LAPORAN PRAKTIKUM PERTEMUAN 7,8,9, DAN 10.

PRAKTIKUM 7

Pada awal kode, kita mendefinisikan beberapa variabel seperti BOX_SIZE untuk ukuran objek 3D, angle untuk rotasi objek, dan textureId untuk menyimpan ID tekstur OpenGL yang dihasilkan dari gambar.

```
const float BOX_SIZE = 7.0f;

float angle = 0.0f;

GLuint textureId;
```

Fungsi handleKeypress digunakan untuk menutup jendela program jika tombol Escape ditekan.

```
void handleKeypress(GLFWwindow* window, int key, int scancode, int action, int mods) {
   if (key == GLFW_KEY_ESCAPE && action == GLFW_PRESS)
      glfwSetWindowShouldClose(window, GLFW_TRUE);}
```

Fungsi loadTexture digunakan untuk memuat gambar dan mengatur filter pada tekstur, memastikan tekstur ditampilkan dengan baik.

```
GLuint loadTexture(Image* image) {
   GLuint textureId;
   glGenTextures(1, &textureId);
   glBindTexture(GL_TEXTURE_2D, textureId);
   glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image->width, image->height, 0, GL_RGB, GL_UNSIGNED_BYTE, image->pixels);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
   glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
   return textureId;}
```

Fungsi initRendering di sini mengaktifkan beberapa fitur OpenGL seperti pengujian kedalaman, pencahayaan, dan material warna. Kemudian, gambar dimuat menggunakan loadBMP() dan loadTexture() untuk mendapatkan ID tekstur.

```
void initRendering() {
   glEnable(GL_DEPTH_TEST);
   glEnable(GL_LIGHTING);
```

```
    glEnable(GL_LIGHT0);
    glEnable(GL_NORMALIZE);
    glEnable(GL_COLOR_MATERIAL);
    Image* image = loadBMP("852.bmp");
    textureId = loadTexture(image);
    delete image;}
```

Fungsi drawScene digunakan untuk menggambar objek 3D. Di sini, kita mengatur pencahayaan ambient dan posisi sumber cahaya, lalu menggambar kubus dengan tekstur yang sudah diterapkan.

```
void drawScene() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glTranslatef(0.0f, 0.0f, -20.0f);

    GLfloat ambientLight[] = { 0.3f, 0.3f, 0.3f, 1.0f };
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, ambientLight);
    GLfloat lightColor[] = { 0.7f, 0.7f, 0.7f, 1.0f };
    GLfloat lightPos[] = { -2 * BOX_SIZE, BOX_SIZE, 4 * BOX_SIZE, 1.0f };
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightColor);
    glLightfv(GL_LIGHT0, GL_POSITION, lightPos);

    glEnable(GL_TEXTURE_2D);
    glBindTexture(GL_TEXTURE_2D, textureId);
    glBegin(GL_QUADS);
    glNormal3f(0.0, 0.0f, 1.0f);
    glTexCoord2f(0.0f, 0.0f); glVertex3f(-BOX_SIZE/2, -BOX_SIZE/2,
BOX_SIZE/2);
    glTexCoord2f(1.0f, 0.0f); glVertex3f( BOX_SIZE/2, -BOX_SIZE/2,
BOX_SIZE/2);
    glTexCoord2f(1.0f, 1.0f); glVertex3f( BOX_SIZE/2, BOX_SIZE/2,
BOX_SIZE/2);
```

```
        glTexCoord2f(0.0f, 1.0f); glVertex3f(-BOX_SIZE/2,  BOX_SIZE/2,
    BOX_SIZE/2);
        glEnd();
        glDisable(GL_TEXTURE_2D)}
```

- Fungsi main menginisialisasi GLFW dan GLEW, membuat jendela grafis dengan ukuran 800x600, dan memulai loop utama rendering.

```
int main() {
    if (!glfwInit()) {
        std::cerr << "Failed to initialize GLFW\n";
        return EXIT_FAILURE;}
```

```
GLFWwindow* window = glfwCreateWindow(800, 600, "GLFW Texture Box", NULL,
NULL);
if (!window) {
    std::cerr << "Failed to create window\n";
    glfwTerminate();
    return EXIT_FAILURE;
}

glewExperimental = GL_TRUE;
if (glewInit() != GLEW_OK) {
    std::cerr << "Failed to initialize GLEW\n";
    return EXIT_FAILURE;
}

glEnable(GL_DEPTH_TEST);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(45.0, 800.0 / 600.0, 1.0, 200.0);

while (!glfwWindowShouldClose(window)) {
```

```
        angle += 0.2f;
        if (angle > 360) angle -= 360;
        drawScene();
        glfwSwapBuffers(window);
        glfwPollEvents();
    }

    glfwDestroyWindow(window);
    glfwTerminate();
    return 0;
}
```

Praktikum 8

Variabel l mewakili posisi cahaya, n adalah normal permukaan tempat bayangan diproyeksikan, dan e adalah titik pandang atau posisi kamera.

```
float l[] = { 0.0, 80.0, 0.0 }; // posisi cahaya
float n[] = { 0.0, -40.0, 0.0 }; // normal bidang proyeksi (lantai)
float e[] = { 0.0, -60.0, 0.0 }; // titik pandang proyeksi
```

ungsi gluCylinder menggambar sebuah objek kerucut, yang akan diberikan bayangan.

```
GLUquadric* quadric = gluNewQuadric();
gluQuadricDrawStyle(quadric, GLU_FILL);
gluCylinder(quadric, 20.0, 0.0, 50.0, 40, 50);
gluDeleteQuadric(quadric);
```

Matriks bayangan dihitung dengan menggunakan rumus proyeksi, dan proyeksi ini diterapkan pada objek yang digambar.

```
float d, c, mat[16];
d = n[0]*l[0] + n[1]*l[1] + n[2]*l[2];
c = e[0]*n[0] + e[1]*n[1] + e[2]*n[2] - d;
mat[0] = l[0]*n[0]+c;
```

glMultMatrixf(mat); // Menerapkan matriks bayangan ke objek

Fungsi render digunakan untuk menggambar objek dan bayangannya secara bersamaan. Bayangan dibuat dengan cara memproyeksikan objek ke permukaan berdasarkan posisi cahaya.

```
void render() {
    glClearColor(0.0, 0.6, 0.9, 0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLightfv(GL_LIGHT0, GL_POSITION, l);

    glDisable(GL_LIGHTING);
    glColor3f(1.0, 1.0, 0.0);
    glPointSize(5.0f);
    glBegin(GL_POINTS);
    glVertex3f(l[0], l[1], l[2]);
    glEnd();

    glColor3f(0.8, 0.8, 0.8);
    glBegin(GL_QUADS);
    glNormal3f(0.0, 1.0, 0.0);
    glVertex3f(-1300.0, e[1]-0.1, 1300.0);
    glVertex3f(1300.0, e[1]-0.1, 1300.0);
    glVertex3f(1300.0, e[1]-0.1, -1300.0);
    glVertex3f(-1300.0, e[1]-0.1, -1300.0);
    glEnd();

    glPushMatrix();
    glRotatef(ry, 0, 1, 0);
    glRotatef(rx, 1, 0, 0);
    glEnable(GL_LIGHTING);
    glColor3f(0.0, 0.0, 0.8);
    draw();  // Menggambar objek utama
    glPopMatrix();
```

```
    glPushMatrix();
    glShadowProjection(l, e, n);  // Menerapkan proyeksi bayangan
    glRotatef(ry, 0, 1, 0);
    glRotatef(rx, 1, 0, 0);
    glDisable(GL_LIGHTING);
    glColor3f(0.4, 0.4, 0.4);
    draw();  // Menggambar bayangan objek
    glPopMatrix();
}
```

Praktikum 9

Variabel untuk Posisi dan Kamera:
Pada awal kode, kita mendefinisikan beberapa variabel untuk mengatur posisi objek dan juga posisi kamera. Kamera diletakkan di belakang objek dan akan bergerak mengikuti input pengguna.

```
float angle = 0.0f, deltaAngle = 0.0f, ratio;
float x = -5.0f, y = 12.0f, z = 40.0f;
float lx = 0.0f, ly = 0.0f, lz = -1.0f;  // Vektor arah pandang kamera
```

- Variabel angle, deltaAngle, dan ratio digunakan untuk rotasi objek dan pengaturan aspek rasio kamera.
- Variabel x, y, z mengatur posisi kamera di ruang 3D, sementara lx, ly, dan lz digunakan untuk mengatur arah pandang kamera.

Pengaturan Input Keyboard:
Fungsi keyCallback() menangani input dari keyboard untuk mengontrol rotasi objek, pergerakan kaki, bola, dan aksi tendang. Misalnya, tombol W, S, A, D digunakan untuk merotasi objek, sementara tombol O dan P digunakan untuk menggerakkan bola dan kaki.

```
void keyCallback(GLFWwindow* window, int key, int scancode, int action, int mods) {
    if (action == GLFW_PRESS || action == GLFW_REPEAT) {
        switch (key) {
            case GLFW_KEY_W: rotAngleX += 2; break;  // Rotasi objek pada sumbu X
            case GLFW_KEY_S: rotAngleX -= 2; break;  // Rotasi objek pada sumbu X
            case GLFW_KEY_A: rotAngleY += 2; break;  // Rotasi objek pada sumbu Y
```

```
        case GLFW_KEY_D: rotAngleY -= 2; break;  // Rotasi objek pada sumbu Y
        case GLFW_KEY_O:  // Gerakan kaki
            posXKaki -= 1;
            if (posXBola < -2.9f) posXBola += 1;
            break;
        case GLFW_KEY_P:  // Gerakan bola
            posXKaki += 1;
            posXBola -= 1;
            break;
        case GLFW_KEY_K: kick = 1; break;  // Tendang bola
        case GLFW_KEY_SPACE:  // Reset posisi semua objek
            rotAngleX = rotAngleY = rotAngleZ = 0;
            posXKaki = 10; posXBola = -10; posYKaki = 6; posYBola = -5;
            rotKaki = kick = roll = 0;
            break;
        case GLFW_KEY_ESCAPE: glfwSetWindowShouldClose(window, GLFW_TRUE);
break;
      }
   }
}
```

- Fungsi ini akan memeriksa setiap input tombol yang ditekan. Misalnya, tombol W dan S digunakan untuk merotasi objek pada sumbu X, sementara tombol A dan D digunakan untuk merotasi objek pada sumbu Y.
- Tombol O dan P digunakan untuk menggerakkan kaki dan bola, dan tombol K digunakan untuk menendang bola.

Animasi Pergerakan Kaki dan Bola:
Fungsi pergerakanKaki() mengatur tiga fase animasi tendangan kaki: ayunan ke belakang, dorongan ke depan, dan kembali ke posisi awal. Fungsi ini juga memeriksa apakah bola terkena tendangan dan mulai bergerak.

```
void pergerakanKaki() {
   if (kick == 1) {
      if (rotKaki <= 45) rotKaki += 0.03f;
      if (rotKaki > 44.9f) kick = 2;
```

```
    }
    if (posXBola > -2.9f) touch = 1;  // Memeriksa apakah bola disentuh
    else if (posXBola < -12) touch = 0;

    if (kick == 2) {
        if (rotKaki >= -90) {
            rotKaki -= 0.2f;
            if (rotKaki < 1 && touch == 1) roll = 1;  // Bola mulai bergerak
        }
        if (rotKaki < -90) kick = 3;
    }

    if (kick == 3) {
        if (rotKaki <= 0) rotKaki += 0.05f;
        if (rotKaki > -1) kick = 0;  // Kembali ke posisi awal
    }
}
```

- Animasi tendangan dibagi menjadi tiga fase yang masing-masing memanipulasi rotasi objek kaki.
- Fungsi ini juga memeriksa apakah bola telah disentuh kaki, dan jika ya, bola akan mulai bergerak (roll).

Menggerakkan Bola:
Fungsi pergerakanBola() mengatur pergerakan bola setelah terkena tendangan. Bola bergerak ke kiri hingga jarak tertentu, setelah itu berhenti.

```
void pergerakanBola() {
    if (roll == 1) {
        if (jarak > 0) {
            posXBola -= 0.03f;  // Bola bergerak
            jarak -= 0.01f;
        }
        if (jarak < 0) {
            roll = 0;  // Bola berhenti
```

```
            jarak = 1;
        }
    }
}
```

- Setelah bola tersentuh kaki, fungsi ini menggerakkan bola ke kiri, dan pergerakan berhenti ketika jarak yang ditentukan habis.

Menggambar Objek (Kaki dan Bola):
Fungsi Object() menggambar objek utama, yaitu kaki penendang dan bola yang bergerak.

```
void Object() {
    glPushMatrix();
    glColor3f(0.1f, 0.1f, 0.2f);  // Kaki penendang
    glTranslatef(0, 3, 0);
    Balok(5, 5, 3);  // Menggambar kaki
    glPopMatrix();


    glPushMatrix();
    pergerakanKaki();
    glColor3f(0.8f, 0.3f, 0.3f);  // Bola
    glTranslatef(posXBola, posYBola, 0);
    gluSphere(IDquadric, 1.0, 20, 20);  // Menggambar bola
    glPopMatrix();
}
```

- Fungsi Balok(5, 5, 3) menggambar kaki penendang menggunakan objek balok 3D. Bola digambar dengan menggunakan gluSphere, yang menghasilkan bola 3D.

Menampilkan Scene:
Fungsi display() digunakan untuk menggambar scene secara keseluruhan, memproses input dari kamera, dan merender objek-objek yang telah dipersiapkan.

cpp

Copy code

```
void display(GLFWwindow* window) {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);    // Membersihkan buffer
```

```
    if (deltaMove) moveMeFlat(deltaMove);  // Menggerakkan kamera jika ada input
    if (deltaAngle) {
        angle += deltaAngle;
        orientMe(angle);  // Rotasi kamera
    }

    glPushMatrix();
    glRotated(rotAngleX, 1, 0, 0);
    glRotated(rotAngleY, 0, 1, 0);
    glRotated(rotAngleZ, 0, 0, 1);
    Grid();  // Menggambar grid untuk orientasi
    Object();  // Menggambar objek utama (kaki + bola)
    glPopMatrix();

    glfwSwapBuffers(window);  // Menukar buffer untuk efek animasi
}
```

Praktikum 10

Gerakan Kamera:
Kamera berfungsi untuk melihat dan menjelajahi objek dalam ruang 3D. Fungsi moveMeFlat() digunakan untuk memindahkan posisi kamera maju atau mundur dalam arah pandangnya.

```
void moveMeFlat(int i) {
    x = x + i * (lx) * 0.1f;
    z = z + i * (lz) * 0.1f;
    glLoadIdentity();  // Mengatur ulang matriks transformasi
    gluLookAt(x, y, z, x + lx, y + ly, z + lz, 0.0f, 1.0f, 0.0f);  // Menetapkan posisi dan arah pandang kamera
}
```

Orientasi Kamera:
Fungsi orientMe() digunakan untuk mengubah arah pandang kamera berdasarkan sudut rotasi yang diberikan. Fungsi ini mengubah vektor pandang kamera (lx, lz) berdasarkan input sudut.

```
void orientMe(float ang) {
    lx = sin(ang / 10);
    lz = -cos(ang / 10);
    glLoadIdentity();  // Mengatur ulang matriks transformasi
    gluLookAt(x, y, z, x + lx, y + ly, z + lz, 0.0f, 1.0f, 0.0f);  // Menetapkan arah pandang kamera yang baru
}
```

Pengaturan Pencahayaan dan Material:
Pencahayaan sangat penting dalam rendering 3D agar objek terlihat realistis. Fungsi ini mengaktifkan pencahayaan dan menetapkan properti material untuk objek-objek yang ada di scene.

```
cpp
Copy code
glEnable(GL_LIGHT0);  // Mengaktifkan sumber cahaya
glLightfv(GL_LIGHT0, GL_POSITION, light_position);  // Menetapkan posisi cahaya
glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);  // Menetapkan warna material objek
```

Menggambar Objek:
Untuk menggambar objek 3D, kita menggunakan beberapa fungsi OpenGL seperti glPushMatrix(), glTranslatef(), dan glPopMatrix() untuk transformasi objek, serta gluSphere() untuk menggambar bola.

```
glPushMatrix();
glTranslatef(10.0f, 0.0f, 0.0f);  // Menggeser objek ke posisi yang diinginkan
gluSphere(IDquadric, 1.0f, 20, 20);  // Menggambar objek bola dengan radius 1.0
glPopMatrix();
```

Pergerakan Objek:
Praktikum ini juga mengatur pergerakan objek seperti bola yang bergerak di dalam scene. Fungsi berikut menggerakkan objek bola berdasarkan input yang diberikan.

```
void pergerakanBola() {
    if (roll == 1) {
        if (jarak > 0) {
            posXBola -= 0.03f;  // Menggerakkan bola ke kiri
            jarak -= 0.01f;  // Mengurangi jarak pergerakan bola
        }
```

```
        if (jarak < 0) {
            roll = 0;  // Menghentikan pergerakan bola
            jarak = 1;  // Reset jarak
        }
    }
}
```

Menampilkan Scene:
Fungsi display() menggambar semua objek dan memperbarui tampilan pada setiap frame. Fungsi ini juga menangani input kamera dan animasi objek.

```
void display(GLFWwindow* window) {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);    // Membersihkan buffer layar
    if (deltaMove) moveMeFlat(deltaMove);  // Memindahkan kamera
    if (deltaAngle) {
        angle += deltaAngle;
        orientMe(angle);  // Mengubah arah pandang kamera
    }

    glPushMatrix();
    glRotated(rotAngleX, 1, 0, 0);  // Rotasi objek pada sumbu X
    glRotated(rotAngleY, 0, 1, 0);  // Rotasi objek pada sumbu Y
    glRotated(rotAngleZ, 0, 0, 1);  // Rotasi objek pada sumbu Z
    Grid();  // Menggambar grid sebagai referensi
    Object();  // Menggambar objek utama (bola dan kaki)
    glPopMatrix();

    glfwSwapBuffers(window);  // Menukar buffer untuk animasi yang lebih halus
}
```

Menggambar Grid dan Objek:

Fungsi Grid() menggambar grid untuk memberikan referensi spasial, sedangkan Object() menggambar objek yang ada dalam scene, seperti bola dan kaki.

cpp

Copy code

```
void Grid() {
   glBegin(GL_LINES);
   for (int i = -10; i < 10; i++) {
      glVertex3f(i, 0, -10);
      glVertex3f(i, 0, 10);
      glVertex3f(-10, 0, i);
      glVertex3f(10, 0, i);
   }
   glEnd();
}
```

Loop Utama:
Fungsi main() adalah tempat inisialisasi sistem dan pengaturan kontrol. Pada loop utama, fungsi display() dipanggil secara terus-menerus untuk memperbarui tampilan.

```
int main() {
   if (!glfwInit()) {
      std::cerr << "Failed to initialize GLFW\n";
      return EXIT_FAILURE;
   }

   GLFWwindow* window = glfwCreateWindow(800, 600, "3D Manipulation", NULL, NULL);
   if (!window) {
      std::cerr << "Failed to create window\n";
      glfwTerminate();
      return EXIT_FAILURE;
   }

   glewExperimental = GL_TRUE;
   if (glewInit() != GLEW_OK) {
      std::cerr << "Failed to initialize GLEW\n";
```

```
        return EXIT_FAILURE;
    }

    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, 800.0 / 600.0, 1.0, 200.0);

    while (!glfwWindowShouldClose(window)) {
        display(window);  // Menggambar scene
        glfwSwapBuffers(window);
        glfwPollEvents();  // Menangani event input
    }

    glfwDestroyWindow(window);
    glfwTerminate();
    return 0;
}
```