ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΪΑΤΡΙΚΗ

ΧΑΡΑΞΗ ΚΥΚΛΩΝ

ΕΡΓΑΣΙΑ ΓΙΑ ΤΟ ΜΑΘΗΜΑ ΓΡΑΦΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΚΠΟΝΗΘΗΚΕ ΑΠΟ ΤΟΥΣ:

Καρούτσο Παναγιώτη (pkaroutsos@uth.gr)

Μπάκα Ευγενία - Βαΐα (ebaka@uth.gr)

ΔΙΔΑΣΚΩΝ ΚΑΘΗΓΗΤΗΣ:

Δρακόπουλος Βασίλειος

ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ: 2022-2023

Πίναξ περιεχομένων

Περίληψη

Η εργασία αυτή αφορά τη Χάραξη Κύκλων, όπου δυναμικά και σε πραγματικό χρόνο, μπορούν να μεταβληθούν οι συντεταγμένες του κέντρου, η ακτίνα και το πάχος γραμμής, ενώ παράλληλα σε κάθε μία από αυτές τις αλλαγές, ο κύκλος επαναχαράσσεται. Ο κώδικας αναπτύχθηκε σε γλώσσα C++ με χρήση συναρτήσεων της βιβλιοθήκης γραφικών OpenGL.

Η πρώτη ενότητα περιέχει το θεωρητικό κομμάτι από την ύλη του μαθήματος που χρειάστηκε για την εκπόνηση της εργασίας.

Η δεύτερη ενότητα αποτελείται από μια λεπτομερή ανάλυση του πηγαίου κώδικα που υλοποιήθηκε για τον σχεδιασμό των κύκλων καθώς και την υλοποίηση των δυναμικών αλλαγών. Έπειτα, εξηγείται ο τρόπος με τον οποίο ενώνονται τα επιμέρους κομμάτια κώδικα που χρειάστηκαν.

Στην τρίτη ενότητα εξάγονται τα συμπεράσματα μας και τέλος, καταγράφεται η πληθώρα πηγών που χρησιμοποιήθηκε.

Abstract

This paper focuses on the implementation of Circle Drawing, where the coordinates of the center, the radius, and line thickness can dynamically and in real-time be changed while simultaneously redrawing the circle, following any of these changes. The code was developed in C++ using functions from the OpenGL graphics library.

The first section contains the theoretical information from the course's material that was necessary for the realisation of the paper.

The second section consists of a detailed analysis of the source code implemented for circle drawing as well as the implementation of the aforementioned dynamic changes. Afterwards, the method in which all of the needed code segments were connected is explained.

In third section our conclusions are drawn and finally, the plethora of sources used is recorded.

Λέξεις κλειδιά: Circles, Bresenham, OpenGL, C++

Εισαγωγή

Με τους όρους Χάραξη Κύκλων εννοούμε την ψηφιδόξυση γεωμετρικών κύκλων σε μια επιφάνεια, συνήθως σε ένα γραφικό περιβάλλον. Η χάραξη κύκλων αποτελεί ένα σημαντικό θέμα στην υπολογιστική γραφική και έχει πολλές εφαρμογές στην απεικόνιση γραφικών αντικειμένων, τη δημιουργία περιγραφικών γεωμετρικών σχημάτων και άλλα.

Σκοπός αυτής της εργασίας είναι η περιγραφή της θεωρίας που αφορά τη χάραξη κύκλων, καθώς και η υλοποίησή της σε ένα περιβάλλον C++ OpenGL. Η εργασία χωρίζεται σε δύο κύριες ενότητες. Στην πρώτη ενότητα, παρουσιάζεται η θεωρία πίσω από την χάραξη κύκλων, συμπεριλαμβανομένων των αλγορίθμων και τεχνικών που χρησιμοποιούνται. Αναλύονται οι μαθηματικές αρχές και οι συναρτήσεις που απαιτούνται για να πραγματοποιηθεί η χάραξη κύκλων με ακρίβεια και αποτελεσματικότητα. Στην δεύτερη ενότητα, παρουσιάζεται η υλοποίηση της χάραξης κύκλων σε ένα περιβάλλον C++ OpenGL. Αναλύονται τα βήματα που ακολουθήθηκαν για να δημιουργηθεί η αντίστοιχη εφαρμογή, περιλαμβανομένων των σχεδιαστικών αποφάσεων και των προγραμματιστικών τεχνικών που εφαρμόστηκαν. Επίσης, παρουσιάζονται και αξιολογούνται τα αποτελέσματα που προέκυψαν από την υλοποίηση της χάραξης κύκλων.

1 Θεωρία

«Κύκλος ἐστὶ σχῆμα ἐπίπεδον ὑπὸ μιᾶς γραμμῆς περιεχόμενον [ἣ καλεῖται περιφέρεια], πρὸς ἣν ἀφ᾽ ἑνὸς σημείου τῶν ἐντὸς τοῦ σχήματος κειμένων πᾶσαι αἱ προσπίπτουσαι εὐθεῖαι [πρὸς τὴν τοῦ κύκλου περιφέρειαν] ἴσαι ἀλλήλαις εἰσίν.

Σύμφωνα με τον Ευκλείδη δηλαδή: κύκλος είναι το επίπεδο σχήμα το οποίο περιέχεται σε μια γραμμή που ονομάζεται περιφέρεια, της οποίας τα σημεία ισαπέχουν από ένα σημείο που βρίσκεται στο εσωτερικό του κύκλου. [Γαρυφαλίδης, 2016]

Η χάραξη ενός συνεχούς ομαλού τόξου στην οθόνη του υπολογιστή δεν είναι μια εύκολη υπόθεση αφού η οθόνη απαρτίζεται από έναν πίνακα από pixel [Greenberg, 2016]. Για αυτό τον λόγο, σε πολλές βιβλιοθήκες γραφικών περιλαμβάνονται συναρτήσεις υπεύθυνες για την παραγωγή και προβολή καμπυλών, τόξων ή πλήρων κύκλων [GeeksForGeeks, 2021].

Για ένα οποιοδήποτε σημείο του κύκλου(x,y), η σχέση απόστασης του σημείου από το κέντρο ονομάζεται ακτίνα(r) και εκφράζεται από το πυθαγόρειο θεώρημα σε Καρτεσιανές συντεταγμένες ως: $(x - x_c)^2 + (y-y_c)^2 = r^2$ ενώ από αυτήν την εξίσωση μπορεί κανείς να προσδιορίσει τις συντεταγμένες των σημείων στην περίμετρο ενός κυκλικού σχήματος, διασχίζοντας σταδιακά κατά μήκος του οριζόντιου άξονα και υπολογίζοντας την αντίστοιχη κατακόρυφη θέση σε κάθε βήμα. Αυτή η διαδικασία περιλαμβάνει τη

χρήση της εξίσωσης $y = y_c \pm \sqrt{r^2 - (x_c - x)^2}$. [Hearn et al., 2014] . Ωστόσο, θα πρέπει να αναγνωριστεί ότι αυτή η προσέγγιση μπορεί να μην αποφέρει σταθερά τη βέλτιστη αποτελεσματικότητα, λόγω του σημαντικού πλήθους υπολογισμών. [Hearn et al., 2014]

Μια εναλλακτική προσέγγιση για τον υπολογισμό της θέσης των σημείων στην περιφέρεια του κύκλου είναι να λάβουμε υπόψη τη συμμετρία των κύκλων, χωρίζοντας τον κύκλο σε 8 οκταμόρια όπως φαίνεται στην **εικόνα 1**. Οι 8 περιοχές είναι αποτέλεσμα των συμμετριών του κύκλου ως προς τους άξονες x, y και την διαγώνιο. Έτσι, αν υπολογίσουμε τα σημεία σε μία από αυτές, μπορούμε να προσθέσουμε συμμετρικά τα αντίστοιχα σημεία στις υπόλοιπες, με αποτέλεσμα να μειώσουμε την πολυπλοκότητα. [Mukherjee and Jana, 2010]

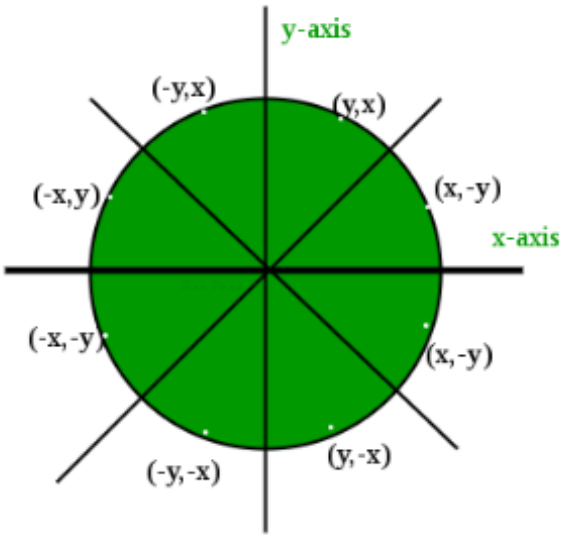

Figure 1: Οκταμόρια

Έτσι, οι δύο επικρατέστεροι τρόποι χάραξης κύκλων στη γραφική υπολογιστών είναι ο αλγόριθμος χάραξης ενδιάμεσου σημείου και ο αλγόριθμος χάραξης κύκλων του Bresenham.[Foley et al., 1996]

Όσον αφορά τον αλγόριθμο κύκλου ενδιάμεσου σημείου(Middlepoint Circle Algorithm) αξιοποιούμε τη συμμετρία του κύκλου και άρα τα 8 τμήματα που ορίσαμε παραπάνω. Αυτό σημαίνει ότι μπορούμε να υπολογίσουμε τα σημεία στην περιφέρεια του κύκλου μόνο για ένα ογδοημόριο και στη συνέχεια να τα αντιγράψουμε στα υπόλοιπα ογδοημόρια χρησιμοποιώντας τις αντίστοιχες συμμετρίες. Ο αλγόριθμος κύκλου ενδιάμεσου σημείου χρησιμοποιεί τη συμμετρία του κύκλου για να υπολογίσει τα σημεία στο πρώτο ογδοημόριο. Ξεκινάμε από το (0, r) και κινούμαστε κατακόρυφα και διαγωνίως μέχρι x >= y. Οι θετικές τιμές είναι όταν x > y, ενώ οι αρνητικές όταν x <= y. Κατά την προσέγγιση αυτή, οι πράξεις

ελαττώνονται σημαντικά, παρ' όλα αυτά ο αλγόριθμος κύκλου ενδιάμεσου σημείου δεν είναι ακριβής και δεν χρησιμοποιείται συχνά σε εφαρμογές που απαιτούν ακρίβεια στην περιγραφή του κύκλου. Ωστόσο, η απλότητά του τον καθιστά χρήσιμο για γραφικές εφαρμογές όπου η ακρίβεια δεν είναι το κυρίαρχο κριτήριο.[Hearn et al., 2014][Mukherjee and Jana, 2010]

Συνεπώς, ένας αλγόριθμος εξίσου απλός αλλά περισσότερο ακριβής, γρήγορος και με ομορφότερο οπτικό αποτέλεσμα στη γραφική υπολογιστών, είναι ο αλγόριθμος χάραξης κύκλων του Bresenham.[Foley et al., 1996]

Στον παρακάτω πίνακα προβάλλονται τα πλεονεκτήματα και τα μειονεκτήματα των 2 αυτών αλγορίθμων. [Hearn et al., 2014][Whitrow, 2008]

Table 1: Πίνακας 1

Πλεονέκτημα	**Middlepoint**	**Bresenham**
Ακρίβεια	Χαμηλή	Υψηλή
Πολυπλοκότητα	Χαμηλή	Χαμηλή
Απόδοση	Μέτρια	Υψηλή
Ανεξαρτησία από κλίση	Ναι	Όχι
Ανεξαρτησία από ακτίνα κύκλου	Όχι	Ναι
Πλήθος απαιτούμενων πράξεων	Χαμηλό	Μέτριο
Μπορεί να επεκταθεί σε ελλείψεις	Ναι	Ναι

Όπως συμπεραίνουμε από τον **πίνακα 1**, αποδοτικότερος για την υλοποίηση της χάραξης κύκλων σε opengl περιβάλλον, θεωρήθηκε ο αλγόριθμος του Bresenham.

2 Υλοποίηση

Σε αυτό το κομμάτι, θα επικεντρωθούμε στον κώδικα που συγγράψαμε για την υλοποίηση των ερωτημάτων της εργασίας.

2.1 Ψηφιδόξυση του κύκλου με τη διαδικασία circle

```
void set8pixels(int xc, int yc, int x, int y, float pointSize)
```

```
class Circle
```

```
void Circle::draw() const   // Draws the circle using Bresenham's algorithm
    for circle drawing
{
    glPointSize(lineSize);
```

```
    glColor3f(1.0, 0.0, 0.0);
    int x = 0;
    int y = static_cast<int>(radius);
    int e = -static_cast<int>(radius);

    while (x <= y)
    {
        set8pixels(static_cast<int>(centerX), static_cast<int>(centerY), x, y, lineSize);
        x = x + 1;
        e = e + 2 * x + 1;
        if (e >= 0)
        {
            y = y - 1;
            e = e - 2 * y + 2;
        }
    }
}
```

Για τη δημιουργία του κύκλου, αρχικά δημιουργήσαμε τη συνάρτηση set8pixels, η οποία σχεδιάζει 8 συμμετρικά pixels, σύμφωνα με την οκταπλή συμμετρία ενός κύκλου με κέντρο xc και yc και πάχος του pixel pointSize. [Δρακόπουλος, 2023]

Έπειτα δημιουργείται μια κλάση (ένα αντικείμενο) Circle, με όλα τα χαρακτηριστικά του κύκλου που χρειαζόμαστε, ώστε να δημιουργήσουμε μια μεταβλητή τύπου Circle για τη σχεδίασή του.

Η Circle::draw() καθορίζει τον τρόπο που θα σχεδιαστεί το αντικείμενο αυτό, στη συγκεκριμένη περίπτωση δηλαδή, μέσω του αλγορίθμου του Bresenham. (Ενότητα 1, σελίδα 5)[Foley et al., 1996]

```
void MousePressed(int button, int state, int x, int y)  // Checks if the left mouse button is pressed
{
    isMouseDown = (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN); // Sets isMouseDown to listen for when the left mouse button is pressed
    if (isMouseDown && !isFirstClick)   // If the mouse click isn't the first click it then becomes the first click (so that the circle is drawn)
    {
        isFirstClick = true;
    }
```

```
}
```

Στη συνέχεια, δημιουργήσαμε τη συνάρτηση MousePressed, η οποία ελέγχει αν έχει πατηθεί το αριστερό κουμπί του ποντικιού. Ορίζει έναν Listener στη boolean μεταβλητή isMouseDown, ο οποίος αρχικοποιείται με την τιμή false και παίρνει την τιμή true όποτε πατιέται το κουμπί. Έπειτα, αν οι isMouseDown και isFirstClick έχουν την τιμή true και false αντίστοιχα, τότε η isFirstClick παίρνει την τιμή true, ώστε να χρησιμοποιηθεί παρακάτω.

```
void Draw()
```

```
    if (isFirstClick)   // If a left mouse button click is the first click, draw the circle, where mouseX = xc, mouseY = yc, currentRadius = r, currentLineSize = size
    {
        Circle circle(mouseX, mouseY, currentRadius, currentLineSize);
        circle.draw();
    }
```

Εντός της συνάρτησης Draw(), ελέγχεται αν το isFirstClick έχει την τιμή true, δηλαδή αν το αριστερό κλικ που πατήθηκε ήταν το πρώτο κλικ, όσο τρέχει το πρόγραμμα. Σε αυτήν την περίπτωση καλείται ένα αντικείμενο Circle με το όνομα circle και τις παραμέτρους mouseX, mousey, currentRadius και currentLineSize (αντίστοιχα xc, yc, r, size). Έπειτα σχεδιάζεται ο κύκλος με την εντολή circle.draw().

2.2 Μεταβολή των συντεταγμένων του κέντρου του κύκλου (xc, yc) σε πραγματικό χρόνο

```
float mouseX = 0.0, mouseY = 0.0;
bool isMouseDown = false;
```

```
void MouseMove(int x, int y)    // Moves the circle according to the movement of the mouse only while the left mouse button is pressed
{
    if (isMouseDown)
    {
        mouseX = x - windowWidth / 2;
        mouseY = windowHeight / 2 - y;
    }
}
```

Η συνάρτηση MouseMove ελέγχει αν έχει πατηθεί το αριστερό κουμπί του ποντικιού, δηλαδή αν η isMouseDown από τη συνάρτηση MousePressed που προαναφέραμε, έχει την τιμή true, στην οποία

περίπτωση τότε, μεταβάλλει τις μεταβλητές mouseX και mouseY κατάλληλα, ώστε να φαίνεται ότι ο κύκλος μετακινείται σύμφωνα με την κίνηση του κέρσορα εντός του παραθύρου προβολής.

```
void idle(){    // Redisplay the circle if no other event is happening (
    idle state)
    glutSetWindow(MAIN_WINDOW); // Specifies the currently active window in
     case something goes wrong
    glutPostRedisplay();
}
```

Με τη συνάρτηση idle(), επανασχεδιάζουμε τον κύκλο όταν δεν υπάρχει κανένα άλλο συμβάν όσο τρέχει το πρόγραμμα, δηλαδή όταν δεν συμβαίνει τίποτα. Έτσι, μετά από κάθε άλλαγή, ο κύκλος επανασχεδιάζεται ώστε η κάθε αλλαγή που επέβαλλε ο χρήστης να είναι ορατή.

Τη μεταβολή των συντεταγμένων του κέντρου του κύκλου, την υλοποιήσαμε και με αυτόν τον εναλλακτικό τρόπο, που χρησιμοποιεί τα πλήκτρα W, A, S και D για να μετακινήσει τον κύκλο πάνω, αριστερά, κάτω και δεξιά αντίστοιχα.

```
enum key_state {NOTPUSHED, PUSHED} keyArray[127];   // Enumeration type (
    such as true/false) array to store the state (pushed or not) of keyboard
     keys
```

Η key_state είναι μια μεταβλητή απαρίθμησης τύπου boolean, με βάση την οποία δημιουργούμε τον πίνακα keyArray, στον οποίο θα αποθηκεύεται το κάθε πλήκτρο που χρησιμοποιήσαμε, με την τιμή PUSHED ή NOTPUSHED.

```
void KeyboardFunc(unsigned char key, int x, int y)  // Checks if a
    specified key is pressed and saves it as PUSHED in the keyArray
```

```
    if (key == 'W' || key == 'w'){ // W key
        keyArray['W'] = PUSHED;
        keyArray['w'] = PUSHED;
    }
```

Η KeyboardFunc ελέγχει αν έχει πατηθεί οποιοδήποτε από τα W, A, S και D πλήκτρα. Αν κάποιο από αυτά έχει πατηθεί, αποθηκεύεται στον keyArray με την τιμή PUSHED. (Στην ίδια συνάρτηση ελέγχεται επίσης αν έχει πατηθεί το πλήκτρο Esc, όπου τότε το πρόγραμμα τερματίζεται.[theasciicode.com,])

```
void KeyboardUpFunc(unsigned char key, int x, int y)    // Checks if a
    specified key is no longer pushed and saves it as NOTPUSHED in the
    keyArray
```

```
    if (key == 'W' || key == 'w'){ // W key
        keyArray['W'] = NOTPUSHED;
        keyArray['w'] = NOTPUSHED;
    }
```

Ομοίως, η KeyboardUpFunc ελέγχει αν έχει αφεθεί κάποιο από αυτά τα πλήκτρα (πλην του Esc) και στην περίπτωση αυτή, αποθηκεύεται το κάθε πλήκτρο στον keyArray με την τιμή NOTPUSHED.

```
    if (keyArray['w'] || keyArray['W']){    // Move circle upwards when 'w'
    or 'W' is pressed
        mouseY += 0.08;
    }
```

```
    if (keyArray['d'] || keyArray['D']){    // Move circle to the right
   when 'd' or 'D' is pressed
        mouseX += 0.08;
    }
```

Εντός της συνάρτησης Draw(), ελέγχεται, κατά αυτόν τον τρόπο αν έχει πατηθεί το κάθε κουμπί ξεχωριστά και έπειτα μετακινείται ο κύκλος μέσω της mouseX (αντιστοιχεί στη xc) και της mouseY (αντιστοιχεί στην yc).

Ελέγχοντας αν έχουν πατηθεί τα κουμπιά κατά αυτόν τον τρόπο, επιτρέπεται στο χρήστη να μετακινήσει τον κύκλο ταυτόχρονα ως προς και τους 2 άξονες, δηλαδή να μεταβάλλει ταυτόχρονα και την xc και την yc (mouseX και mouseY).

```
void context_menu(int option)   // Pop up menu that displays all movement/
   adjustment options for the circle
```

Παρόμοια λειτουργικότητα υπάρχει και στο pop-up μενού που μπορεί ο χρήστης να καλέσει με δεξί κλικ, όπου υπάρχουν 4 επιλογές, με την κάθε μια να αντιστοιχεί σε κίνηση του κύκλου, αυξάνοντας ή μειώνοντας τη mouseX και τη mouseY.

2.3 Μεταβολή της ακτίνας r του κύκλου και του πάχους της γραμμής σχεδίασης σε πραγματικό χρόνο

```
float minLineSize = 1.0;
float maxLineSize = 100.0; // OpenGl and/or system limitations do not
   visually allow higher lineSize for the circle than this
float currentLineSize = minLineSize;
```

```
float minRadius = 20.0;
float maxRadius = 410.0;
float currentRadius = minRadius;
```

Αρχικοποιούμε τις απαραίτητες μεταβλητές για την ελάχιστη και μέγιστη τιμή της ακτίνας, καθώς και για το ελάχιστο και μέγιστο πάχος γραμμής. Επίσης αρχικοποιούμε μεταβλητές και την τρέχουσα ακτίνα και το τρέχον πάχος γραμμής, στις οποίες αρχικά ορίζουμε την ελάχιστη τιμή. Εδώ αξίζει να σημειώσουμε, πως αυξάνοντας το πάχος γραμμής, δημιουργείται ένας Δακτύλιος. Η ακτίνα, την οποία διατηρούμε σταθερή, είναι ο μέσος όρος της εσωτερικής και της εξωτερικής ακτίνας.

Ο κώδικας που απαιτείται είναι ακριβώς ο ίδιος με αυτόν που χρειάστηκε για τα πλήκτρα W, A, S και D, μόνο που εδώ χρησιμοποιήσαμε τις συναρτήσεις SpecialFunc και SpecialUpFunc για να ελέγχουμε πότε πατήθηκε και πότε αφέθηκε ένα από τα Arrow πλήκτρα. [GameDevelopment, 2018]

Συγκεκριμένα, το δεξί και το αριστερό βέλος αυξάνουν και μειώνουν την ακτίνα του κύκλου αντίστοιχα, ενώ το πάνω και κάτω αριστερό βέλος αυξάνουν και μειώνουν το πάχος γραμμής του κύκλου αντίστοιχα, μέσω των μεταβλητών currentRadius και currentLineSize αντίστοιχα. Επιπλέον, υπάρχει έλεγχος ώστε να μην ξεπερνάται το maxRadius και το maxLineSize (Οι αντίστοιχες minimum τιμές δεν ξεπερνώνται λόγω του αρχικού ορισμού τιμής στις current μεταβλητές). Οι έλεγχοι αυτοί βρίσκονται επίσης εντός της συνάρτησης Draw(), γεγονός που επιτρέπει στο χρήστη να μεταβάλλει σε πραγματικό χρόνο, ταυτόχρονα και τη θέση του κύκλου αλλά και την ακτίνα και το πάχος γραμμής. [gamedev.net, 2010] [stack overflow, 2009]

Αντίστοιχη λειτουργικότητα υπάρχει και στο pop-up μενού που μπορεί ο χρήστης να καλέσει με δεξί κλικ, όπου υπάρχουν ακόμα 4 επιλογές, με την κάθε μια να αντιστοιχεί σε μεταβολή της ακτίνας και του πάχους γραμμής, αυξάνοντας ή μειώνοντας την currentRadius και την currentLineSize αντίστοιχα.

2.4 Γραφικά Αποτελέσματα

Στις παρακάτω εικόνες θα δούμε το γραφικό αποτέλεσμα του κώδικα καθώς και στιγμιότυπα μετά από αυξομειώσεις πάχους γραμμής και ακτίνας.

2.5 Προσθήκη Γραφικών GLUI

Στην παρακάτω εικόνα, παρατηρούμε ένα πλήρες στιγμιότυπο της εφαρμογής σχεδίασης κύκλων. Εκτός από τη διαχείριση του μεγέθους της ακτίνας κύκλου και του πάχους γραμμής, καθώς και τη μετατόπιση του κύκλου με τη χρήση κουμπιών και ποντικιού, πλέον δίνεται η δυνατότητα να ορίσουμε όλες τις παραπάνω μεταβλητές χρησιμοποιώντας το γραφικό περιβάλλον GLUI.

Figure 2: 3 Στιγμιότυπα του εκτελέσιμου αρχείου

3 Συμπεράσματα

Στο πλαίσιο αυτής της εργασίας, πραγματοποιήθηκε η περιγραφή και η υλοποίηση της χάραξης κύκλων σε ένα περιβάλλον C++ OpenGL. Μέσω της μελέτης της θεωρίας και των αλγορίθμων που αφορούν την χάραξη κύκλων, κατανοήθηκαν οι απαιτήσεις και οι διαδικασίες που εμπλέκονται σε αυτήν την διαδικασία. Η υλοποίηση της χάραξης κύκλων στο προγραμματιστικό περιβάλλον, αποδείχθηκε αποτελεσματική και ακριβής. Ο αλγόριθμος Bresenham και οι τεχνικές που χρησιμοποιήθηκαν, επέτρεψαν τη σωστή απεικόνιση ενός γεωμετρικού κύκλου στην οθόνη ενώ ταυτόχρονα ικανοποιούν τις απαιτήσεις της εκφώνησης. Τα αποτελέσματα επιβεβαιώνουν την ακρίβεια και την αποτελεσματικότητα της μεθόδου που χρησιμοποιήθηκε.

Ευρετήριο Όρων

GLUI Η GLUI είναι μια βιβλιοθήκη διεπαφής χρήστη για γλώσσες προγραμματισμού όπως ο C++ που παρέχει ένα σύνολο ελέγχων για εφαρμογές OpenGL. Βασίζεται στο GLUT και παρέχει εύκολο τρόπο δημιουργίας παραθύρων διεπαφής χρήστη με στοιχεία όπως κουμπιά, πλαίσια επιλογής, περιστροφείς και άλλα. Οι ελέγχοι μπορούν να συνδεθούν με μεταβλητές και να προκαλούν call-backs, επιτρέποντας διαδραστική επεξεργασία και αλληλεπίδραση με το χρήστη. [Rademacher, 1999]. 1, 10

Listener Στον προγραμματισμό, listener είναι ένα στοιχείο ή συνάρτηση που περιμένει και αποκρίνεται σε συμβάντα ή μηνύματα που ενεργοποιούνται από άλλα μέρη ενός προγράμματος ή συστήματος. ”Ακούει” συγκεκριμένα συμβάντα και εκτελεί κώδικα ως απόκριση σε αυτά. Οι ακροατές χρησιμοποιούνται συνήθως σε παραδείγματα προγραμματισμού που βασίζονται σε συμβάντα για να επιτρέψουν τη διαδραστικότητα και να χειριστούν αποτελεσματικά τις εισροές των χρηστών ή τα συμβάντα του

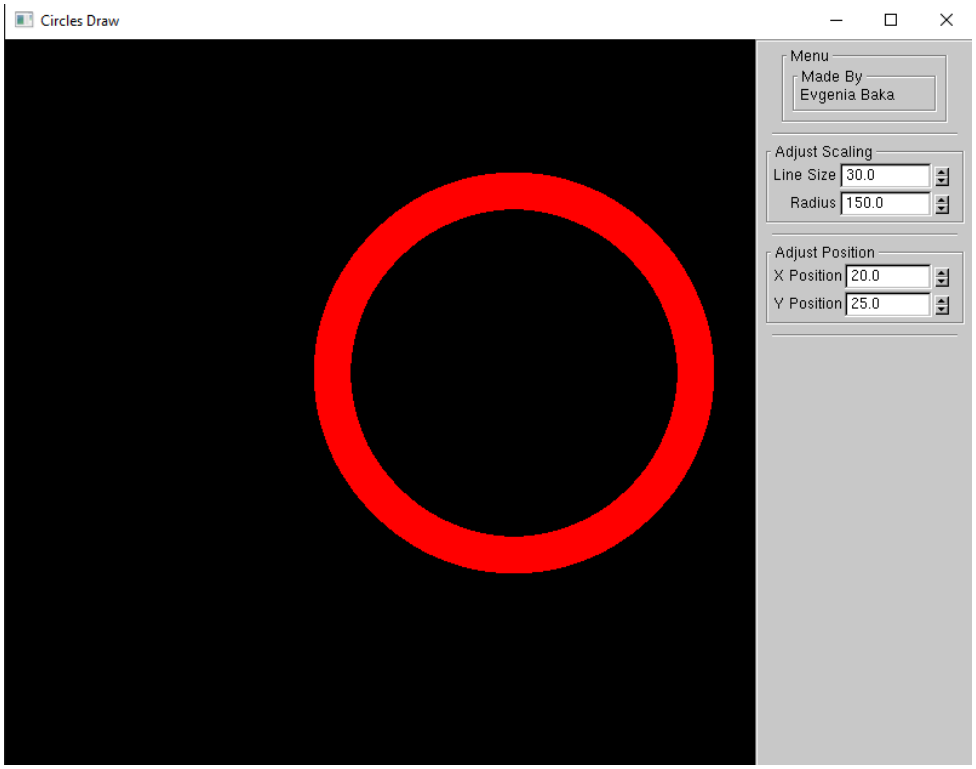

Figure 3: Στιγμιότυπο του εκτελέσιμου αρχείου

συστήματος. [Freeman et al., 2004]. 7

OpenGL Η OpenGL (Open Graphics Library) είναι μια προγραμματιστική διεπαφή γραφικών που χρησιμοποιείται ευρέως για την απεικόνιση 2D και 3D γραφικών σε εφαρμογές λογισμικού. Αρχικά αναπτύχθηκε από την εταιρεία Silicon Graphics Inc.[Hearn et al., 2014]. 2

Δακτύλιος Μια γεωμετρική μορφή που αποτελείται από δύο κύκλους που βρίσκονται στον ίδιο ευθύγραμμο άξονα αλλά έχουν διαφορετική ακτίνα. Ο δακτύλιος περικλείει την περιοχή μεταξύ των δύο κύκλων. Ο δακτύλιος έχει δύο ακτίνες, την εξωτερική ακτίνα (R) και την εσωτερική ακτίνα (r). [Haunsperger and Kennedy, 2006]. 10

Αναφορές - Βιβλιογραφία

[Foley et al., 1996] Foley, J., van Dam, A., Feiner, S., and Hughes, J. (1996). *Computer Graphics: Principles and Practice*. Addison-Wesley systems programming series. Addison-Wesley.

[Freeman et al., 2004] Freeman, E., Robson, E., Freeman, E., Sierra, K., and Bates, B. (2004). *Head First Design Patterns*. A brain-friendly guide. O'Reilly Media, Incorporated.

[GameDevelopment, 2018] GameDevelopment (2018). How can i use keybord keys like w, a, s and d in glut and c++?

[gamedev.net, 2010] gamedev.net (2010). How to detect multiple key presses in opengl?

[GeeksForGeeks, 2021] GeeksForGeeks (2021). Bresenham's circle drawing algorithm.

[Greenberg, 2016] Greenberg, I. (2016). *Processing: Creative Coding and Computational Art*. Apress.

[Haunsperger and Kennedy, 2006] Haunsperger, D. and Kennedy, S. (2006). *The Edge of the Universe: Celebrating Ten Years of Math Horizons*. MAA spectrum. Mathematical Association of America.

[Hearn et al., 2014] Hearn, D., Baker, P., and Carithers, W. (2014). *Computer Graphics with OpenGL*. PEARSON, 4th edition.

[Mukherjee and Jana, 2010] Mukherjee, D. and Jana, D. (2010). *Computer Graphics : Algorithms and Implementations*. PHI Learning.

[Rademacher, 1999] Rademacher, P. (1999). *GLUI: A GLUT-Based User Interface Library*, 2.0 edition.

[stack overflow, 2009] stack overflow (2009). Using opengl /glut how would i detect if two keys are held down at the same time?

[theasciicode.com,] theasciicode.com. Ascii table , ascii codes.

[Whitrow, 2008] Whitrow, R. (2008). *OpenGL Graphics Through Applications*. Springer London.

[Γαρυφαλίδης, 2016] Γαρυφαλίδης (2016). Εφαπτομένη καμπύλης : Μια διαδρομή από τον Ευκλείδη στον Απολλώνιο και τον Αρχιμήδη. Master's thesis, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών - Πανεπιστήμιο Κύπρου.

[Δρακόπουλος, 2023] Δρακόπουλος, Β. (2023). Ηλεκτρονική Τάξη Μαθήματος Γραφικής Υπολογιστών.