Nama: Nur Cahyo Ihsan Prastyawan

Kelas: 2 D4 TK B

NRP: 2210181048

**WORKSHOP GRAFIKA KOMPUTER**

**PRIMITVE DRAWING**

1. Program 1

Code:

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <stdio.h>

int main() {

GLFWwindow\* window;

float points[] = {

0.0f, 0.5f, 0.0f,

0.5f, -0.5f, 0.0f,

-0.5f, -0.5f, 0.0f

};

if (!glfwInit()) return -1;

window = glfwCreateWindow(640, 480, "Primitive Drawing", NULL, NULL);

if (!window) {

glfwTerminate();

return -1;

}

glfwMakeContextCurrent(window);

glewExperimental = GL\_TRUE;

glewInit();

const GLubyte\* renderer = glGetString(GL\_RENDERER);

const GLubyte\* version = glGetString(GL\_VERSION);

printf("Renderer : %s\n", renderer);

printf("OpenGL version supported %s\n", version);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

GLuint vbo;

glGenBuffers(1, &vbo);

glBindBuffer(GL\_ARRAY\_BUFFER, vbo);

glBufferData(GL\_ARRAY\_BUFFER, sizeof(points), points, GL\_STATIC\_DRAW);

GLuint vao = 0;

glGenVertexArrays(1, &vao);

glBindVertexArray(vao);

glEnableVertexAttribArray(0);

glBindBuffer(GL\_ARRAY\_BUFFER, vbo);

glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, 0, NULL);

const char\* vertex\_shader = "#version 400\n"

"in vec3 vp;"

"void main(){"

" gl\_Position = vec4(vp, 1.0);"

"}";

const char\* fragment\_shader =

"#version 400\n"

"out vec4 frag\_colour;"

"void main(){"

" frag\_colour = vec4(0.5, 0.25, 0.1, 1);"

"}";

GLuint vs = glCreateShader(GL\_VERTEX\_SHADER);

glShaderSource(vs, 1, &vertex\_shader, NULL);

glCompileShader(vs);

GLuint fs = glCreateShader(GL\_FRAGMENT\_SHADER);

glShaderSource(fs, 1, &fragment\_shader, NULL);

glCompileShader(fs);

GLuint shader\_programme = glCreateProgram();

glAttachShader(shader\_programme, fs);

glAttachShader(shader\_programme, vs);

glLinkProgram(shader\_programme);

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glUseProgram(shader\_programme);

glBindVertexArray(vao);

glDrawArrays(GL\_TRIANGLES, 0, 3);

glPointSize(20.0);

glfwPollEvents();

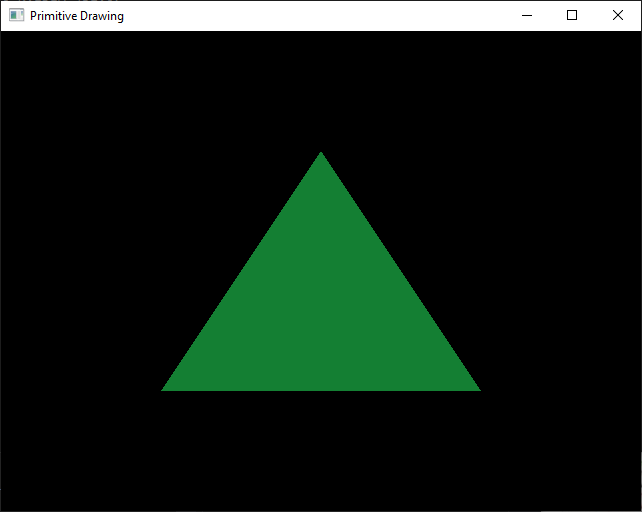
glfwSwapBuffers(window);

}

glfwTerminate();

}

Hasil:



2. Program 2

Code:

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <stdio.h>

int main() {

GLFWwindow\* window;

float points[] = {

0.0f, 0.5f, 0.0f,

0.5f, -0.5f, 0.0f,

-0.5f, -0.5f, 0.0f

};

if (!glfwInit()) {

fprintf(stderr, "ERROR: could not start GLFW3\n");

return -1;

}

window = glfwCreateWindow(640, 480, "Primitive Drawing", NULL, NULL);

if (!window) {

fprintf(stderr, "ERROR: could not open window with GLFW3\n");

glfwTerminate();

return -1;

}

glfwMakeContextCurrent(window);

glewExperimental = GL\_TRUE;

glewInit();

const GLubyte\* renderer = glGetString(GL\_RENDERER);

const GLubyte\* version = glGetString(GL\_VERSION);

printf("Renderer : %s\n", renderer);

printf("OpenGL version supported %s\n", version);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

GLuint vbo;

glGenBuffers(1, &vbo);

glBindBuffer(GL\_ARRAY\_BUFFER, vbo);

glBufferData(GL\_ARRAY\_BUFFER, sizeof(points), points, GL\_STATIC\_DRAW);

GLuint vao = 0;

glGenVertexArrays(1, &vao);

glBindVertexArray(vao);

glEnableVertexAttribArray(0);

glBindBuffer(GL\_ARRAY\_BUFFER, vbo);

glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, 0, NULL);

const char\* vertex\_shader = "#version 400\n"

"in vec3 vp;"

"void main(){"

" gl\_Position = vec4(vp, 1.0);"

"}";

const char\* fragment\_shader =

"#version 400\n"

"out vec4 frag\_colour;"

"void main(){"

" frag\_colour = vec4(161, 155, 0, 1);"

"}";

GLuint vs = glCreateShader(GL\_VERTEX\_SHADER);

glShaderSource(vs, 1, &vertex\_shader, NULL);

glCompileShader(vs);

GLuint fs = glCreateShader(GL\_FRAGMENT\_SHADER);

glShaderSource(fs, 1, &fragment\_shader, NULL);

glCompileShader(fs);

GLuint shader\_programme = glCreateProgram();

glAttachShader(shader\_programme, fs);

glAttachShader(shader\_programme, vs);

glLinkProgram(shader\_programme);

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glUseProgram(shader\_programme);

glBindVertexArray(vao);

glDrawArrays(GL\_POINTS, 0, 3);

glPointSize(15.0);

glfwPollEvents();

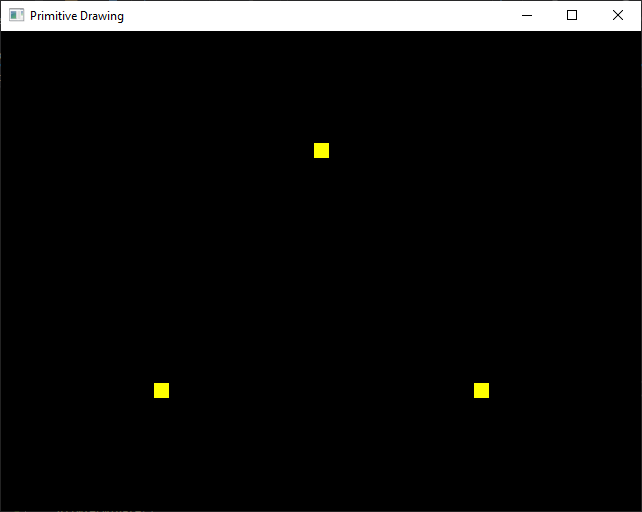
glfwSwapBuffers(window);

}

glfwTerminate();

}

Hasil



3. Program 3

Code:

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <stdio.h>

int main() {

GLFWwindow\* window;

float points[] = {

0.5f, 0.5f, 0.0f,

0.5f, -0.5f, 0.0f,

-0.5f, -0.5f, 0.0f,

-0.5f, 0.5f, 0.0f,

0.5f, 0.5f, 0.0f,

-0.5f, 0.5f, 0.0f,

0.5f, -0.5f, 0.0f,

-0.5f, -0.5f, 0.0f

};

if (!glfwInit()) return -1;

window = glfwCreateWindow(640, 480, "Primitive Drawing", NULL, NULL);

if (!window) {

glfwTerminate();

return -1;

}

glfwMakeContextCurrent(window);

glewExperimental = GL\_TRUE;

glewInit();

const GLubyte\* renderer = glGetString(GL\_RENDERER);

const GLubyte\* version = glGetString(GL\_VERSION);

printf("Renderer : %s\n", renderer);

printf("OpenGL version supported %s\n", version);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

GLuint vbo;

glGenBuffers(1, &vbo);

glBindBuffer(GL\_ARRAY\_BUFFER, vbo);

glBufferData(GL\_ARRAY\_BUFFER, sizeof(points), points, GL\_STATIC\_DRAW);

GLuint vao = 0;

glGenVertexArrays(1, &vao);

glBindVertexArray(vao);

glEnableVertexAttribArray(0);

glBindBuffer(GL\_ARRAY\_BUFFER, vbo);

glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, 0, NULL);

const char\* vertex\_shader = "#version 400\n"

"in vec3 vp;"

"void main(){"

" gl\_Position = vec4(vp, 1.0);"

"}";

const char\* fragment\_shader =

"#version 400\n"

"out vec4 frag\_colour;"

"void main(){"

" frag\_colour = vec4(0.5, 0.25, 0.1, 1);"

"}";

GLuint vs = glCreateShader(GL\_VERTEX\_SHADER);

glShaderSource(vs, 1, &vertex\_shader, NULL);

glCompileShader(vs);

GLuint fs = glCreateShader(GL\_FRAGMENT\_SHADER);

glShaderSource(fs, 1, &fragment\_shader, NULL);

glCompileShader(fs);

GLuint shader\_programme = glCreateProgram();

glAttachShader(shader\_programme, fs);

glAttachShader(shader\_programme, vs);

glLinkProgram(shader\_programme);

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glUseProgram(shader\_programme);

glBindVertexArray(vao);

glDrawArrays(GL\_LINE, 0, 4);

glfwPollEvents();

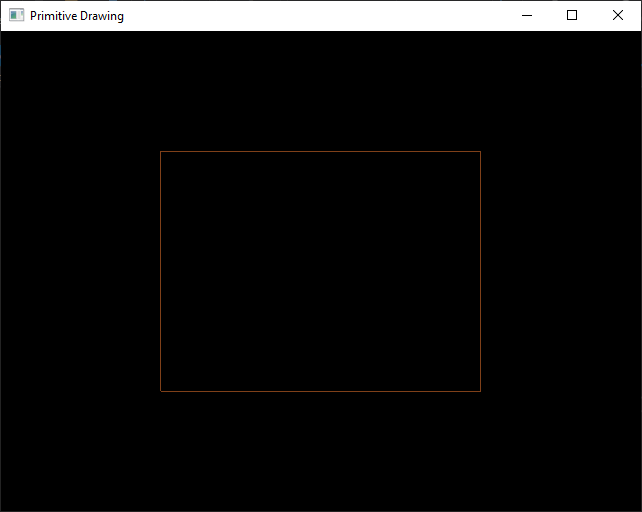
glfwSwapBuffers(window);

}

glfwTerminate();

}

Hasil



4. Program 4

Code:

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <stdio.h>

int main() {

GLFWwindow\* window;

float points[] = {

0.5f, 0.5f, 0.0f,

0.5f, -0.5f, 0.0f,

-0.5f, -0.5f, 0.0f,

-0.5f, 0.5f, 0.0f,

0.5f, 0.5f, 0.0f,

-0.5f, 0.5f, 0.0f,

0.5f, -0.5f, 0.0f,

-0.5f, -0.5f, 0.0f

};

if (!glfwInit()) return -1;

window = glfwCreateWindow(640, 480, "Primitive Drawing", NULL, NULL);

if (!window) {

glfwTerminate();

return -1;

}

glfwMakeContextCurrent(window);

glewExperimental = GL\_TRUE;

glewInit();

const GLubyte\* renderer = glGetString(GL\_RENDERER);

const GLubyte\* version = glGetString(GL\_VERSION);

printf("Renderer : %s\n", renderer);

printf("OpenGL version supported %s\n", version);

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

GLuint vbo;

glGenBuffers(1, &vbo);

glBindBuffer(GL\_ARRAY\_BUFFER, vbo);

glBufferData(GL\_ARRAY\_BUFFER, sizeof(points), points, GL\_STATIC\_DRAW);

GLuint vao = 0;

glGenVertexArrays(1, &vao);

glBindVertexArray(vao);

glEnableVertexAttribArray(0);

glBindBuffer(GL\_ARRAY\_BUFFER, vbo);

glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, 0, NULL);

const char\* vertex\_shader = "#version 400\n"

"in vec3 vp;"

"void main(){"

" gl\_Position = vec4(vp, 1.0);"

"}";

const char\* fragment\_shader =

"#version 400\n"

"out vec4 frag\_colour;"

"void main(){"

" frag\_colour = vec4(0.5, 0.25, 0.1, 1);"

"}";

GLuint vs = glCreateShader(GL\_VERTEX\_SHADER);

glShaderSource(vs, 1, &vertex\_shader, NULL);

glCompileShader(vs);

GLuint fs = glCreateShader(GL\_FRAGMENT\_SHADER);

glShaderSource(fs, 1, &fragment\_shader, NULL);

glCompileShader(fs);

GLuint shader\_programme = glCreateProgram();

glAttachShader(shader\_programme, fs);

glAttachShader(shader\_programme, vs);

glLinkProgram(shader\_programme);

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glUseProgram(shader\_programme);

glBindVertexArray(vao);

glDrawArrays(GL\_LINE\_LOOP, 0, 4);

glfwPollEvents();

glfwSwapBuffers(window);

}

glfwTerminate();

}

