Nama: Nur Cahyo Ihsan Prastyawan

Kelas: 2 D4 TK B

NRP: 2210181048

**WORKSHOP GRAFIKA KOMPUTER**

**MINI PROJECT**

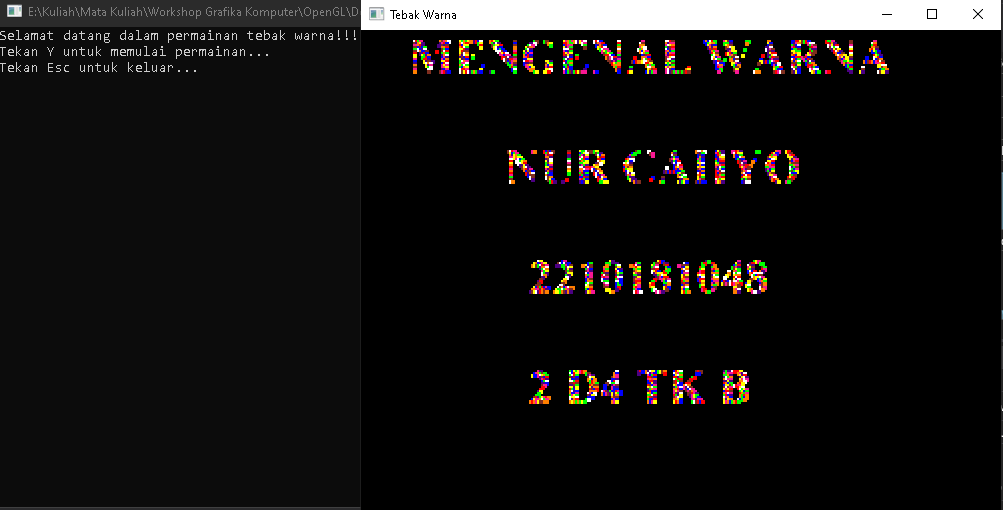
Deskripsi

Program ini ditujukan untuk anak - anak yang sedang mengenal warna. Program akan menampilkan beberapa persegi dengan berbagai macam warna. User memilih warna yang tepat sesuai perintah dari program. Terdapat sistem level pada program ini.

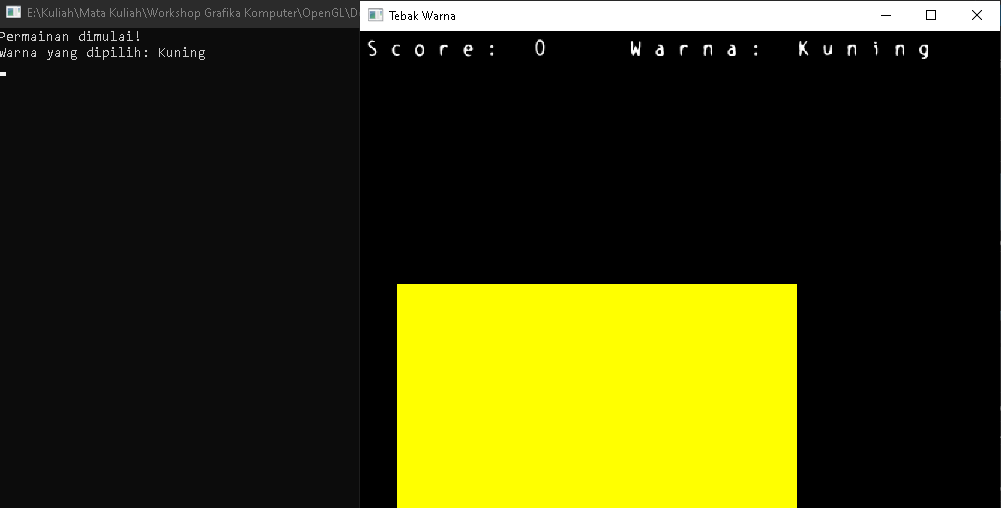
Alur program:

1. Program melakukan inisialisasi awal.
2. Program menampilkan beberapa persegi dengan warna – warna tertentu dan menentukan pilihan warna.
3. User mengklik persegi
4. Program menentukan apakah warna yang dipilih benar atau tidak
   1. Jika benar maka dilanjutkan
   2. Jika salah maka akan ditampilkan skor dan program berakhir.
5. Jika warna yang benar sudah habis, program akan melanjutkan ke level berikutnya
6. Pada level berikutnya program akan menggambar persegi lebih banyak lagi.

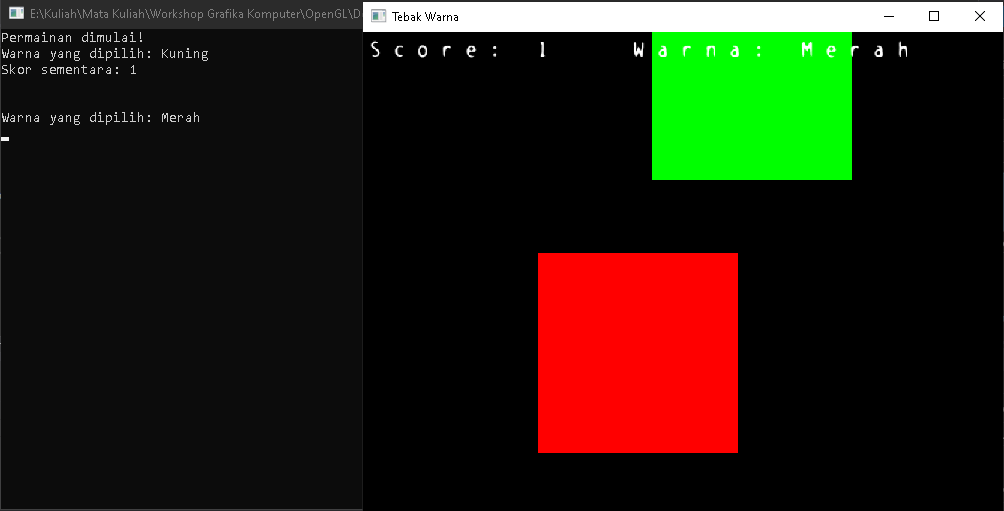
Hasil:



Tampilan Screen Awal



Tampilan ketika permainan dimulai saat awal



Tampilan ketika persegi yang ditekan benar, akan muncul tambahan satu persegi dan juga skor



Tampilan ketika mengklik persegi yang salah

Analisa

Dalam program ini, yang pertama dilakukan adalah menginisialisasi library seperti glfw dan glew yang lalu membuat window. Setelah itu membuat model view projection nya untuk koordinatnya

Berikut adalah contoh dari kode MVP

glm::mat4 Projection = glm::ortho(0.0f, (float)WIDTH, (float)HEIGHT, 0.0f, -1.0f, 1.0f);

glm::mat4 View = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, 0.0f));

glm::mat4 MVP = Projection \* View;

Kemudian persegi akan diinisialisasi menggunakan kelas Rectangle. Inisialisasi tersebut program membuat VAO, VBO dan juga shader. Dalam class Rectangle yang dibutuhkan adalah posisi, warna dan ukuran. Setelah inisialisasi tersebut, program mulai membuat tampilan awal dengan menggenerate banyak persegi yang warna nya acak. Program juga menunggu input dari pengguna. JIka menekan ‘Y; maka akan dilanjutkan ke permainan. Jika menekan ESC, maka akan keluar.

Berikut adalah kode untuk animasi nya

while (!next) {

if (glfwGetKey(window, GLFW\_KEY\_Y) == GLFW\_PRESS) next = true;

if (glfwGetKey(window, GLFW\_KEY\_ESCAPE) == GLFW\_PRESS) exit(1);

for (int i = 0; i < 100; i++) {

Rectangle anim1(Point{ WIDTH - 30.0f \* i,HEIGHT - 30.0f \* i }, generateRandomColor(), 30.0f);

renderer.AddRectangle(anim1, shader, MVP);

renderer.Draw(va, ib, shader);

}

for (int i = 0; i < 100; i++) {

Rectangle anim1(Point{ WIDTH - 30.0f\*i,30.0f + 30.0f \* i }, generateRandomColor(), 30.0f);

renderer.AddRectangle(anim1, shader, MVP);

renderer.Draw(va, ib, shader);

}

Ketika permainan dimulai, fungsi start akan dipanggil. Masuk pada fungsi while utama, program akan merender setiap persegi yang ada pada std::vector<Rectangle>.rect. Setelah semua digambar program akan menunggu input mouse dari user yang didefinisikan di fungsi callback mouse.

Ketika user klik kiri, program akan membaca posisi dari kursor. Jika posisi kursor masih pada area persegi dan persegi tersebut memiliki warna yang sama dengan pilihan program, maka skor akan bertambah 1. Jika sudah habis persegi yang benar, maka program akan melakukan level up. Sementara jika posisi kursor pada area persegi tetapi persegi tersebut memiliki warna yang tidak sama dengan pilihan program maka program akan mengeksekusi fungsi game over.

Skor akan bertambah selama user mengklik persegi yang benar dan level akan bertambah selama user menghabiskan semua persegi yang benar. Warna dan posisi dari persegi dihasilkan secara acak. Ukuran persegi tergantung dari level nya, semakin besar level maka semakin kecil persegi yang dihasilkan dan juga semakin banyak persegi yang ada. Game akan selesai jika user mengklik persegi yang salah.

Loading screen menggunakan file bitmap dengan resolusi yang lebih rendah dari layar. File bitmap tersebut dirender setiap pixel nya dengan bentuk persegi. Library yang digunakan adalah EasyBMP untuk membaca file bitmap tersebut.

Untuk menampilkan skor dan warna yang dipilih di window terdapat file texture berformat dds. File ini berisi huruf dan angka yang dapat dibaca di program. Program akan mengambil satu per satu huruf atau angka kemudian ditampilkan di window ketika terdapat perintah untuk mencetak huruf di window. Ukuran dan posisi bisa diatur sesuai dengan keinginan.

Untuk full source code ada di link berikut:

[https://github.com/cprastyawan/OpenGLProject/](https://github.com/cprastyawan/OpenGLProject/tree/master/MiniProject)

main.cpp

#include <GL/glew.h>

#include <GLFW/glfw3.h>

#include <user/IndexBuffer.h>

#include <user/Renderer.h>

#include <random>

#include <ctime>

#include <user/VertexArray.h>

#include <user/VertexBuffer.h>

#include <user/Shader.h>

#include <glm/glm.hpp>

#include <glm/gtc/matrix\_transform.hpp>

#include <stdio.h>

#include <iostream>

#include <vector>

#include <user/VertexBufferLayout.h>

#include <user/Rectangle.h>

#include <EasyBMP/EasyBMP.h>

#include <user/text2D.hpp>

#define WIDTH 640

#define HEIGHT 480

std::vector<Point> loadbmp(const char\* filename);

Color Red = { 1.0f, 0.0f, 0.0f, 1.0f };

Color Green = { 0.0f, 1.0f, 0.0f, 1.0f };

Color Blue = { 0.0f, 0.0f, 1.0f, 1.0f };

Color Yellow = { 1.0f, 1.0f, 0.0f, 1.0f };

Color Orange = { 1.0f, 127.0f / 255.0f, 0.0f, 1.0f };

Color Violet = { 75.0f / 255.0f, 0.0f, 130.0f / 255.0f, 1.0f };

Color Brown = { 126.0f / 255.0f, 46.0f / 255.0f, 31.0f / 255.0f, 1.0f };

Color Pink = { 248.0f / 255.0f, 24.0f / 255.0f, 148.0f / 255.0f, 1.0f };

Color White = { 1.0f,1.0f,1.0f,1.0f };

void errorMessage(std::string text);

int randomGenerator(int min, int max);

static double xpos, ypos;

static Color colorPick;

Rectangle initRect(Point{ WIDTH / 2, HEIGHT / 2 }, Color{ 0.0f, 0.0f, 0.0f, 0.0f }, 0.0f);

std::vector<Rectangle> rect;

Color selectedColor;

static unsigned int score;

static unsigned int level;

static unsigned int countTrueRectangle;

Color generateRandomColor(); //Fungsi untuk menghasilkan warna secara acak

void mouseButtonCallback(GLFWwindow\* window, int button, int action, int mods); //Fungsi callback mouse

std::vector<Rectangle> generateRectangle(int count, Color colorChoose); //Fungsi untuk menghasilkan persegi

void clearRectangle(std::vector<Rectangle> &rect); //Fungsi untuk menghapus semua persegi

void start(std::vector<Rectangle> &rect, Color &, unsigned int &level); //fungsi untuk memulai permainan

void levelUp(std::vector <Rectangle> &rect, unsigned int &level, Color &buffer, unsigned int &score); //fungsi untuk naik level

void scoreUp(unsigned int &score); //Fungsi untuk skor

void gameOver(unsigned int &score, std::vector<Rectangle> &rect, Color &buffer); //Fungsi untuk kalah

void restartGame(unsigned int &score, std::vector<Rectangle>& rect, Color &buffer); //Fungsi untuk mengulangi permainan

unsigned int checkTrueRectangle(Color c, std::vector<Rectangle> rect); //Fungsi untuk mengecek apakah persegi masih ada yang benar atau tidak

const char\* printColor(Color c); //Fungsi untuk mencetak nama dari warna

int main() {

srand((unsigned)time(NULL));

glfwWindowHint(GLFW\_VERSION\_MAJOR, 3);

glfwWindowHint(GLFW\_VERSION\_MINOR, 3);

glfwWindowHint(GLFW\_OPENGL\_PROFILE, GLFW\_OPENGL\_CORE\_PROFILE);

if (!glfwInit()) errorMessage("glfwInit() error!\n");

GLFWwindow\* window = glfwCreateWindow(WIDTH, HEIGHT, "Tebak Warna", NULL, NULL);

if (!window) errorMessage("glewInit() error!\n");

glfwMakeContextCurrent(window);

glfwSetMouseButtonCallback(window, mouseButtonCallback);

if (glewInit() != GLEW\_OK) errorMessage("glewInit() error!\n");

glm::mat4 Projection = glm::ortho(0.0f, (float)WIDTH, (float)HEIGHT, 0.0f, -1.0f, 1.0f);

glm::mat4 View = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, 0.0f));

glm::mat4 MVP = Projection \* View;

printf("Selamat datang dalam permainan tebak warna!!!\n");

printf("Tekan Y untuk memulai permainan...\n");

printf("Tekan Esc untuk keluar...\n");

bool next = false;

std::vector<Point> bmp = loadbmp("no1.bmp");

Rectangle anim(Point{ 0.0f,0.0f }, Color{ 0.0f,0.0f,0.0f,0.0f }, 2.0f);

glfwSwapInterval(3);

//char \*text = (char\*)"Test123";

initText2D("../res/font/Holstein.DDS");

while (!next) {

VertexArray va;

VertexBuffer vb(initRect.pos, initRect.sizeRect);

VertexBufferLayout layout;

layout.Push<float>(3);

va.AddBuffer(vb, layout);

IndexBuffer ib(initRect.indices, initRect.indicescount);

Shader shader("res/shaders/Basic.shader");

shader.Bind();

va.Unbind();

vb.Unbind();

ib.Unbind();

shader.Unbind();

Renderer renderer;

if (glfwGetKey(window, GLFW\_KEY\_Y) == GLFW\_PRESS) next = true;

if (glfwGetKey(window, GLFW\_KEY\_ESCAPE) == GLFW\_PRESS) exit(1);

for (int i = 0; i < bmp.size(); i++) {

anim.color = generateRandomColor();

anim.Position = glm::translate(glm::mat4(), glm::vec3(2 \* bmp.at(i).x, 2 \* bmp.at(i).y, 0.0f));

renderer.AddRectangle(anim, shader, MVP);

renderer.Draw(va, ib, shader);

}

glfwSwapBuffers(window);

glfwPollEvents();

}

system("cls");

glfwSwapInterval(1);

start(rect, selectedColor, level);

while (!glfwWindowShouldClose(window)) {

VertexArray va;

VertexBuffer vb(initRect.pos, initRect.sizeRect);

VertexBufferLayout layout;

layout.Push<float>(3);

va.AddBuffer(vb, layout);

IndexBuffer ib(initRect.indices, initRect.indicescount);

Shader shader("res/shaders/Basic.shader");

shader.Bind();

va.Unbind();

vb.Unbind();

ib.Unbind();

shader.Unbind();

Renderer renderer;

renderer.Clear();

shader.Bind();

va.Bind();

vb.Bind();

ib.Bind();

renderer.AddRectangle(initRect, shader, MVP);

renderer.Draw(va, ib, shader);

for (int i = 0; i < rect.size(); i++) {

renderer.AddRectangle(rect.at(i), shader, MVP);

renderer.Draw(va, ib, shader);

}

std::string text = "Score: " + std::to\_string(score) + " Warna: " + printColor(selectedColor);

printText2D(text, 0, HEIGHT + 85, 30);

glfwSwapBuffers(window);

glfwPollEvents();

}

cleanupText2D();

}

inline void errorMessage(std::string text) {

std::cout << text;

system("pause");

exit(-1);

}

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

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

glfwGetCursorPos(window, &xpos, &ypos);

glReadPixels(xpos, -ypos + HEIGHT, 1, 1, GL\_RGBA, GL\_FLOAT, &colorPick);

for (int i = 0; i < rect.size(); i++) {

if ((xpos <= rect.at(i).Max.x && xpos >= rect.at(i).Min.x) && (ypos <= rect.at(i).Max.y && ypos >= rect.at(i).Min.y)

&& rect.at(i).color == selectedColor) {

rect.erase(rect.begin() + i);

scoreUp(score);

if (checkTrueRectangle(selectedColor, rect) == 0) levelUp(rect, level, selectedColor, score);

}

else if ((xpos <= rect.at(i).Max.x && xpos >= rect.at(i).Min.x) && (ypos <= rect.at(i).Max.y && ypos >= rect.at(i).Min.y)

&& rect.at(i).color != selectedColor)

gameOver(score, rect, selectedColor);

}

}

}

int randomGenerator(int min, int max) {

return rand() % max + min;

}

std::vector<Rectangle> generateRectangle(int count, Color colorChoose) {

std::vector<Rectangle> temp;

for (int i = 0; i < count; i++) {

Color c;

if (i == 0) c = colorChoose;

else c = generateRandomColor();

Point point;

point.x = (float)randomGenerator(0, WIDTH);

point.y = (float)randomGenerator(0, HEIGHT);

temp.push\_back(Rectangle(point, c, 200 / count));

}

return temp;

}

void clearRectangle(std::vector<Rectangle> &rect) {

rect.clear();

}

void start(std::vector<Rectangle> &rect, Color &buffer, unsigned int &level) {

printf("Permainan dimulai!\n");

level = 1;

int chooseColor = randomGenerator(1, 3);

buffer = generateRandomColor();

rect = generateRectangle(level, buffer);

printf("Warna yang dipilih: %s\n", printColor(buffer));

}

void levelUp(std::vector<Rectangle> &rect, unsigned int &level, Color &buffer, unsigned int &score) {

printf("Skor sementara: %d\n", score);

rect.clear();

level += 1;

buffer = generateRandomColor();

rect = generateRectangle(level, buffer);

printf("\n\n");

printf("Warna yang dipilih: %s\n", printColor(buffer));

}

Color generateRandomColor() {

int x = randomGenerator(1, 9);

if (x == 1) return Red;

else if (x == 2) return Green;

else if (x == 3) return Blue;

else if (x == 4) return Orange;

else if (x == 5) return Yellow;

else if (x == 6) return Violet;

else if (x == 7) return Brown;

else if (x == 8) return Pink;

else if (x == 9) return White;

}

void scoreUp(unsigned int &score) {

score++;

}

void gameOver(unsigned int &score, std::vector<Rectangle>& rect, Color &buffer) {

char input;

printf("Anda salah menekan warna!\n");

printf("Score anda: %d\n", score);

printf("Masukkan y untuk main lagi: ");

std::cin >> input;

if (input == 'y' || input == 'Y') restartGame(score, rect, buffer);

else exit(1);

}

void restartGame(unsigned int &score, std::vector<Rectangle>& rect, Color &buffer) {

rect.clear();

score = 0;

system("pause");

start(rect, buffer, level);

}

unsigned int checkTrueRectangle(Color c, std::vector<Rectangle> rect) {

int count = 0;

for (int i = 0; i < rect.size(); i++) {

if (rect.at(i).color == c) count++;

else continue;

}

return count;

}

const char\* printColor(Color c) {

if (c == Red) return "Merah";

else if (c == Green) return "Hijau";

else if (c == Blue) return "Biru";

else if (c == Orange) return "Oranye";

else if (c == Yellow) return "Kuning";

else if (c == Violet) return "Ungu";

else if (c == Brown) return "Coklat";

else if (c == Pink) return "Merah Muda";

else if (c == White) return "Putih";

else return "Warna tidak terdaftar";

}

std::vector<Point> loadbmp(const char\* filename) {

BMP bmp;

bmp.ReadFromFile(filename);

std::vector<Point> buf;

for (int i = 0; i < bmp.TellHeight(); i++) {

for (int j = 0; j < bmp.TellWidth(); j++) {

if (bmp(j, i)->Red == 0) buf.push\_back(Point{ (float)j, (float)i });

}

}

return buf;

}