Nama: Nur Cahyo Ihsan Prastyawan

Kelas: 2 D4 TK B

NRP: 2210181048

**WORKSHOP GRAFIKA KOMPUTER**

**INPUT HANDLER FUNCTION**

1. Program 1

Code:

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <math.h>

#include <iostream>

#define SCREEN\_WIDTH 560

#define SCREEN\_HEIGHT 340

void keyCallback(GLFWwindow\* window, int key, int scancode, int action, int mods) {

std::cout << key << std::endl;

if (key == GLFW\_KEY\_SPACE && action == GLFW\_PRESS) {

std::cout << "Space Key Pressed" << std::endl;

}

}

int main() {

GLFWwindow\* window;

if (!glfwInit()) {

return -1;

}

window = glfwCreateWindow(SCREEN\_WIDTH, SCREEN\_HEIGHT, "HELLO WORLD", NULL, NULL);

int screenWidth, screenHeight;

glfwGetFramebufferSize(window, &screenWidth, &screenHeight);

if (!window) {

glfwTerminate();

return -1;

}

glfwMakeContextCurrent(window);

glfwSetKeyCallback(window, keyCallback);

glfwSetInputMode(window, GLFW\_STICKY\_KEYS, 1);

glViewport(0.0f, 0.0f, screenWidth, screenHeight);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, SCREEN\_WIDTH, 0, SCREEN\_HEIGHT, 0, 1);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glfwSwapBuffers(window);

glfwPollEvents();

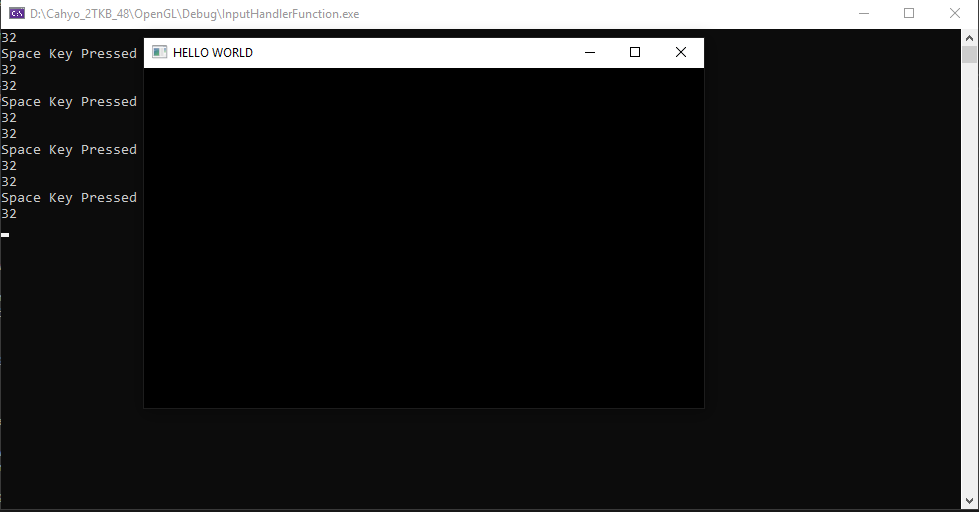
}

glfwTerminate();

return 0;

}

Hasil:



2. Program 2

Code:

void keyCallback(GLFWwindow\* window, int key, int scancode, int action, int mods) {

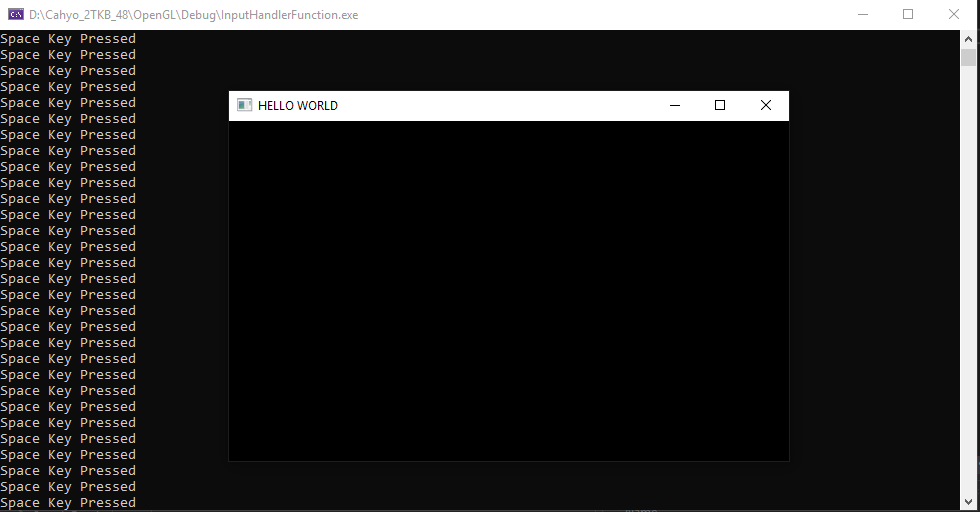
if (key == GLFW\_KEY\_SPACE && action == GLFW\_REPEAT) {

std::cout << "Space Key Pressed" << std::endl;

}

}

Hasil:



Analisa:

Perbedaan Program 2 dengan Program 1 yaitu pada Program 2 menggunakanvariabel action menggunakan GLFW\_REPEAT sementara pada Program 1 menggunakan GLFW\_PRESS.

GLFW\_PRESS akan menjalankan fungsi keyCallback jika tombol ditekan sekali lalu dilepaskan.

GLFW\_REPEAT akan menjalankan fungsi keyCallback jika tombol ditekan tanpa dilepaskan selama beberapa waktu.

3. Program 3

Code:

void keyCallback(GLFWwindow\* window, int key, int scancode, int action, int mods) {

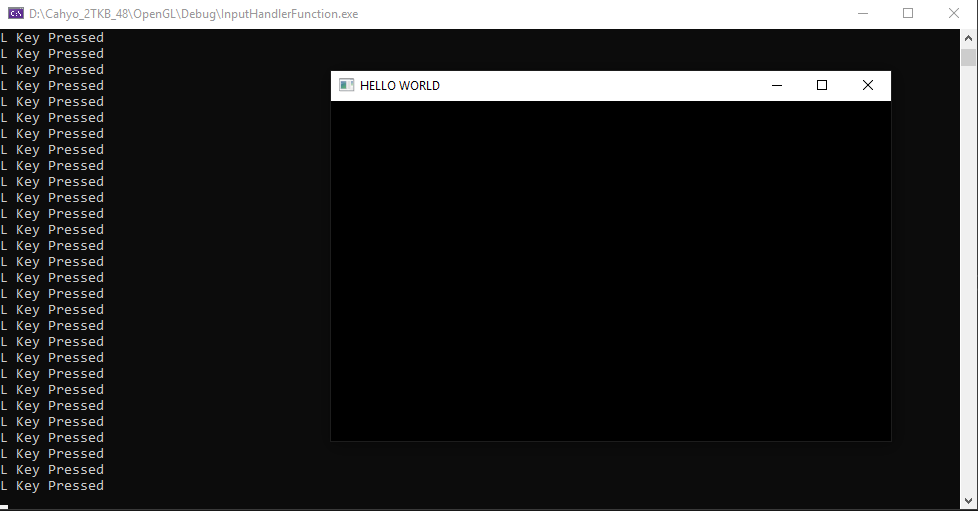
if (key == GLFW\_KEY\_L && action == GLFW\_REPEAT) {

std::cout << "L Key Pressed" << std::endl;

}

}

Hasil:



4. Program 4

Code:

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <math.h>

#include <iostream>

#define SCREEN\_WIDTH 640

#define SCREEN\_HEIGHT 480

static void cursorPositionCallback(GLFWwindow\* window, double xpos, double ypos);

void cursorEnterCallback(GLFWwindow\* window, int entered);

void mouseButtonCallback(GLFWwindow\* window, int button, int action, int mods);

void scrollCallback(GLFWwindow\* window, double xoffset, double yoffset);

int main() {

GLFWwindow\* window;

if (!glfwInit()) {

return -1;

}

window = glfwCreateWindow(SCREEN\_WIDTH, SCREEN\_HEIGHT, "HELLO WORLD", NULL, NULL);

glfwSetCursorPosCallback(window, cursorPositionCallback);

glfwSetInputMode(window, GLFW\_CURSOR, GLFW\_CURSOR\_NORMAL);

if (!window) {

glfwTerminate();

return -1;

}

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT);

double xpos, ypos;

glfwGetCursorPos(window, &xpos, &ypos);

glfwSwapBuffers(window);

glfwPollEvents();

}

glfwTerminate();

return 0;

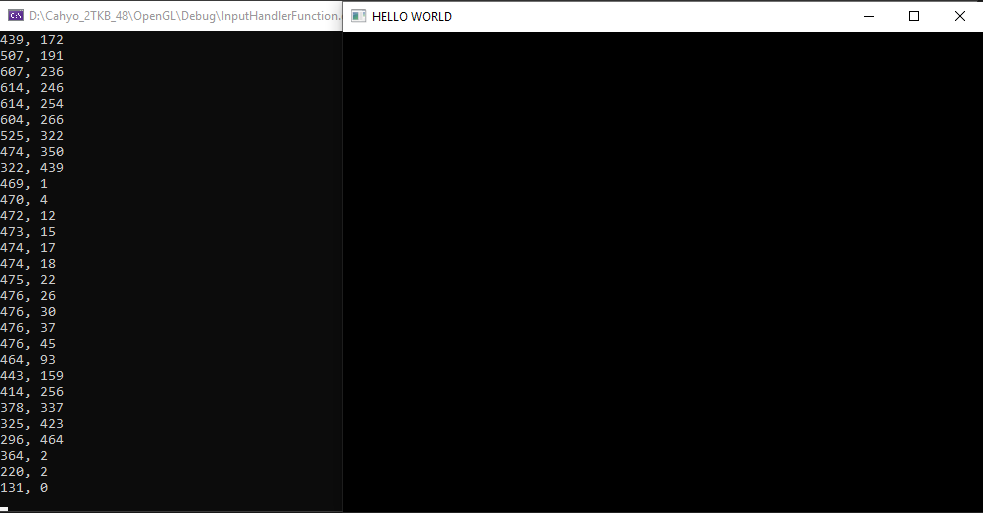
}

static void cursorPositionCallback(GLFWwindow\* window, double xpos, double ypos) {

std::cout << xpos << ", " << ypos << std::endl;

}

Hasil:



5. Program 5

Code:

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <math.h>

#include <iostream>

#define SCREEN\_WIDTH 640

#define SCREEN\_HEIGHT 480

static void cursorPositionCallback(GLFWwindow\* window, double xpos, double ypos);

void cursorEnterCallback(GLFWwindow\* window, int entered);

void mouseButtonCallback(GLFWwindow\* window, int button, int action, int mods);

void scrollCallback(GLFWwindow\* window, double xoffset, double yoffset);

int main() {

GLFWwindow\* window;

if (!glfwInit()) {

return -1;

}

window = glfwCreateWindow(SCREEN\_WIDTH, SCREEN\_HEIGHT, "HELLO WORLD", NULL, NULL);

//glfwSetCursorPosCallback(window, cursorPositionCallback);

//glfwSetInputMode(window, GLFW\_CURSOR, GLFW\_CURSOR\_NORMAL);

glfwSetCursorEnterCallback(window, cursorEnterCallback);

if (!window) {

glfwTerminate();

return -1;

}

unsigned char pixels[16 \* 16 \* 4];

memset(pixels, 0xff, sizeof(pixels));

GLFWimage image;

image.width = 16;

image.height = 16;

image.pixels = pixels;

GLFWcursor\* cursor = glfwCreateCursor(&image, 0, 0);

glfwSetCursor(window, cursor);

int screenWidth, screenHeight;

glfwGetFramebufferSize(window, &screenWidth, &screenHeight);

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glfwSwapBuffers(window);

glfwPollEvents();

}

glfwTerminate();

return 0;

}

static void cursorPositionCallback(GLFWwindow\* window, double xpos, double ypos) {

std::cout << xpos << ", " << ypos << std::endl;

}

void cursorEnterCallback(GLFWwindow\* window, int entered) {

if (entered) std::cout << "Entered window" << std::endl;

else std::cout << "Left window" << std::endl;

}

Hasil:



6. Program 6

Code:

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <math.h>

#include <iostream>

#define SCREEN\_WIDTH 640

#define SCREEN\_HEIGHT 480

static void cursorPositionCallback(GLFWwindow\* window, double xpos, double ypos);

void cursorEnterCallback(GLFWwindow\* window, int entered);

void mouseButtonCallback(GLFWwindow\* window, int button, int action, int mods);

void scrollCallback(GLFWwindow\* window, double xoffset, double yoffset);

int main() {

GLFWwindow\* window;

if (!glfwInit()) {

return -1;

}

window = glfwCreateWindow(SCREEN\_WIDTH, SCREEN\_HEIGHT, "HELLO WORLD", NULL, NULL);

//glfwSetCursorPosCallback(window, cursorPositionCallback);

//glfwSetInputMode(window, GLFW\_CURSOR, GLFW\_CURSOR\_NORMAL);

//glfwSetCursorEnterCallback(window, cursorEnterCallback);

glfwSetMouseButtonCallback(window, mouseButtonCallback);

if (!window) {

glfwTerminate();

return -1;

}

unsigned char pixels[16 \* 16 \* 4];

memset(pixels, 0xff, sizeof(pixels));

GLFWimage image;

image.width = 16;

image.height = 16;

image.pixels = pixels;

GLFWcursor\* cursor = glfwCreateCursor(&image, 0, 0);

glfwSetCursor(window, cursor);

int screenWidth, screenHeight;

glfwGetFramebufferSize(window, &screenWidth, &screenHeight);

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glfwSwapBuffers(window);

glfwPollEvents();

}

glfwTerminate();

return 0;

}

static void cursorPositionCallback(GLFWwindow\* window, double xpos, double ypos) {

std::cout << xpos << ", " << ypos << std::endl;

}

void cursorEnterCallback(GLFWwindow\* window, int entered) {

if (entered) std::cout << "Entered window" << std::endl;

else std::cout << "Left window" << std::endl;

}

void mouseButtonCallback(GLFWwindow\* window, int button, int action, int mods) {

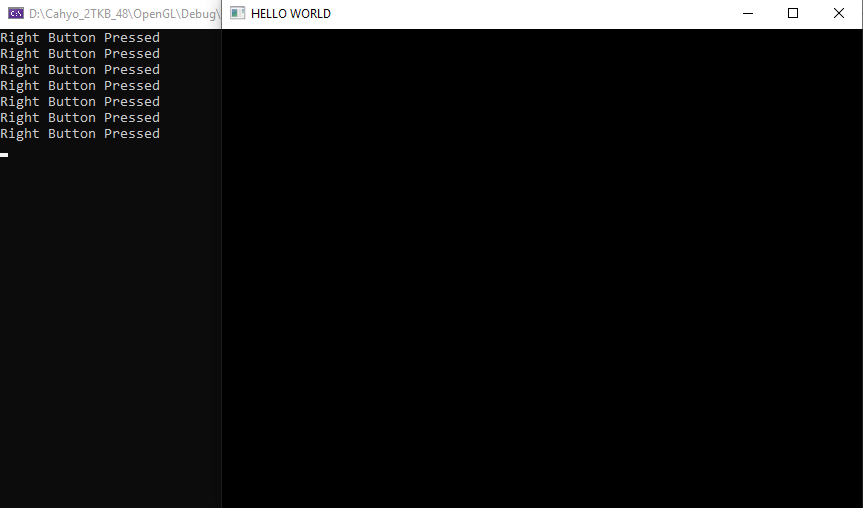
if (button == GLFW\_MOUSE\_BUTTON\_RIGHT && action == GLFW\_PRESS) {

std::cout << "Right Button Pressed" << std::endl;

}

}

Hasil:



Analisa:

Ketika di dalam window dan mouse diklik kanan, maka akan muncul kalimat “Right Button Pressed”. Callback yang digunakan memakai GLFW\_MOUSE\_BUTTON\_RIGHT dan GLFW\_PRESS sehingga ketika dua kondisi terpenuhi maka fungsi mouseButtonCallback dijalankan. Juga pada fungsi main diinisialisasi menggunakan glfwSetMouseButtonCallback() dengan parameter window yang dipakai dan fungsi callback nya.

7. Program 7

Code:

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <math.h>

#include <iostream>

#define SCREEN\_WIDTH 640

#define SCREEN\_HEIGHT 480

static void cursorPositionCallback(GLFWwindow\* window, double xpos, double ypos);

void cursorEnterCallback(GLFWwindow\* window, int entered);

void mouseButtonCallback(GLFWwindow\* window, int button, int action, int mods);

void scrollCallback(GLFWwindow\* window, double xoffset, double yoffset);

int main() {

GLFWwindow\* window;

if (!glfwInit()) {

return -1;

}

window = glfwCreateWindow(SCREEN\_WIDTH, SCREEN\_HEIGHT, "HELLO WORLD", NULL, NULL);

//glfwSetCursorPosCallback(window, cursorPositionCallback);

//glfwSetInputMode(window, GLFW\_CURSOR, GLFW\_CURSOR\_NORMAL);

//glfwSetCursorEnterCallback(window, cursorEnterCallback);

glfwSetMouseButtonCallback(window, mouseButtonCallback);

glfwSetScrollCallback(window, scrollCallback);

if (!window) {

glfwTerminate();

return -1;

}

unsigned char pixels[16 \* 16 \* 4];

memset(pixels, 0xff, sizeof(pixels));

GLFWimage image;

image.width = 16;

image.height = 16;

image.pixels = pixels;

GLFWcursor\* cursor = glfwCreateCursor(&image, 0, 0);

glfwSetCursor(window, cursor);

int screenWidth, screenHeight;

glfwGetFramebufferSize(window, &screenWidth, &screenHeight);

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glfwSwapBuffers(window);

glfwPollEvents();

}

glfwTerminate();

return 0;

}

static void cursorPositionCallback(GLFWwindow\* window, double xpos, double ypos) {

std::cout << xpos << ", " << ypos << std::endl;

}

void cursorEnterCallback(GLFWwindow\* window, int entered) {

if (entered) std::cout << "Entered window" << std::endl;

else std::cout << "Left window" << std::endl;

}

void mouseButtonCallback(GLFWwindow\* window, int button, int action, int mods) {

if (button == GLFW\_MOUSE\_BUTTON\_RIGHT && action == GLFW\_PRESS) {

std::cout << "Right Button Pressed" << std::endl;

}

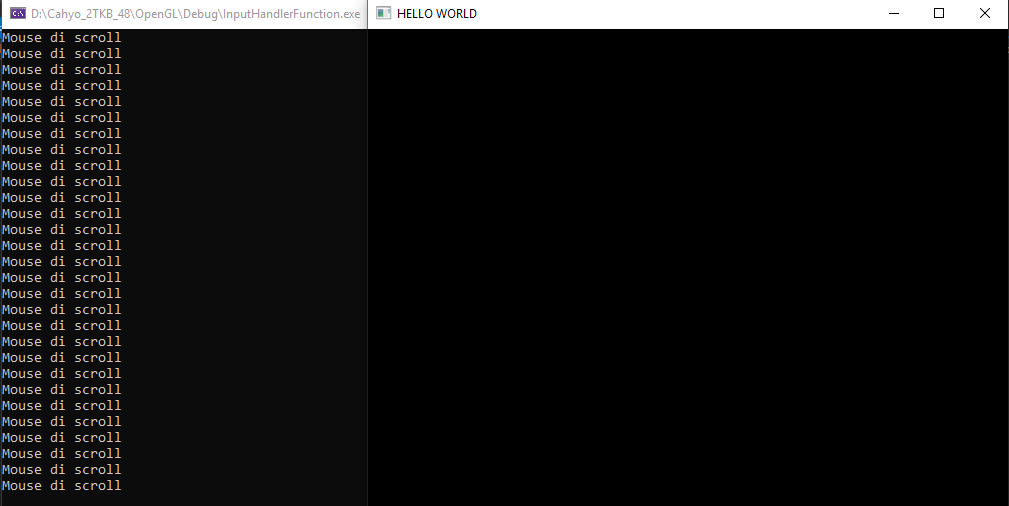
}

void scrollCallback(GLFWwindow\* window, double xoffset, double yoffset) {

std::cout << "Mouse di scroll" << std::endl;

}

Hasil:



Analisa:

Ketika mouse discroll maka akan muncul tulisan “Mouse di scroll”. Menggunakan fungsi scrollCallback untuk mengatur fungsi yang akan dilakukan jika mouse discroll. Lalu juga diinisialisasi callback scroll nya menggunakan fungsi glfwSetScrollCallback() dengan parameter window dan fungsi callback nya.

8. Program 8

Code:

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <iostream>

using namespace std;

constexpr auto SCREEN\_WIDTH = 640;

constexpr auto SCREEN\_HEIGHT = 480;

void character\_callback(GLFWwindow\* window, unsigned int codepoint);

void charmods\_callback(GLFWwindow\* window, unsigned int codepoint, int mods);

int main() {

GLFWwindow\* window;

if (!glfwInit()) return -1;

window = glfwCreateWindow(SCREEN\_WIDTH, SCREEN\_HEIGHT, "Hello World", NULL, NULL);

glfwSetCharModsCallback(window, charmods\_callback);

int screenWidth, screenHeight;

glfwGetFramebufferSize(window, &screenWidth, &screenHeight);

if (!window) {

glfwTerminate();

return -1;

}

glfwMakeContextCurrent(window);

glViewport(0.0f, 0.0f, screenWidth, screenHeight);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

glOrtho(0, SCREEN\_WIDTH, 0, SCREEN\_HEIGHT, 0, 1);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glfwSwapBuffers(window);

glfwPollEvents();

}

glfwTerminate();

return 0;

}

void character\_callback(GLFWwindow\* window, unsigned int codepoint) {

cout << codepoint << endl;

}

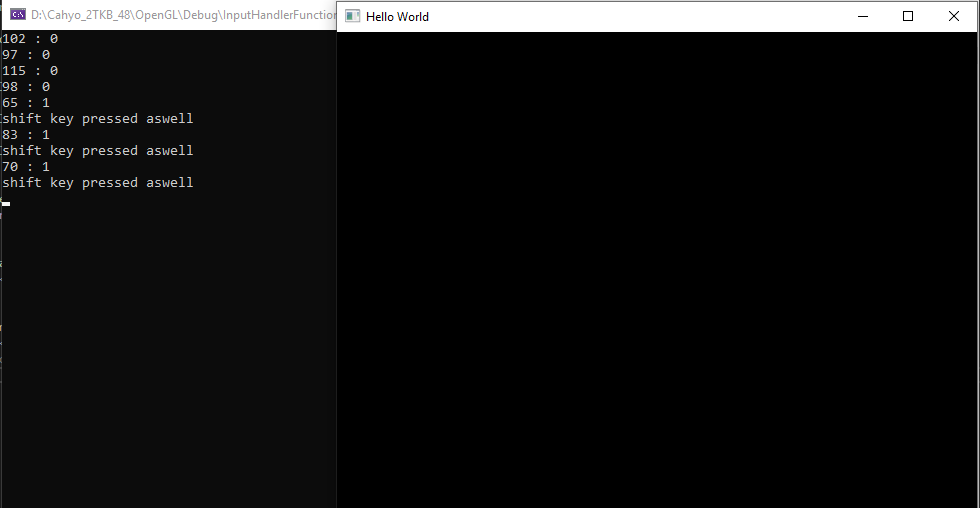
void charmods\_callback(GLFWwindow\* window, unsigned int codepoint, int mods) {

cout << codepoint << " : " << mods << endl;

if (mods == 1) cout << "shift key pressed aswell" << endl;

}

Hasil:



9. Program 9

Code:

#include <iostream>

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <math.h>

using namespace std;

void mouseButtonCallback(GLFWwindow\* window, int button, int action, int mods);

void keyCallback(GLFWwindow\* window, int key, int scancode, int action, int mods);

void drawCircle(GLfloat x, GLfloat y, GLfloat z, GLfloat radius, GLint numberOfSides);

int main() {

GLFWwindow\* window;

if (!glfwInit()) return -1;

window = glfwCreateWindow(640, 480, "Halo Halo Bandung", NULL, NULL);

glfwSetMouseButtonCallback(window, mouseButtonCallback);

glfwSetKeyCallback(window, keyCallback);

if (!window) {

glfwTerminate();

return -1;

}

while (!glfwWindowShouldClose(window)) {

glClear(GL\_COLOR\_BUFFER\_BIT);

glfwSwapBuffers(window);

glfwPollEvents();

}

glfwTerminate();

}

void mouseButtonCallback(GLFWwindow\* window, int button, int action, int mods) {

if (button == GLFW\_MOUSE\_BUTTON\_RIGHT && action == GLFW\_PRESS) {

glfwMakeContextCurrent(window);

float points[] = {

0.0f, 0.5f, 0.0f,

0.5f, -0.5f, 0.0f,

-0.5f, -0.5f, 0.0f

};

glewExperimental = GL\_TRUE;

glewInit();

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);

}

}

}

void keyCallback(GLFWwindow\* window, int key, int scancode, int action, int mods) {

if (key == GLFW\_KEY\_SPACE && action == GLFW\_PRESS) {

glfwMakeContextCurrent(window);

float points[] = {

0.0f, 0.5f, 0.0f,

0.5f, -0.5f, 0.0f,

-0.5f, -0.5f, 0.0f

};

glewExperimental = GL\_TRUE;

glewInit();

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(0.1, 0.75, 0.4, 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();

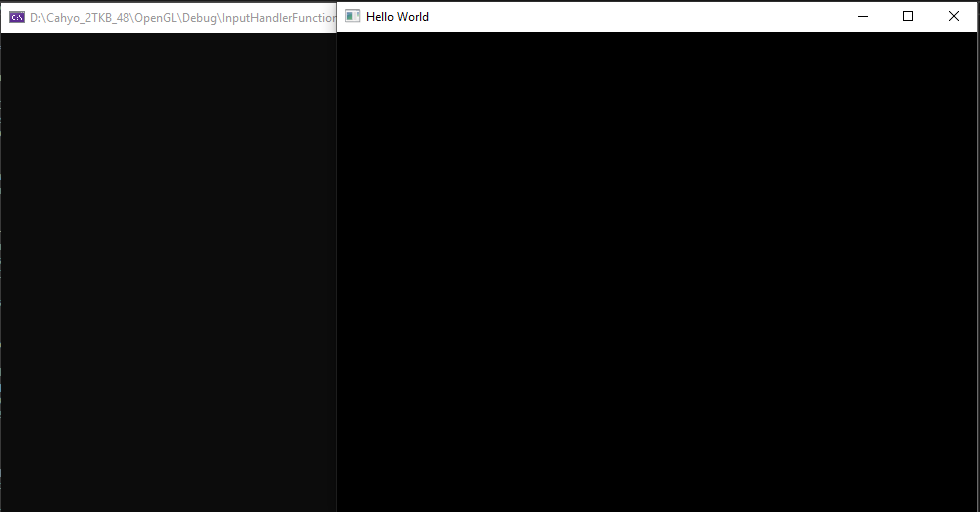
glfwSwapBuffers(window);

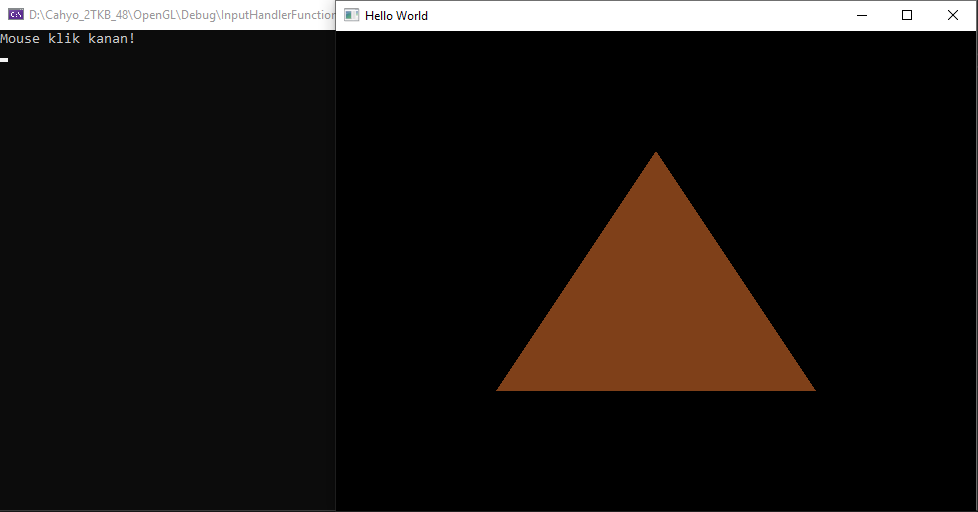
}

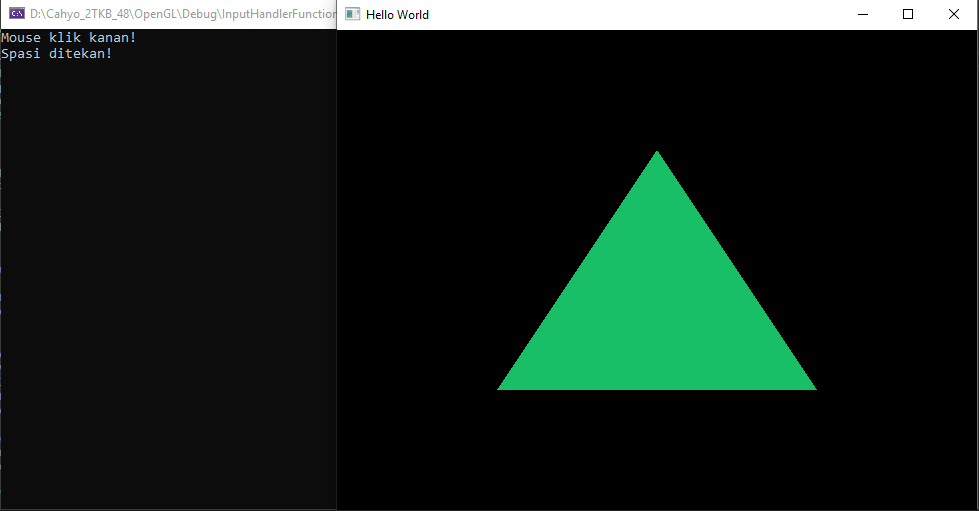
}

}

Hasil:

Sebelum ditekan

Klik Kanan

Spasi ditekan