UCS505

COMPUTER GRAPHICS

A Project Report on Analog Clock

Submitted to

Ms. Kudratdeep Aulakh

Submitted by

Jus Pratap Singh (101803014)

Amitoj Singh Sidhu (10183016)

COE - 2

Table of Contents

Introduction

Analog Clock

It is a tool used for reading the time of day. The shortest hand indicates the hour, a longer hand indicates the minutes, and the longest arm indicates the seconds. Some analog clocks have only two hands: the shorter hand indicating the hour and the longer hand indicating the minutes.

Through this project we will be visualising the running analog clock with the help of C++ and the OpenGL library, displaying all the three hands of the clock with the appropriate lengths and appropriate positions at a given time.

Computer Graphics Concepts

Bresenham's Circle Drawing Algorithm

It is a circle drawing algorithm that selects the nearest pixel position to complete the arc. The unique part of this algorithm is that is uses only integer arithmetic which makes it significantly faster than other algorithms using floating point arithmetic in classical processors. The strategy that is followed in this algorithm is to select the pixel which has the least distance with the true circle boundary and with then keep calculating the successive points on the circle. We have used this algorithm to draw the circle for the clock.

Flood Fill Algorithm

It is mainly used to determine a bounded area connected to a given node in a multi-dimensional array. It is a close resemblance to the bucket tool in paint programs. It has been exercised to fill the interior of the clock. Given a 2D screen, location of a pixel in the screen and a color, we will replace color of the given pixel and all adjacent same coloured pixels with the given color.

Bresenham's Line lgorithm

It is a line drawing algorithm that determines the points of an n-dimensional raster that should be selected in order to form a close approximation to a straight line between two points. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly. It has been applied to draw the hands of the clock.

User Defined Functions Description

- void circle_plot (int x, int y)
  Plotting the 8 way symmetry of points as a base for implementing the bresenhams circle algo.

- void draw_circle (int r)
  Drawing the circle using bresenhams circle algorithm for the basic circular clock layout.

- void fill_circle()
  Using the flood fill algo to fill in the clock with black color.

- void add_numbers()
  Adding the numbers to the clock at proper distance along the perimeter of the clock.

- void draw_line(int x, int y, int smh)
  Drawing the 3 hands of the clock using Bresenhams line's algo.

- void show_time()
  Calculating the time using the systems local time and some trigonometric functions and displaying the three hands on the clock.

- void display(void)
  Displaying all the elements of the clock in a window using all the above functions at one time.

- void Timer(int value)
  Displaying the clock window after every 1000 milliseconds. This is the main function that change the time in the clock window.

Code

```
#define _CRT_SECURE_NO_WARNINGS
#include <GL/glut.h>
#include <freeglut.h>
#include <math.h>
#include<iostream>

using namespace std;


void circle_plot(int x, int y)
{
	glPointSize(3);
	glBegin(GL_POINTS);
	glColor3f(0, 0, 0);
	glVertex2i(x, y);
	glVertex2i(y, x);
	glVertex2i(y, -x);
	glVertex2i(x, -y);
	glVertex2i(-x, -y);
	glVertex2i(-y, -x);
	glVertex2i(-y, x);
	glVertex2i(-x, y);
	glEnd();
}

void draw_circle(int r)
{
	int d, x, y;
	x = 0;
	y = r;
	d = 3 - 2 * r;
	while (x <= y) {
		circle_plot(x, y);
		if (d < 0)
			d += 4 * x + 6;
		else {
			d += 4 * (x - y) + 10;
			y--;
		}
		x++;
	}
}

void fill_circle() {
	int temp;
	glBegin(GL_POINTS);
	glColor3f(0, 0, 0);
	for (int i = 0; i <= 250; i++) {
		temp = pow(pow(250, 2) - pow(i, 2), .5);
		for (int j = -temp; j <= temp; j++) {
			glVertex2i(j, i);
			glVertex2i(j, -i);
```

```
                }
        }
        glEnd();
}

void add_numbers() {
        glColor3f(1, 1, 0);
        glRasterPos2f(-10, 220);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"12");
        glRasterPos2f(-5, -230);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"6");

        glRasterPos2f(225, 0);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"3");
        glRasterPos2f(-230, 0);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"9");

        glRasterPos2f(190, 105);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"2");
        glRasterPos2f(105, 190);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"1");

        glRasterPos2f(195, -105);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"4");
        glRasterPos2f(120, -190);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"5");

        glRasterPos2f(-200, 105);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"10");
        glRasterPos2f(-115, 190);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"11");

        glRasterPos2f(-210, -105);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"8");
        glRasterPos2f(-130, -190);
        glutBitmapString(GLUT_BITMAP_TIMES_ROMAN_24, (const unsigned char*)"7");
}

void draw_line(int x, int y, int smh) {
        int dx, dy, i, e;
        int incx, incy, inc1, inc2;

        dx = x;
        dy = y;

        if (dx < 0) dx = -dx;
        if (dy < 0) dy = -dy;
        incx = 1;
        if (x < 0) incx = -1;
        incy = 1;
        if (y < 0) incy = -1;
        x = 0; y = 0;
        glBegin(GL_POINTS);
        if (smh == 1) glColor3f(1, 0, 0);
        if (smh == 2) glColor3f(0, 1, 0);
```

```
	if (smh == 3) glColor3f(0, 0, 1);
	if (dx > dy) {
		glVertex2i(x, y);
		e = 2 * dy - dx;
		inc1 = 2 * (dy - dx);
		inc2 = 2 * dy;
		for (i = 0; i < dx; i++) {
			if (e >= 0) {
				y += incy;
				e += inc1;
			}
			else
				e += inc2;
			x += incx;
			glVertex2i(x, y);
		}

	}
	else {
		glVertex2i(x, y);
		e = 2 * dx - dy;
		inc1 = 2 * (dx - dy);
		inc2 = 2 * dx;
		for (i = 0; i < dy; i++) {
			if (e >= 0) {
				x += incx;
				e += inc1;
			}
			else
				e += inc2;
			y += incy;
			glVertex2i(x, y);
		}
	}
	glEnd();
}

void show_time() {
	time_t t = time(0);
	float pie = 3.14 / 180;
	struct tm* time = localtime(&t);
	draw_line(190 * sin(pie * time->tm_sec * 6), 190 * cos(pie * time->tm_sec * 6), 1);
	draw_line(165 * sin(pie * time->tm_min * 6), 165 * cos(pie * time->tm_min * 6), 2);
	draw_line(110 * sin(pie * (time->tm_hour * 30 + time->tm_min / 2)), 110 * cos(pie * (time-
>tm_hour * 30 + time->tm_min / 2)), 3);
}

void display(void) {
	glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
	glClear(GL_COLOR_BUFFER_BIT);
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-300, 300, -300, 300);
	draw_circle(250);
	fill_circle();
```

```
        add_numbers();
        show_time();
        glFlush();
}

void Timer(int value) {
        glutPostRedisplay();      // Post re-paint request to activate display()
        glutTimerFunc(1000, Timer, 0); // next Timer call milliseconds later
}

void main(int argc, char** argv) {
        glutInit(&argc, argv); //Initialize glut
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        //Set the mode of the window-depth buffer, single buffer, color model
        glutInitWindowPosition(100, 100); //Set the position of the window
        glutInitWindowSize(600, 600); //Set the size of the window
        glutCreateWindow("Analog Clock"); //Create window and assign title
        glutDisplayFunc(display); //Call display to transfer the drawing to the window. The prototype
of this function is glutDisplayFunc(void)
        glutTimerFunc(0, Timer, 0);
        glutMainLoop(); //Enter the loop and wait
}
```

Output