5(xI4 + (ybottom – yI4)/m, ybottom) =  =(5 + 0.75/-2.5, 1)=>I5(4.7, 1)  T(C) = {I5, I4, I3, C}  B(I2)={ I1, A, F, E } |

Βήμα 17:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop E.  yE < ybottom => S outside  Ενώ για το F, που είναι το επόμενο στη λίστα:  yF > ybottom => E inside  Άρα έχω περίπτωση outside-inside, οπότε επιστρέφω I6 και E:  I6(xE - (ybottom – yE)/m, ybottom) =  =(4.5 - 0.5/1.5, 1)=>I6(4.2, 1)  T(C) = {F, I6, I5, I4, I3, C}  B(I2)={ I1, A, F } |

Βήμα 18:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop F.  yF > ybottom => S inside  Ενώ για το A, που είναι το επόμενο στη λίστα:  yA < ybottom => E outside  Άρα έχω περίπτωση inside-outside, οπότε επιστρέφω το I7:  I7(xF + (ybottom – yF)/m, ybottom) =  =(3 - 1/1.5, 1)=>I7(2.3, 1)  T(C) = { I7, F, I6, I5, I4, I3, C }  B(I2)={ I1, A} |

Βήμα 19:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop A.  yA < ybottom => S outside  Ενώ για το I1:  yI1 > ybottom => E inside  Άρα έχω περίπτωση outside-inside, οπότε επιστρέφω το I8 και το E:  I8(xA + (ybottom – yA)/m, ybottom) =  =(2 - 0.5/1.3, 1)=>I8(1.62, 1)  T(C) = {I1, I8, I7, F, I6, I5, I4, I3, C }  B(I2)={ I1 } |



Βήμα 20:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop I1.  yI1 > ybottom => S inside  Ενώ για το I2:  yI2 > ybottom => E inside  Άρα έχω περίπτωση inside-inside, οπότε επιστρέφω μόνο το E:  T(C) = {I2, I1, I8, I7, F, I6, I5, I4, I3, C }  B(I2)={ } |

Βήμα 21:



|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Κάνω pop C από την T() πλέον.  yC > yTop=> S outside  Ενώ για το I3, που είναι το επόμενο στη λίστα:  yI3 < yTop => E inside  Άρα έχω περίπτωση outside-inside, οπότε επιστρέφω I9 και E:  I9(xC + (ytop – yC)/m, ytop) =  =(2.5 + 0.5/0.4, 4)=>I9(3.75, 4)  Final:{I3, I9}  T(C) = {I2, I1, I8, I7, F, I6, I5, I4, I3 } |

Βήμα 22:

|  |  |
| --- | --- |
|  | Απαραίτητες μεταβλητές και oι τιμές τους για τον αλγόριθμο S-H  Μετά από όμοιους υπολογισμούς για όλη την Τ() λίστα, βλέπουμε ότι όλα τα σημεία είναι inside-inside, άρα στο προτελευταίο είναι:  Final={ I2, I1, I8, I7, F, I6, I5, I4, I3, I9}  Τ(C)={I2}  Pop I2: yI2<ytop=>S inside  yC > ytop => E outside  Υπολογίζω το Ι10:  I10(xI2 + (ytop – yI2)/m, ytop) =  =(1 + 1/1, 4)=>I10(2, 4)  Άρα τελική λίστα:  Final={I10, I2, I1, I8, I7, F, I6, I5, I4, I3, I9}  Και οι τελικές ακμές είναι αυτές με μπλε χρώμα στο διπλανό σχήμα. |

## Άσκηση 8

….

## Άσκηση 9

RGB to HSV:

Max = max(R,G,B)

Min = min(R, G ,B)

H = 60\*(((G-B)/(max - min))mod6), if max = R

H = 60\*((B-R)/(max - min)+2), if max = G

H = 60\*((R-G)/(max - min)+4), if max = B

Else H = 0

S = (max - min)/max or S =0, if max = 0

V = max

HSV to RGB:

C = V\*S

X = C\*(1 – (H/60)mod2 -1)

m = V-C

(R’, G’, B’) = (C,X,0), if 0<H<60

= (X,C,0) if 60<H<120

= (0,C,X) if 120<H<180

= (0,X,C) if 180<H<240

= (X,0,C) if 240<H<300

= (C,0,X) if 300<H<360

(R,G,B) = ((R’+m)\*255, (G’+m)\*255, (B’+m)\*255)

# ΒΙΒΛΙΟΓΡΑΦΙΑ

1. [Program to Change RGB color model to HSV color model - GeeksforGeeks](https://www.geeksforgeeks.org/program-change-rgb-color-model-hsv-color-model/), 17.02.2022
2. [untitled (gomezconsultants.com)](http://gomezconsultants.com/CSE5280/Clipping/Clipping.html), 17.02.2022
3. [L-14\_Color.ppt (drexel.edu)](https://www.cs.drexel.edu/~david/Classes/CS430/Lectures/L-14_Color.6.pdf), 17.02.2022
4. [Polygon Clipping | Sutherland–Hodgman Algorithm - GeeksforGeeks](https://www.geeksforgeeks.org/polygon-clipping-sutherland-hodgman-algorithm-please-change-bmp-images-jpeg-png/), 24.02.2022

# ΠΑΡΑΡΤΗΜΑ

## Κώδικας Ερώτησης 6

Ακολουθεί ο κώδικας για το ερώτημα 6. Όνομα Αρχείου «exerc.c»

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

\* Ergasia 2 – Askhsh 6 – 24/02/2022

\* Poulianou Chaido - 2613

\* Polykarpou Christophoros - 2668

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

#include <GL/glut.h>

#include <stdlib.h>

#include <stdio.h>

int screenWidth = 640;

int screenHeight = 480;

int polygon\_state = 0;//0 to create polygon, 1 to cut

int sxhma = 0;

int cuttingWindow = 0;//if 1, the cutting window is seen

int size =0 ;

class GLintPoint{

public:

GLint x,y;

};

GLint numCorners=0, polCorners = 0;

void drawPoint(GLint x, GLint y)

{

glColor3f(0.0, 0.0, 1.0);

glBegin(GL\_POINTS);

glVertex2i(x,y);

glEnd();

glFlush();

}

GLintPoint clipWindow[2];

GLintPoint polygon[20]; //MAX polygon vertices 10

GLintPoint vertices\_buffer[20];//keeps the vertices of the previous checked window border

//essentially the output of the sutherland algorithm

void sutherland\_hodgeman(){

GLint xleft, xright, ytop, ybottom;

GLintPoint temp\_buffer[20];

int yinter, xinter,j = 0,k = 0 , i = 0 ;

//find left border

if(clipWindow[0].x < clipWindow[1].x){

xleft = clipWindow[0].x;

xright = clipWindow[1].x;

}

else{

xleft = clipWindow[1].x;

xright = clipWindow[0].x;

}

//find top and bottom

if(clipWindow[0].y < clipWindow[1].y){

ybottom = clipWindow[0].y;

ytop = clipWindow[1].y;

}

else{

ybottom = clipWindow[1].y;

ytop = clipWindow[0].y;

}

printf("L(%d), R(%d), B(%d), T(%d), TOTAL CORNERS %d\n", xleft, xright, ybottom, ytop, polCorners);

//LEFT BORDER WINDOW

//for each vertice check

for(i = 0; i < polCorners-1; i++){

//Check outside S

if(polygon[i].x < xleft){

//check outside-inside

if(polygon[i+1].x > xleft){//put I and E in buffer

printf(" O-I %d\n",i);

//calculate the intersection point I,

//antikatestise gia x = xleft kai vres to y apo thn eksiswsh evtheias P0P1 twn trexwn edges tou polygwnou

//yinter = y1 + m\*(xmin – x1), m = dy/dx

yinter = ((polygon[i].x\*polygon[i+1].y - polygon[i].y\*polygon[i+1].x) \* (ybottom-ytop) -

(polygon[i].y-polygon[i+1].y) \* (xleft\*ytop - ybottom\*xleft))/((polygon[i].x-polygon[i+1].x) \* (ybottom-ytop));

//polygon[i].y + ((polygon[i+1].y - polygon[i].y)/(polygon[i+1].x - polygon[i].x))\*(xleft - polygon[i].x);

xinter = xleft;

vertices\_buffer[j].x = xinter; vertices\_buffer[j].y = yinter;

j++;

vertices\_buffer[j].x = polygon[i+1].x; vertices\_buffer[j].y = polygon[i+1].y;

j++;

}

else{

printf("O-O\n");

}

}

else{

if(polygon[i+1].x > xleft){//inside-inside, output E

printf(" I-I %d\n",i);

vertices\_buffer[j].x = polygon[i+1].x; vertices\_buffer[j].y = polygon[i+1].y;

j++;

}

else{//inside-outside, output I

printf(" I-O %d\n",i);

yinter = ((polygon[i].x\*polygon[i+1].y - polygon[i].y\*polygon[i+1].x) \* (ybottom-ytop) -

(polygon[i].y-polygon[i+1].y) \* (xleft\*ytop - ybottom\*xleft))/((polygon[i].x-polygon[i+1].x) \* (ybottom-ytop));

//polygon[i].y + ((polygon[i+1].y - polygon[i].y)/(polygon[i+1].x - polygon[i].x))\*(-(xleft - polygon[i].x));

xinter = xleft;

vertices\_buffer[j].x = xinter; vertices\_buffer[j].y = yinter;

j++;

}

}

}

//check last with first vertex

if(polygon[polCorners-1].x < xleft){

//check outside-inside

if(polygon[0].x > xleft){//put I and E in buffer

printf(" O-I %d\n",i);

//calculate the intersection point I,

//antikatestise gia x = xleft kai vres to y apo thn eksiswsh evtheias P0P1 twn trexwn edges tou polygwnou

//yinter = y1 + m\*(xmin – x1), m = dy/dx

yinter = ((polygon[polCorners-1].x\*polygon[0].y - polygon[polCorners-1].y\*polygon[0].x) \* (ybottom-ytop) -

(polygon[polCorners-1].y-polygon[0].y) \* (xleft\*ytop - ybottom\*xleft))/((polygon[polCorners-1].x-polygon[0].x) \* (ybottom-ytop));

xinter = xleft;

vertices\_buffer[j].x = xinter; vertices\_buffer[j].y = yinter;

j++;

vertices\_buffer[j].x = polygon[0].x; vertices\_buffer[j].y = polygon[0].y;

j++;

}

}

else{

if(polygon[0].x > xleft){//inside-inside, output E

printf(" I-I %d\n",i);

vertices\_buffer[j].x = polygon[0].x; vertices\_buffer[j].y = polygon[0].y;

j++;

}

else{//inside-outside, output I

printf(" I-O %d\n",i);

yinter = ((polygon[polCorners-1].x\*polygon[0].y - polygon[polCorners-1].y\*polygon[0].x) \* (ybottom-ytop) -

(polygon[polCorners-1].y-polygon[0].y) \* (xleft\*ytop - ybottom\*xleft))/((polygon[polCorners-1].x-polygon[0].x) \* (ybottom-ytop));//polygon[polCorners-1].y + ((polygon[0].y - polygon[i].y)/(polygon[0].x - polygon[polCorners-1].x))\*(-(xleft - polygon[polCorners-1].x));

xinter = xleft;

vertices\_buffer[j].x = xinter; vertices\_buffer[j].y = yinter;

j++;

}

}

printf("\n================LEFT DONE J = %d\n", j);

//save vertices to buffer

for(k = 0; k <j; k++){

temp\_buffer[k] = vertices\_buffer[k];

printf("(%d,%d)->",vertices\_buffer[k].x,vertices\_buffer[k].y);

}

//RIGHT BORDER WINDOW

for(k =0, i = 0; i < j-1; i++){

//Check outside S

if(temp\_buffer[i].x > xright){

//check outside-inside

if(temp\_buffer[i+1].x < xright){//put I and E in buffer

yinter = ((temp\_buffer[i].x\*temp\_buffer[i+1].y - temp\_buffer[i].y\*temp\_buffer[i+1].x) \* (ybottom-ytop) -

(temp\_buffer[i].y-temp\_buffer[i+1].y) \* (xright\*ytop - ybottom\*xright))/((temp\_buffer[i].x-temp\_buffer[i+1].x) \* (ybottom-ytop));

//temp\_buffer[i].y + ((temp\_buffer[i+1].y - temp\_buffer[i].y)/(temp\_buffer[i+1].x - temp\_buffer[i].x))\*(xright - temp\_buffer[i].x);

xinter = xright;

printf("\n O-I %d", i);

vertices\_buffer[k].x = xinter; vertices\_buffer[k].y = yinter;

k++;

vertices\_buffer[k].x = temp\_buffer[i+1].x; vertices\_buffer[k].y = temp\_buffer[i+1].y;

k++;

}

}

else{

if(temp\_buffer[i+1].x < xright){//inside-inside, output E

vertices\_buffer[k].x = temp\_buffer[i+1].x; vertices\_buffer[k].y = temp\_buffer[i+1].y;

printf("\n I-I %d", i);

k++;

}

else{//inside-outside, output I

yinter = ((temp\_buffer[i].x\*temp\_buffer[i+1].y - temp\_buffer[i].y\*temp\_buffer[i+1].x) \* (ybottom-ytop) -

(temp\_buffer[i].y-temp\_buffer[i+1].y) \* (xright\*ytop - ybottom\*xright))/((temp\_buffer[i].x-temp\_buffer[i+1].x) \* (ybottom-ytop));

xinter = xright;

printf("\n I-O %d", i);

vertices\_buffer[k].x = xinter; vertices\_buffer[k].y = yinter;

k++;

}

}

}

//check last with first vertex

if(temp\_buffer[i].x > xright){

//check outside-inside

if(temp\_buffer[0].x < xright){//put I and E in buffer

printf(" O-I %d\n",i);

//calculate the intersection point I,

//antikatestise gia x = xleft kai vres to y apo thn eksiswsh evtheias P0P1 twn trexwn edges tou polygwnou

//yinter = y1 + m\*(xmin – x1), m = dy/dx

yinter = ((temp\_buffer[i].x\*temp\_buffer[0].y - temp\_buffer[i].y\*temp\_buffer[0].x) \* (ybottom-ytop) -

(temp\_buffer[i].y-temp\_buffer[0].y) \* (xright\*ytop - ybottom\*xright))/((temp\_buffer[i].x-temp\_buffer[0].x) \* (ybottom-ytop));

xinter = xright;

vertices\_buffer[k].x = xinter; vertices\_buffer[k].y = yinter;

k++;

vertices\_buffer[k].x = temp\_buffer[0].x; vertices\_buffer[k].y = temp\_buffer[0].y;

k++;

}

}

else{

if(temp\_buffer[0].x < xright){//inside-inside, output E

printf(" I-I %d\n",i);

vertices\_buffer[k].x = temp\_buffer[0].x; vertices\_buffer[k].y = temp\_buffer[0].y;

k++;

}

else{//inside-outside, output I

printf(" I-O %d\n",i);

yinter = ((temp\_buffer[i].x\*temp\_buffer[0].y - temp\_buffer[i].y\*temp\_buffer[0].x) \* (ybottom-ytop) -

(temp\_buffer[i].y-temp\_buffer[0].y) \* (xright\*ytop - ybottom\*xright))/((temp\_buffer[i].x-temp\_buffer[0].x) \* (ybottom-ytop));

xinter = xright;

vertices\_buffer[k].x = xinter; vertices\_buffer[k].y = yinter;

k++;

}

}

printf("\n================RIGHT DONE J = %d\n", k);

//save vertices to buffer

for(j = 0; j < k; j++){

temp\_buffer[j] = vertices\_buffer[j];

printf("(%d,%d)->",vertices\_buffer[j].x,vertices\_buffer[j].y);

}

//BOTTOM BORDER WINDOW

for(i = 0, j =0; i < k-1; i++){

//Check outside S

if(temp\_buffer[i].y < ybottom){

//check outside-inside

if(temp\_buffer[i+1].y > ybottom){//put I and E in buffer

xinter = ((temp\_buffer[i].x\*temp\_buffer[i+1].y - temp\_buffer[i].y\*temp\_buffer[i+1].x) \* (xleft-xright) -

(temp\_buffer[i].x-temp\_buffer[i+1].x) \* (xleft\*ybottom - ybottom\*xright))/(-(temp\_buffer[i].y-temp\_buffer[i+1].y) \* (xleft-xright));

//temp\_buffer[i].x + ((temp\_buffer[i+1].x - temp\_buffer[i].x)/(temp\_buffer[i+1].y - temp\_buffer[i].y))\*(ybottom - temp\_buffer[i].y);

yinter = ybottom;

printf("\n O-I %d", i);

vertices\_buffer[j].x = xinter; vertices\_buffer[j].y = yinter;

j++;

vertices\_buffer[j].x = temp\_buffer[i+1].x; vertices\_buffer[j].y = temp\_buffer[i+1].y;

j++;

}

}

else{

if(temp\_buffer[i+1].y > ybottom){//inside-inside, output E

vertices\_buffer[j].x = temp\_buffer[i+1].x; vertices\_buffer[j].y = temp\_buffer[i+1].y;

printf("\n I-I %d", i);

j++;

}

else{//inside-outside, output I

xinter = ((temp\_buffer[i].x\*temp\_buffer[i+1].y - temp\_buffer[i].y\*temp\_buffer[i+1].x) \* (xleft-xright) -

(temp\_buffer[i].x-temp\_buffer[i+1].x) \* (xleft\*ybottom - ybottom\*xright))/(-(temp\_buffer[i].y-temp\_buffer[i+1].y) \* (xleft-xright));

//temp\_buffer[i].x + ((temp\_buffer[i+1].x - temp\_buffer[i].x)/(temp\_buffer[i+1].y - temp\_buffer[i].y))\*(ybottom - temp\_buffer[i].y);

yinter = ybottom;

printf("\n I-O %d", i);

vertices\_buffer[j].x = xinter; vertices\_buffer[j].y = yinter;

j++;

}

}

}

//check last with first vertex

//Check outside S

if(temp\_buffer[i].y < ybottom){

//check outside-inside

if(temp\_buffer[0].y > ybottom){//put I and E in buffer

xinter = ((temp\_buffer[i].x\*temp\_buffer[0].y - temp\_buffer[i].y\*temp\_buffer[0].x) \* (xleft-xright) -

(temp\_buffer[i].x-temp\_buffer[0].x) \* (xleft\*ybottom - ybottom\*xright))/(-(temp\_buffer[i].y-temp\_buffer[0].y) \* (xleft-xright));

//temp\_buffer[i].x + ((temp\_buffer[0].x - temp\_buffer[i].x)/(temp\_buffer[0].y - temp\_buffer[i].y))\*(ybottom - temp\_buffer[i].y);

yinter = ybottom;

printf("\n O-I %d", i);

vertices\_buffer[j].x = xinter; vertices\_buffer[j].y = yinter;

j++;

vertices\_buffer[j].x = temp\_buffer[0].x; vertices\_buffer[j].y = temp\_buffer[0].y;

j++;

}

}

else{

if(temp\_buffer[0].y > ybottom){//inside-inside, output E

vertices\_buffer[j].x = temp\_buffer[0].x; vertices\_buffer[j].y = temp\_buffer[0].y;

printf("\n I-I %d", i);

j++;

}

else{//inside-outside, output I

xinter = ((temp\_buffer[i].x\*temp\_buffer[0].y - temp\_buffer[i].y\*temp\_buffer[0].x) \* (xleft-xright) -

(temp\_buffer[i].x-temp\_buffer[0].x) \* (xleft\*ybottom - ybottom\*xright))/(-(temp\_buffer[i].y-temp\_buffer[0].y) \* (xleft-xright));

//temp\_buffer[i].x + ((temp\_buffer[0].x - temp\_buffer[i].x)/(temp\_buffer[0].y - temp\_buffer[i].y))\*(ybottom - temp\_buffer[i].y);

yinter = ybottom;

printf("\n I-O %d", i);

vertices\_buffer[j].x = xinter; vertices\_buffer[j].y = yinter;

j++;

}

}

printf("\n================BOTTOM DONE J = %d\n", j);

//save vertices to buffer

for(k = 0; k < j; k++){

temp\_buffer[k] = vertices\_buffer[k];

printf("(%d,%d)->",vertices\_buffer[k].x,vertices\_buffer[k].y);

}

//TOP BORDER WINDOW

for(i = 0, k =0; i < j-1; i++){

//Check outside S

if(temp\_buffer[i].y > ytop){

//check outside-inside

if(temp\_buffer[i+1].y < ytop){//put I and E in buffer

printf("\n O-I %d",i);

xinter = ((temp\_buffer[i].x\*temp\_buffer[i+1].y - temp\_buffer[i].y\*temp\_buffer[i+1].x) \* (xleft-xright) -

(temp\_buffer[i].x-temp\_buffer[i+1].x) \* (xleft\*ytop - ytop\*xright))/(-(temp\_buffer[i].y-temp\_buffer[i+1].y) \* (xleft-xright));

//temp\_buffer[i].x + ((temp\_buffer[i+1].x - temp\_buffer[i].x)/(temp\_buffer[i+1].y - temp\_buffer[i].y))\*(ytop - temp\_buffer[i].y);

yinter = ytop;

vertices\_buffer[k].x = xinter; vertices\_buffer[k].y = yinter;

k++;

vertices\_buffer[k].x = temp\_buffer[i+1].x; vertices\_buffer[k].y = temp\_buffer[i+1].y;

k++;

}

}

else{

if(temp\_buffer[i+1].y < ytop){//inside-inside, output E

printf("\n I-I %d",i);

vertices\_buffer[k].x = temp\_buffer[i+1].x; vertices\_buffer[k].y = temp\_buffer[i+1].y;

k++;

}

else{//inside-outside, output I

xinter = ((temp\_buffer[i].x\*temp\_buffer[i+1].y - temp\_buffer[i].y\*temp\_buffer[i+1].x) \* (xleft-xright) -

(temp\_buffer[i].x-temp\_buffer[i+1].x) \* (xleft\*ytop - ytop\*xright))/(-(temp\_buffer[i].y-temp\_buffer[i+1].y) \* (xleft-xright));

yinter = ytop;

printf("\n I-O %d",i);

vertices\_buffer[k].x = xinter; vertices\_buffer[k].y = yinter;

k++;

}

}

}

//check last vertice

//Check outside S

if(temp\_buffer[i].y > ytop){

//check outside-inside

if(temp\_buffer[0].y < ytop){//put I and E in buffer

printf("\n O-I %d",i);

xinter = ((temp\_buffer[i].x\*temp\_buffer[0].y - temp\_buffer[i].y\*temp\_buffer[0].x) \* (xleft-xright) -

(temp\_buffer[i].x-temp\_buffer[0].x) \* (xleft\*ytop - ytop\*xright))/(-(temp\_buffer[i].y-temp\_buffer[0].y) \* (xleft-xright));

yinter = ytop;

vertices\_buffer[k].x = xinter; vertices\_buffer[k].y = yinter;

k++;

vertices\_buffer[k].x = temp\_buffer[0].x; vertices\_buffer[k].y = temp\_buffer[0].y;

k++;

}

}

else{

if(temp\_buffer[0].y < ytop){//inside-inside, output E

printf("\n I-I %d",i);

vertices\_buffer[k].x = temp\_buffer[0].x; vertices\_buffer[k].y = temp\_buffer[0].y;

k++;

}

else{//inside-outside, output I

xinter = ((temp\_buffer[i].x\*temp\_buffer[0].y - temp\_buffer[i].y\*temp\_buffer[0].x) \* (xleft-xright) -

(temp\_buffer[i].x-temp\_buffer[0].x) \* (xleft\*ytop - ytop\*xright))/(-(temp\_buffer[i].y-temp\_buffer[0].y) \* (xleft-xright));

yinter = ytop;

printf("\n I-O %d %d",i, xinter);

vertices\_buffer[k].x = xinter; vertices\_buffer[k].y = yinter;

k++;

}

}

printf("\n================TOP DONE J = %d\n", k);

for(i = 0;i < k; i++){

polygon[i].x = vertices\_buffer[i].x;

polygon[i].y = vertices\_buffer[i].y;

printf("(%d, %d) ",vertices\_buffer[i].x,vertices\_buffer[i].y);

}

size = k;

polCorners = size;

};

void init (void)

{

glClear(GL\_COLOR\_BUFFER\_BIT);//The buffers currently enabled for color writing

glClearColor (0.0, 0.0, 0.0, 1.0); // select clearing (background) color

glColor3f(0.0, 0.0, 1.0);

glPointSize(6);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0.0,(GLdouble)screenWidth,0.0,(GLdouble)screenHeight); //define a 2-D orthographic projection matrix

}

void display(void){

//glClear(GL\_COLOR\_BUFFER\_BIT); //The buffers currently enabled for color writing

//glColor3f(1.0, 0.0, 0.0);

if(sxhma == 1){

//printf("WINDOW\n");

if (numCorners ==2){

glClear(GL\_COLOR\_BUFFER\_BIT);//everytime clear the scene, otherwise bug

glColor3f(1.0, 0.0, 0.0);

glRecti(clipWindow[0].x, clipWindow[0].y, clipWindow[1].x, clipWindow[1].y);

cuttingWindow = 1;

//create the polygon

glBegin(GL\_POLYGON);

glColor3f(0.0, 0.0, 1.0);

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

glVertex2i(polygon[i].x,polygon[i].y);

}

glEnd();

glFlush();

}

}

if(sxhma == 2) //create and fill polygon

{

//create the polygon

glBegin(GL\_POLYGON);

glColor3f(0.0, 0.0, 1.0);

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

glVertex2i(polygon[i].x,polygon[i].y);

}

glEnd();

glFlush();

}

if(sxhma == 3){//cut polygon

//create the cut polygon

printf("Sutherland %d\n", size);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 0.0, 0.0);

glRecti(clipWindow[0].x, clipWindow[0].y, clipWindow[1].x, clipWindow[1].y);

glBegin(GL\_POLYGON);

glColor3f(0.0, 0.0, 1.0);

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

glVertex2i(vertices\_buffer[i].x,vertices\_buffer[i].y);

}

glEnd();

glFlush();

}

glFlush();

}

void keyCut(int key,int x,int y){ // x,y mouse location when key pressed

switch (key) {

// Polygon clip-> call Sutherland – Hodgeman routine

//F1, go to polygon cutting state

case GLUT\_KEY\_F1:

default:

polygon\_state = 1;

break;

}

}

// triggered when a keyboard key is pressed

void keyEvent(unsigned char key,int x,int y){ // x,y mouse location when key pressed

switch (key) {

// Polygon clip-> call Sutherland – Hodgeman routine

case 'C':

case 'c':

sutherland\_hodgeman();

sxhma = 3;

glutPostRedisplay();

break;

//pressing P draws the polygon

case 'P':

case 'p':

sxhma = 2;

glutPostRedisplay();

break;

//space: show/hide cutting polygon

case 32:

glClear(GL\_COLOR\_BUFFER\_BIT); // clear the window

//check if polygon cut window active

//if yes, hide it, just redraw the polygon

//if no, draw it and then redraw polygon

if(cuttingWindow == 0){

glColor3f(1.0, 0.0, 0.0);

glRecti(clipWindow[0].x, clipWindow[0].y, clipWindow[1].x, clipWindow[1].y);

glutPostRedisplay();

cuttingWindow = 1;

}

else

cuttingWindow = 0;

//and redraw polygon

sxhma = 2;

glutPostRedisplay();

break;

case 'Q':

exit(0);

break;

default:

break;

}

}

void myMouse(int button, int state, GLint xm, GLint ym)

{

//polygon cutting

if(button==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN && polygon\_state == 1)

{

glColor3f(0.0, 0.0, 1.0);

clipWindow[0].x= xm;

clipWindow[0].y =screenHeight - ym;

numCorners =1;

glutPostRedisplay();

}

//polygon creating

else if(button==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN && polygon\_state == 0)

{

glColor3f(1.0, 1.0, 1.0);

printf("POINT %d\n\n", polCorners);

polygon[polCorners].x= xm;

polygon[polCorners].y =screenHeight - ym;

drawPoint(polygon[polCorners].x, polygon[polCorners].y);

printf("(%d, %d)\n", polygon[polCorners].x, polygon[polCorners].y);

polCorners++;

glFlush();

glutPostRedisplay();

}

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

printf("Right click\n");

glClear(GL\_COLOR\_BUFFER\_BIT);

polCorners = 0;

numCorners = 0;

cuttingWindow = 0;

polygon\_state = 0;

sxhma = 0;

}

if (button==GLUT\_LEFT\_BUTTON && state==GLUT\_UP && polygon\_state == 1){

printf("RELEASED\n");

}

glFlush();

}

void moveMouse(int xnew, int ynew)

{

if(polygon\_state == 1){

clipWindow[1].x= xnew;

clipWindow[1].y =screenHeight - ynew;

numCorners=2;

sxhma = 1;

glutPostRedisplay();

}

}

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

{

//initialize toolkit

glutInit(&argc, argv);

//set display mode

glutInitDisplayMode (GLUT\_SINGLE | GLUT\_RGB);

//set window size

glutInitWindowSize (640, 480);

//set window initial position

glutInitWindowPosition (1, 1);

//creates a window

glutCreateWindow ("glutMouseFunc2");

glutDisplayFunc(display);

//registers a keyboard handler

glutKeyboardFunc(keyEvent);

glutSpecialFunc(keyCut);

glutMouseFunc(myMouse);

glutMotionFunc(moveMouse);

//additional initializatons

init ();

//perpetual ever processing loop

glutMainLoop();

return 0;

}