UTM

UNIVERSITI TEKNOLOGI MALAYSIA

Assignment 1b

Greeting Card Festival

MUHAMMAD AMIRUL FAHMI | SCSV 2213 (Sec 01) | March 04 2020

TABLE OF CONTENTS

1. INTRODUCTION

This is the second short report for assignment in Fundamental of Computer Graphics subject (SCSV 2213 – Sec 01). In this report, we will discuss on how to make a greeting card for festival using OpenGL. For section 2, I will highlight on what tools or functions that I use to make the greeting cards. As for section 3, I will show on the coding including output.

2. HOW TO MAKE GREETING CARDS

For this assignment, I use two functions to complete the assignment. The first one that I often used is GL_QUADS command. This function is used for making a square or rectangle or similar to it as long as it has four vertices. I also used GL_POLYGON functions to make a triangular shape. I used these functions to make a simple Islamic geometric pattern on the card.

Next, I use GLUT type functions to display a text in the card. Unfortunately, by using GLUT functions, my access to display text with various font are limited as GLUT only has certain font to choose. One of the example fonts available is GLUT_BITMAP_TIMES_ROMAN_24.

3. THE CODING AND THE OUTPUT

Coding:

```
#include <GL/glut.h> // Include the GLUT header file
#include <fstream>
#include <stdlib.h>
#include <iostream>

using namespace std;

void renderbitmap(float x, float y, void* font, char* string) {

	char* c;
	glRasterPos2f(x, y);
	for (c = string; *c != '\0'; c++) {
		glutBitmapCharacter(font, *c);
	}
}

void reshape(int width, int height) {

	glViewport(0, 0, width, height);
}

void introscreen() {
	glColor3f(1.f, 1.f, 0.3f);
```

```
	char buf[100] = { 0 };
	sprintf_s(buf, "SELAMAT HARI RAYA");
	renderbitmap(-0.5, 0.0, GLUT_BITMAP_TIMES_ROMAN_24, buf);
	sprintf_s(buf, "MAAF ZAHIR DAN BATIN");
	renderbitmap(-0.4, -0.1, GLUT_BITMAP_HELVETICA_18, buf);
}

void display(void) {
	glClearColor(0.0f, 0.6f, 0.6f, 1.0f); // Clear the background of our window to dark turqoise
	glClear(GL_COLOR_BUFFER_BIT); //Clear the colour buffer (more buffers later on)
	glLoadIdentity(); // Load the Identity Matrix to reset our drawing locations
	glOrtho(-1.0, 1.0, -1.0, 1.0, 0.0, 1.0);

	glBegin(GL_QUADS);
	glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // left upper
	glVertex2f(-1.f, 0.1f);
	glVertex2f(-0.9f, 0.2f);
	glVertex2f(-0.9f, 1.f);
	glVertex2f(-1.f, 1.f);
	glEnd();

	glBegin(GL_QUADS);				// left upper
	glColor4f(1.0f, 0.9f, 0.9f, 0.5f);
	glVertex2f(-1.f, 1.f);
	glVertex2f(-1.f, 0.9f);		// Define vertices in counter-clockwise (CCW) order
	glVertex2f(-0.2f, 0.9f);	//  so that the normal (front-face) is facing you
	glVertex2f(-0.1f, 1.f);
	glEnd();

	glBegin(GL_QUADS);				//Left lower
	glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
	glVertex2f(-1.f, -1.f);
	glVertex2f(-0.1f, -1.f);
	glVertex2f(-0.2f, -0.9f);
	glVertex2f(-1.f, -0.9f);
	glEnd();

	glBegin(GL_QUADS);				// Left lower
	glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
	glVertex2f(-1.f, -1.f);
	glVertex2f(-0.9f, -1.f);
	glVertex2f(-0.9f, -0.2f);
	glVertex2f(-1.f, -0.1f);
	glEnd();

	glBegin(GL_QUADS);				// Right upper
```

```
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(0.1f, 1.f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(0.2f, 0.9f);      //  so that the normal (front-
face) is facing you
      glVertex2f(1.f, 0.9f);
      glVertex2f(1.f, 1.f);
      glEnd();

      glBegin(GL_QUADS);                //Right upper
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(0.9f, 1.f);
      glVertex2f(0.9f, 0.2f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(1.f, 0.1f);      //  so that the normal (front-face)
is facing you
      glVertex2f(1.f, 1.f);
      glEnd();

      glBegin(GL_QUADS);                // Right lower
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(0.1f, -1.f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(1.f, -1.f);
      glVertex2f(1.f, -0.9f);
      glVertex2f(0.2f, -0.9f);
      glEnd();

      glBegin(GL_QUADS);                //Right lower
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(0.9f, -1.f);
      glVertex2f(1.f, -1.f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(1.f, -0.1f);      //  so that the normal (front-
face) is facing you
      glVertex2f(0.9f, -0.2f);
      glEnd();

      glBegin(GL_QUADS);                // Each set of 4 vertices form
a quad
      glColor4f(0.0f, 0.6f, 0.6f, 1.0f); // Turqoise
      glVertex2f(0.0f, 1.f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(-1.f, 0.0f);      //  so that the normal (front-
face) is facing you
      glVertex2f(0.0f, -1.f);
      glVertex2f(1.f, 0.0f);
      glEnd();

      glBegin(GL_QUADS);                // Each set of 4 vertices form
a quad
```

```
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(-0.8f, 0.8f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(-0.8f, -0.8f);      //  so that the normal (front-
face) is facing you
      glVertex2f(0.8f, -0.8f);
      glVertex2f(0.8f, 0.8f);
      glEnd();

      glBegin(GL_QUADS);                // Each set of 4 vertices form
a quad
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(0.0f, 1.f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(-1.f, 0.0f);      //  so that the normal (front-
face) is facing you
      glVertex2f(0.0f, -1.f);
      glVertex2f(1.f, 0.0f);
      glEnd();

      glBegin(GL_QUADS);                // Each set of 4 vertices form
a quad
      glColor4f(0.0f, 0.6f, 0.6f, 1.0f); // Turqoise
      glVertex2f(-0.7f, 0.7f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(-0.7f, -0.7f);      //  so that the normal (front-
face) is facing you
      glVertex2f(0.7f, -0.7f);
      glVertex2f(0.7f, 0.7f);
      glEnd();

      glBegin(GL_QUADS);                // Each set of 4 vertices form
a quad
      glColor4f(0.0f, 0.6f, 0.6f, 1.0f); // Turqoise
      glVertex2f(0.0f, 0.9f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(-0.9f, 0.0f);      //  so that the normal (front-
face) is facing you
      glVertex2f(0.0f, -0.9f);
      glVertex2f(0.9f, 0.0f);
      glEnd();

      glBegin(GL_POLYGON);                // Each set of 3 vertices
form a polygon
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(-0.6f, 0.6f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(-0.6f, 0.3f);      //  so that the normal (front-
face) is facing you
      glVertex2f(-0.3f, 0.6f);
      glEnd();
```

```
      glBegin(GL_POLYGON);                // Each set of 3 vertices
form a polygon
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(0.6f, 0.6f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(0.6f, 0.3f);      //  so that the normal (front-
face) is facing you
      glVertex2f(0.3f, 0.6f);
      glEnd();

      glBegin(GL_POLYGON);                // Each set of 3 vertices
form a polygon
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(0.6f, -0.6f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(0.6f, -0.3f);      //  so that the normal (front-
face) is facing you
      glVertex2f(0.3f, -0.6f);
      glEnd();

      glBegin(GL_POLYGON);                // Each set of 3 vertices
form a polygon
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(0.6f, -0.6f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(0.6f, -0.3f);      //  so that the normal (front-
face) is facing you
      glVertex2f(0.3f, -0.6f);
      glEnd();

      glBegin(GL_POLYGON);                // Each set of 3 vertices
form a polygon
      glColor4f(1.0f, 0.9f, 0.9f, 0.5f); // White
      glVertex2f(-0.6f, -0.6f);      // Define vertices in counter-
clockwise (CCW) order
      glVertex2f(-0.6f, -0.3f);      //  so that the normal (front-
face) is facing you
      glVertex2f(-0.3f, -0.6f);
      glEnd();

      introscreen();

      glFlush(); // Flush the OpenGL buffers to the window

      glutPostRedisplay();

}

int main(int argc, char** argv) {
      glutInit(&argc, argv); // Initialize GLUT
```

```
	glutInitDisplayMode(GLUT_SINGLE); // Set up a basic display
buffer (only single buffered for now)
	glutInitWindowSize(500, 500); // Set the width and height of
the window
	glutInitWindowPosition(100, 100); // Set the position of the
window
	glutCreateWindow("Lab 1"); // Set the title for the window

	glutDisplayFunc(display); // Tell GLUT to use the method
"display" for rendering
	glutReshapeFunc(reshape);

	glutMainLoop(); // Enter GLUT's main loop
}
```

Output:

4. CONCLUSION

There are many things that I learn from completing this assignment. I learn how to make a 2d shape or object using OpenGL. I also learned how to change the colour of the shape or the screen more effectively. Finally, I learned how to make a better design using OpenGL compared to the first time I used it.

5. ACKNOWLEDGEMENT

I would like to thank my Fundamental Computer Graphics lecturer, Dr. Norhaida binti Mohd Suaib, for guiding me to finish my 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 people who were participated in helping me finishing my assignment directly or indirectly.