UTM

UNIVERSITI TEKNOLOGI MALAYSIA

Assignment 2

Fundamental of Computer Graphics – SCSV 221

Name : Muhammad Amirul Fahmi bin Noor Anim

Matrics Num : B19EC0018

Sec Num : SCSV 2213(Sec 01)

TABLE OF CONTENTS

1. ACKNOWLEDGEMENT

I would like to thank my Fundamental Computer Graphics lecturer, Dr. Norhaida binti Mohd Suaib, for guiding us to finish our assignment in time. I would also like to thank my fellow colleagues who suggesting me many great ideas on starting this assignment. I would also like to thank to the youtuber and blogger who give me much information about OpenGL which helps me finish this assignment faster. Finally, I would also like to thank to the people who were participated in helping me finishing my assignment directly or indirectly.

2. INTRODUCTION

This task is given by our Fundamental of Computer Graphics lecturer, Dr. Norhaida bt Mohd Suaib, as a second assignment for us. In this short report, we will discuss on producing a mini 'Paint' software in Windows by using OpenGL. However, we do not have to make a complicated function in the software such as 'Fill' or more advanced functions. We are required to make 'Paint' software program that could at least generate certain output primitives. The program should generate a line based on user input by using DDA or Bresenham line algorithm. This program also should generate a circle based on Midpoint Circle algorithm and an ellipse based on Midpoint Ellipse algorithm. Student also can add extra function to the coding such as size or colour.

The objective of this assignment is to study available software/tool that can generate various output primitives. This assignment also makes us develop our own mini software that can generate basic output primitives such as line, circle and ellipse extended by the algorithms taught during lectures. This assignment also makes us prepare a suitable documentation and report.

3. STRUCTURE OF CODING

When running the program, the screen will be displayed with white colour. User are required to open a menu to use the function on the program. A sentence is displayed to guide user how to open the menu. The coding has a menu and submenu function, that is glutCreateMenu(). This function is used to make an option for the user to run the function according to user needs.

There are three menus in the program. The first one is 'Clear' that has a function to clear the buffer of the screen. The second menu is 'Draw' that has a function to draw shapes according to user need. The third one is 'Quit' that has a function to exit the program. The second menu has three submenu that is 'DDA Line algorithm', 'Midpoint Circle algorithm' and 'Midpoint Ellipse Algorithm'. The function of these submenus is to create a shape according to user need that is a line, circle or an ellipse.

To create a shape, the user needs to give input coordinates to the program. glutMousefunc() is used in this program to record the input given by user. There are three mouse functions in this program. Each function is implemented on each submenu with various number of inputs been recorded to the program. When the user put the required number of inputs to the program. A shape will be created based on user option.

4. NEW FUNCTION LEARNT IN OPENGL

glScissor()

This function is used to create a function that only happens in a rectangle-sized box provided by user. We used this function to change the instruction for the user based on the algorithm that the user want to draw.

glutCreateMenu()

This function is used for creating a menu or submenu in a program. We used this function to provide multiple options for the user to use the program.

5. CONCLUSION

There are many new things that I have learnt during finishing this assignment. I learn some new functions that I searched on the Internet. This makes me more aware that OpenGl has some helpful tricks to make a successful program. I also learn more on how to make an OpenGL program. This ease my work during making project and assignment on this course subject. Plus, this assignment makes me more interested on creating OpenGL program and maybe one day, OpenGL will be the source of my income in the future.

6. CODING

```
//MUHAMMAD AMIRUL FAHMI BIN NOOR ANIM
//B19EC0018

#include <windows.h>
#include <iostream>
#include <fstream>
#include <math.h>
#include <GL/glut.h>

float xstart, ystart, xend, yend;   // for dda line algorithm
int xcenter, ycenter, radius;       //for midpoint circle algorithm
int xCenter, yCenter, xradius, yradius;       //for midpoint ellipse
algorithm
static int pt = 0;
static char algorithm;
static int window;
static int menu_id;
static int submenu_id;
static int value = 0;

float distance(int xi, int yi, int xj, int yj) {        // to calculate the
distance between the first coordinate and second coordinate
    return sqrt(pow(xj - xi,2) + pow(yj - yi,2));
}

void setPixel(float x, float y) {
    glColor3f(1.0f, 0.0f, 0.0f);    //Color of the line
    glPointSize(2);                           //Size of the line
    glBegin(GL_POINTS);                   //Start drawing point primitive
    glVertex2f(x, y);
    glEnd();
}

void lineDDA(float x0, float y0, float xEnd, float yEnd)    //DDA Line
algorithm
{
    float dx = xEnd - x0, dy = yEnd - y0, steps, k;
```

```
    float xIncrement, yIncrement, x = x0, y = y0;
    if (fabs(dx) > fabs(dy))
        steps = fabs(dx);
    else
        steps = fabs(dy);
    xIncrement = float(dx) / float(steps);
    yIncrement = float(dy) / float(steps);
    setPixel(round(x), round(y));
    for (k = 0; k < steps; k++) {
        x += xIncrement;
        y += yIncrement;
        setPixel(round(x), round(y));
    }
}

void circleMidPoint(int xCenter, int yCenter, int radius)   //Circle
midpoint algorithm
{
    int x = 0;
    int y = radius;
    int p = 1 - radius;//5/4 is rounded to 1 for integer radius
    while (x < y) {// iterates to draw the first sector
        x++;
        if (p < 0)// the mid point is inside the circle
            p += 2 * x + 1;
        else {// the mid point is outside or at the circle
            y--;
            p += 2 * (x - y) + 1;
        }
        glBegin(GL_POINTS);
        glVertex2i(xCenter + x, yCenter + y);
        glVertex2i(xCenter - x, yCenter + y);
        glVertex2i(xCenter + x, yCenter - y);
        glVertex2i(xCenter - x, yCenter - y);
        glVertex2i(xCenter + y, yCenter + x);
        glVertex2i(xCenter - y, yCenter + x);
        glVertex2i(xCenter + y, yCenter - x);
        glVertex2i(xCenter - y, yCenter - x);
        glEnd();
    }
    // OPTIONAL:-> center of the circle
    glBegin(GL_POINTS);
    glVertex2i(xCenter, yCenter);
    glEnd();
}

void ellipsePlotPoints(int xCenter, int yCenter, int x, int y)
{
    setPixel(xCenter + x, yCenter + y);
    setPixel(xCenter - x, yCenter + y);
    setPixel(xCenter + x, yCenter - y);
    setPixel(xCenter - x, yCenter - y);
}
```

```
void ellipseMidpoint(int xCenter, int yCenter, int Rx, int Ry)  // midpoint
ellipse algorithm
{
    int Rx2 = Rx * Rx;
    int Ry2 = Ry * Ry;
    int twoRx2 = 2 * Rx2;
    int twoRy2 = 2 * Ry2;
    int p;
    int x = 0;
    int y = Ry;
    int px = 0;
    int py = twoRx2 * y;

    /* Plot the initial point in each quadrant. */
    ellipsePlotPoints(xCenter, yCenter, x, y);
    /* Region 1 */
    p = round(Ry2 - (Rx2 * Ry) + (0.25 * Rx2));
    while (px < py) {
        x++;
        px += twoRy2;
        if (p < 0)
            p += Ry2 + px;
        else {
            y--;
            py -= twoRx2;
            p += Ry2 + px - py;
        }
        ellipsePlotPoints(xCenter, yCenter, x, y);
    }
    /* Region 2 */
    p = round(Ry2 * (x + 0.5) * (x + 0.5) + Rx2 * (y - 1) * (y - 1) - Rx2 *
Ry2);
    while (y > 0) {
        y--;
        py -= twoRx2;
        if (p > 0)
            p += Rx2 - py;
        else {
            x++;
            px += twoRy2;
            p += Rx2 - py + px;
        }
        ellipsePlotPoints(xCenter, yCenter, x, y);
    }
}

void mouse3(int button, int state, int mousex, int mousey)      // mouse
input for midpoint ellipse algorithm
{
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
```

```
        if (pt == 0) {                      //to make sure if there is
existing point or not
            xCenter = mousex;                //make the first x-coordinate
the starting point
            yCenter = 480 - mousey;          //make the first y-coordinate
the starting point
            pt++;
            setPixel(xCenter, yCenter);
        }
        else if (pt == 1) {
            int x1 = mousex;                                //set x-
coordinate and y-coordinate as the x-radius
            int x2 = 480 - mousey;
            pt++;
            xradius = int(distance(xCenter, yCenter, x1, x2));  //calculate
the distance to change to radius x
        }
        else {
            int x11 = mousex;
//set x-coordinate and y-coordinate as the y-radius
            int x22 = 480 - mousey;
            yradius = int(distance(xCenter, yCenter, x11, x22));
//calculate the distance to change to radius y
            ellipseMidpoint(xCenter, yCenter, xradius, yradius);
//initiate dda line algorithm
        }
    }
    else if (button == GLUT_RIGHT_BUTTON && state ==
GLUT_DOWN)//undo(clear)the drawing
    {
        glClearColor(1, 1, 1, 0);
        glClear(GL_COLOR_BUFFER_BIT);
        while (pt > 0) {
            pt--;
        }
    }
}

void mouse2(int button, int state, int mousex, int mousey)      // mouse
input for midpoint circle algorithm
{
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        if (pt == 0) {                      //to make sure if there is
existing point or not

            xcenter = mousex;
            ycenter = 480 - mousey;
            pt++;
            setPixel(xcenter, ycenter);
        }

        else {
```

```
            int x1 = mousex;                                    //set x-
coordinate as the radius
            int x2 = 480 - mousey;
            radius = int (distance(xcenter, ycenter, x1, x2));
            circleMidPoint(xcenter, ycenter, radius);   //initiate circle
midpoint algorithm

        }
    }

    else if (button == GLUT_RIGHT_BUTTON && state ==
GLUT_DOWN)//undo(clear)the drawing
    {
        glClearColor(1, 1, 1, 0);
        glClear(GL_COLOR_BUFFER_BIT);

        while (pt > 0) {
            pt--;
        }
    }

    glutPostRedisplay();
}

void mouse(int button, int state, int mousex, int mousey)       // mouse
input for dda line algorithm
{
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        if (pt == 0) {                      //to make sure if there is
existing point or not
            xstart = mousex;                //make the first x-coordinate
the starting point
            ystart = 480 - mousey;          //make the first y-coordinate
the starting point
            pt++;
            setPixel(xstart, ystart);
        }
        else {
            xend = mousex;                                  //make the first x-
coordinate the end point
            yend = 480 - mousey;                            //make the first y-
coordinate the end point
            lineDDA(xstart, ystart, xend, yend);        //initiate dda line
algorithm
        }
    }
    else if (button == GLUT_RIGHT_BUTTON && state ==
GLUT_DOWN)//undo(clear)the drawing
    {
        glClearColor(1, 1, 1, 0);
        glClear(GL_COLOR_BUFFER_BIT);
        while (pt > 0) {
```

```
            pt--;
        }
    }
}

void renderbitmap(float x, float y, void* font, char* string) {     //
rendering the word from the parameter
    char* c;
    glRasterPos2f(x, y);
    for (c = string; *c != '\0'; c++) {
        glutBitmapCharacter(font, *c);
    }
}

void introscreen() {                    // display instruction for user to
open menu
    glColor3f(0.f, 0.f, 0.f);
    char buf[100] = { 0 };
    sprintf_s(buf, "Press middle button mouse to open the menu");
    renderbitmap(5, 470, GLUT_BITMAP_8_BY_13, buf);
}

void instructionscreen() {              // display instruction for user to
create shape
    glColor3f(0.f, 0.f, 0.f);
    char buf[100] = { 0 };
    if (value == 2) {
        sprintf_s(buf, "Click the coordinate of first point then second
point.");
        renderbitmap(5, 455, GLUT_BITMAP_8_BY_13, buf);
    }
    else if (value == 3) {
        sprintf_s(buf, "Click the coordinate of center point then the
radius.");
        renderbitmap(5, 455, GLUT_BITMAP_8_BY_13, buf);
    }
    else if (value == 4) {
        sprintf_s(buf, "Click the coordinate of center point, radius of x
then radius of y.");
        renderbitmap(5, 455, GLUT_BITMAP_8_BY_13, buf);
    }
}

void menu(int num) {                    // To close the program if the user
click 'Quit' button
    if (num == 0) {
        glutDestroyWindow(window);
        exit(0);
    }
    else {
        value = num;
    }
    glutPostRedisplay();
```

```
}

void createMenu(void) {
    submenu_id = glutCreateMenu(menu);                      // creating
submenu for draw
    glutAddMenuEntry("DDA Line Algorithm", 2);
    glutAddMenuEntry("Midpoint Circle Algorithm", 3);
    glutAddMenuEntry("Midpoint Ellipse Algorithm", 4);
    menu_id = glutCreateMenu(menu);                         // creating
menu for the program
    glutAddMenuEntry("Clear", 1);
    glutAddSubMenu("Draw", submenu_id);                     // implementing
the submenu into 'Draw' menu
    glutAddMenuEntry("Quit", 0);
    glutAttachMenu(GLUT_MIDDLE_BUTTON);                     // user will
press middle button to display menu
}

void display(void) {
    glMatrixMode(GL_PROJECTION);            // sets the current matrix to
projection
    glLoadIdentity();                       //multiply the current matrix
by identity matrix
    gluOrtho2D(0.0, 640.0, 0.0, 480.0);     //sets the
parallel(orthographic) projection of the full frame buffer

    introscreen();
    if (value == 1) {
        glClearColor(1.0, 1.0, 1.0, 1.0);   // to clear buffer of the
screen
        glClear(GL_COLOR_BUFFER_BIT);
        introscreen();
    }
    else if (value == 2) {
        introscreen();
        glEnable(GL_SCISSOR_TEST);          //to clear buffer of certain
places only
        glScissor(5, 453, 580, 12);
        glClearColor(1.0, 1.0, 1.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT);
        glDisable(GL_SCISSOR_TEST);
        instructionscreen();
        glutMouseFunc(mouse);
    }
    else if (value == 3) {
        introscreen();
        glEnable(GL_SCISSOR_TEST);           //to clear buffer of certain
places only
        glScissor(5, 453, 580, 12);          // the coordinate and the size
of the box
        glClearColor(1.0, 1.0, 1.0, 1.0);    // the color of the box when
the buffers cleared
        glClear(GL_COLOR_BUFFER_BIT);
```

```
        glDisable(GL_SCISSOR_TEST);
        instructionscreen();
        glutMouseFunc(mouse2);
    }
    else if (value == 4) {
        introscreen();
        glEnable(GL_SCISSOR_TEST);           //to clear buffer of certain
places only
        glScissor(5, 453, 580, 12);
        glClearColor(1.0, 1.0, 1.0, 1.0);
        glClear(GL_COLOR_BUFFER_BIT);
        glDisable(GL_SCISSOR_TEST);
        instructionscreen();
        glutMouseFunc(mouse3);
    }
    glFlush();
}

int main(int argc, char** argv) {
    //M Amirul Fahmi
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGBA | GLUT_SINGLE);
    glutInitWindowSize(640, 480);
    glutInitWindowPosition(100, 100);
    window = glutCreateWindow("Menus and Submenus - Programming
Techniques");
    createMenu();
    glClearColor(1, 1, 1, 0); // sets the background color to white light
    glClear(GL_COLOR_BUFFER_BIT); // clears the frame buffer and set values
defined in glClearColor() function call
    glutDisplayFunc(display);
    glutMainLoop();
}
```

7. REFERENCES

1. Programming Techniques (2012), 'GLUT Tutorial – Creating Menus and Submenus in GLUT', Reviewed by: https://www.programming-techniques.com/2012/05/glut-tutorial-creating-menus-and-submenus-in-glut.html

2. SH Academy (2017), 'OpenGL mouse function example for beginners to draw an image with mouse', Reviewed by: https://youtu.be/79Zy-ATRwGE