LAPORAN PRAKTIKUM

BAHASA PEMROGRAMAN 2

MODUL 4

Dosen pengampu : Yulyanto, S.Kom., M.TI.

Disusun oleh :

Nama : Muhammad Rizal Nurfirdaus

NIM : 20230810088

Jadwal : Rabu, 14:40 – 16:15

Kelas : TINFC-2023-04

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS KUNINGAN

2025

DAFTAR ISI

BAB I

PRETEST

Tuliskan Struktur Program Applet Java

Kerjakan pda selembar kertas dengan waktu 5 menit dan dikumpulkan melalui Asisten lab.

```
import java.applet.Applet;

import java.awt.Graphics;


// Kelas Applet yang merupakan turunan dari kelas Applet

public class MyApplet extends Applet {


   // Metode ini dipanggil pertama kali saat applet dimuat

   @Override

   public void init() {

      // Inisialisasi komponen atau variabel

      System.out.println("Applet diinisialisasi");

   }


   // Metode ini dipanggil setiap kali applet dijalankan

   @Override

   public void start() {

      System.out.println("Applet dimulai");

   }


   // Metode ini digunakan untuk menggambar grafik atau teks di applet

   @Override

   public void paint(Graphics g) {

      g.drawString("Hello, Applet!", 20, 20);

      System.out.println("Applet digambar");

   }
```

```
    // Metode ini dipanggil ketika applet dihentikan sementara
    @Override
    public void stop() {
        System.out.println("Applet dihentikan sementara");
    }

    // Metode ini dipanggil ketika applet ditutup atau pengguna meninggalkan halaman
    @Override
    public void destroy() {
        System.out.println("Applet dihancurkan");
    }
}
```

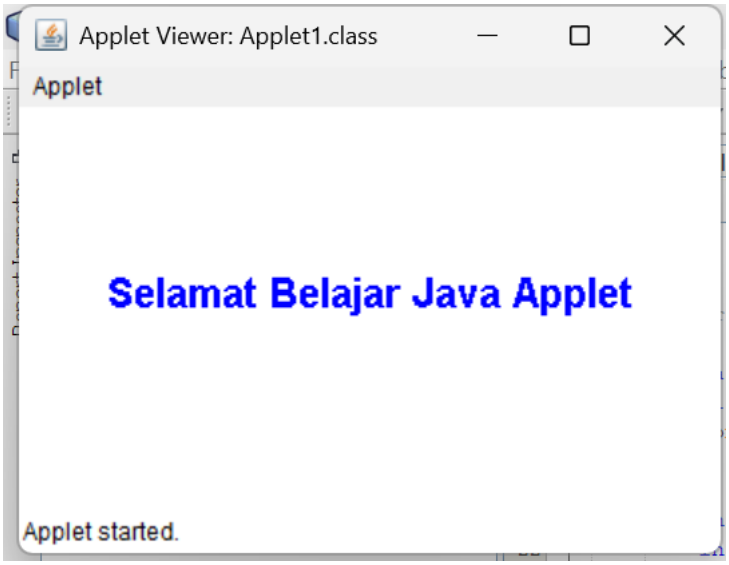
BAB II

PRAKTIKUM

Applet1

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Applet.java to edit this template
 */

import java.applet.Applet;
import java.awt.Color;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;

/**
 *
 * @author Muhammad Rizal Nurfirdaus
 */
public class Applet1 extends Applet {
    public void paint(Graphics g){
      Font f = new Font("sansSerif", Font.BOLD, 20);
      g.setFont(f);
      g.setColor(Color.BLUE);
      int xPusat=this.getSize().width/2;
      int yPusat=this.getSize().height/2;
      String s="Selamat Belajar Java Applet";
      FontMetrics fm= this.getFontMetrics(f);
      int posisiX = xPusat-(fm.stringWidth(s)/2);
```

```
        g.drawString("Selamat Belajar Java Applet",posisiX, yPusat);
    }
}
```



Analisis : Kode ini adalah applet Java sederhana yang digunakan untuk menampilkan teks "Selamat Belajar Java Applet" secara terpusat di area applet. Kode ini mengimpor beberapa kelas penting seperti Applet untuk dasar aplikasi, Color untuk warna teks, Font untuk gaya teks, FontMetrics untuk pengukuran teks, dan Graphics untuk menggambar teks pada applet.

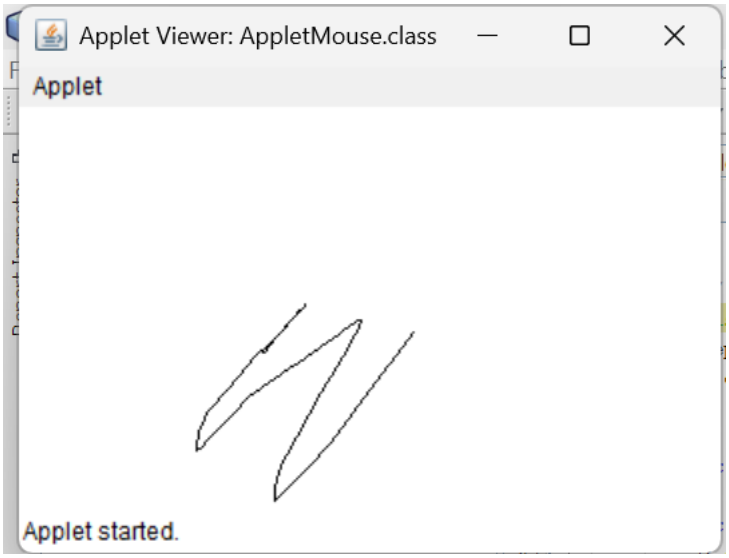
Metode paint(Graphics g) yang didefinisikan dalam kelas ini adalah inti dari penggambaran teks. Font yang digunakan adalah **sansSerif**, bergaya **BOLD**, dan berukuran 20, yang diterapkan menggunakan g.setFont(f). Warna teks diatur menjadi biru dengan g.setColor(Color.BLUE). Untuk memastikan teks berada tepat di tengah, kode ini menggunakan getSize() untuk mendapatkan lebar dan tinggi area applet, kemudian menghitung posisi horizontal teks menggunakan FontMetrics, sehingga teks selalu berada di tengah, bahkan jika ukuran jendela applet berubah. Metode drawString() akhirnya digunakan untuk menggambar teks di posisi yang telah dihitung, memberikan hasil yang terpusat secara visual.

AppletMouse

```
import java.applet.Applet;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Point;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;
```

```
import java.awt.event.MouseMotionListener;
import java.util.Vector;

/**
 *
 * @author bayui
 */
public class AppletMouse extends Applet implements MouseListener,
MouseMotionListener {
    int width, height;
    Vector<Point> listOfPositions;

    public void init() {
        width = getSize().width;
        height = getSize().height;
        setBackground(Color.white);
        listOfPositions = new Vector<Point>();
        addMouseListener(this);
        addMouseMotionListener(this);
    }

    public void mouseEntered(MouseEvent e) {}
    public void mouseExited(MouseEvent e) {}
    public void mouseClicked(MouseEvent e) {}
    public void mousePressed(MouseEvent e) {}
    public void mouseReleased(MouseEvent e) {}

    public void mouseMoved(MouseEvent e) {
        if (listOfPositions.size() >= 50) {
            // delete the first element
```

```
            listOfPositions.removeElementAt(0);
        }
        // add the new position to the list
        listOfPositions.addElement(new Point(e.getX(), e.getY()));
        repaint();
        e.consume();
    }

    public void mouseDragged(MouseEvent e) {}

    public void paint(Graphics g) {
        g.setColor(Color.black);  // Changed to black for better visibility
        for (int j = 1; j < listOfPositions.size(); ++j) {
            Point A = listOfPositions.elementAt(j-1);
            Point B = listOfPositions.elementAt(j);
            g.drawLine(A.x, A.y, B.x, B.y);
        }
    }
}
```



Analisis : Applet ini melacak pergerakan mouse dan menggambar jejak garis yang mengikuti kursor secara real-time. Setiap kali mouse bergerak, posisi kursor direkam sebagai objek Point yang disimpan dalam Vector, dengan batas maksimal 50 titik untuk menjaga performa. Jika jumlah titik melebihi batas ini, titik pertama akan dihapus untuk menjaga panjang jejak tetap konsisten. Metode mouseMoved() menambahkan titik baru ke daftar dan memanggil repaint() untuk memperbarui tampilan. Kemudian, metode paint() menggambar garis yang menghubungkan titik-titik ini secara berurutan, menciptakan efek jejak yang halus dan responsif. Latar belakang diatur menjadi putih untuk kontras yang baik dengan garis hitam, memberikan hasil visual yang sederhana namun jelas.

Applet3D

```
import java.applet.Applet;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.Image;

import java.awt.Point;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.event.MouseMotionListener;


/**

 *

 * @author Muhammad Rizal Nurfirdaus

 */


class Point3D {

   public int x, y, z;


   public Point3D(int X, int Y, int Z) {

      x = X;

      y = Y;

      z = Z;

   }
```

```
}

class Edge3D {
    public int a, b;

    public Edge3D(int A, int B) {
        a = A;
        b = B;
    }
}

public class Applet3D extends Applet implements MouseListener,
MouseMotionListener {
    int width, height;
    int mx, my;

    Image backbuffer;
    Graphics backg;

    int azimuth = 35, elevation = 30;

    Point3D[] vertices;
    Edge3D[] edges;

    public void init() {
        width = getSize().width;
        height = getSize().height;

        vertices = new Point3D[8];
        vertices[0] = new Point3D(-1, -1, -1);
```

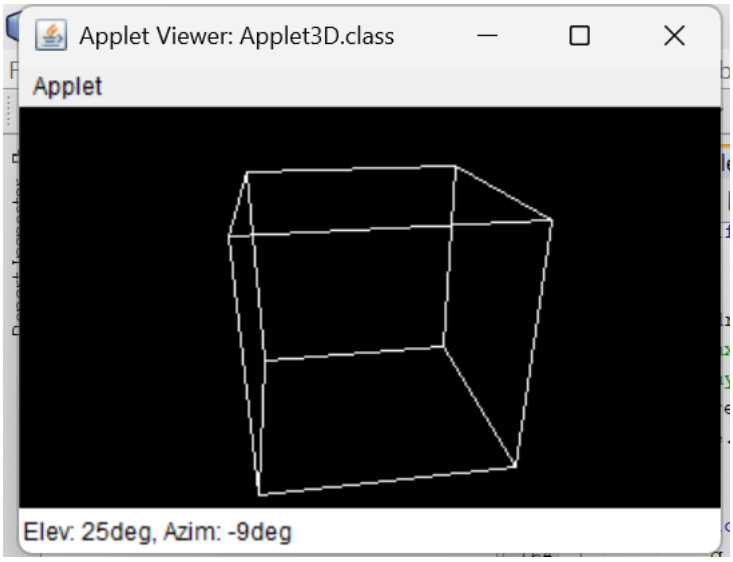
```
    vertices[1] = new Point3D(1, -1, -1);
    vertices[2] = new Point3D(1, 1, -1);
    vertices[3] = new Point3D(-1, 1, -1);
    vertices[4] = new Point3D(-1, -1, 1);
    vertices[5] = new Point3D(1, -1, 1);
    vertices[6] = new Point3D(1, 1, 1);
    vertices[7] = new Point3D(-1, 1, 1);

    edges = new Edge3D[12];
    edges[0] = new Edge3D(0, 1);
    edges[1] = new Edge3D(0, 3);
    edges[2] = new Edge3D(0, 4);
    edges[3] = new Edge3D(1, 2);
    edges[4] = new Edge3D(1, 5);
    edges[5] = new Edge3D(2, 3);
    edges[6] = new Edge3D(2, 6);
    edges[7] = new Edge3D(3, 7);
    edges[8] = new Edge3D(4, 5);
    edges[9] = new Edge3D(4, 7);
    edges[10] = new Edge3D(5, 6);
    edges[11] = new Edge3D(6, 7);

    backbuffer = createImage(width, height);
    backg = backbuffer.getGraphics();
    drawWireframe(backg);

    addMouseListener(this);
    addMouseMotionListener(this);
}
```

```
void drawWireframe(Graphics g) {
    double theta = Math.PI * azimuth / 180.0;
    double phi = Math.PI * elevation / 180.0;

    float cosT = (float) Math.cos(theta), sinT = (float) Math.sin(theta);
    float cosP = (float) Math.cos(phi), sinP = (float) Math.sin(phi);
    float cosTcosP = cosT * cosP, cosTsinP = cosT * sinP;
    float sinTcosP = sinT * cosP, sinTsinP = sinT * sinP;

    Point[] points = new Point[vertices.length];
    int scaleFactor = width / 4;
    float near = 3;
    float nearToObj = 1.5f;

    for (int j = 0; j < vertices.length; ++j) {
        int x0 = vertices[j].x;
        int y0 = vertices[j].y;
        int z0 = vertices[j].z;

        float x1 = cosT * x0 + sinT * z0;
        float y1 = -sinTsinP * x0 + cosP * y0 + cosTsinP * z0;
        float z1 = cosTcosP * z0 - sinTcosP * x0 - sinP * y0;

        x1 = x1 * near / (z1 + near + nearToObj);
        y1 = y1 * near / (z1 + near + nearToObj);

        points[j] = new Point(
                (int) (width / 2 + scaleFactor * x1 + 0.5),
```

```
                (int) (height / 2 - scaleFactor * y1 + 0.5));
        }

        g.setColor(Color.black);
        g.fillRect(0, 0, width, height);
        g.setColor(Color.white);
        for (int j = 0; j < edges.length; ++j) {
            g.drawLine(
                    points[edges[j].a].x, points[edges[j].a].y,
                    points[edges[j].b].x, points[edges[j].b].y);
        }
    }

    public void mouseEntered(MouseEvent e) {
    }

    public void mouseExited(MouseEvent e) {
    }

    public void mouseClicked(MouseEvent e) {
    }

    public void mousePressed(MouseEvent e) {
        mx = e.getX();
        my = e.getY();
        e.consume();
    }

    public void mouseReleased(MouseEvent e) {
```

```
}

public void mouseMoved(MouseEvent e) {
}

public void mouseDragged(MouseEvent e) {
    int new_mx = e.getX();
    int new_my = e.getY();
    azimuth -= new_mx - mx;
    elevation += new_my - my;

    // Clamp elevation to avoid gimbal lock
    if (elevation > 89)
        elevation = 89;
    if (elevation < -89)
        elevation = -89;

    drawWireframe(backg);
    mx = new_mx;
    my = new_my;
    repaint();
    e.consume();
}

public void update(Graphics g) {
    g.drawImage(backbuffer, 0, 0, this);
    showStatus("Elev: " + elevation + "deg, Azim: " + azimuth + "deg");
}
```

```
    public void paint(Graphics g) {
        update(g);
    }
}
```



Analisis : Program **Applet3D** adalah applet Java yang menampilkan model 3D berbentuk kubus yang dapat diputar menggunakan mouse. Program ini menggunakan dua kelas tambahan, **Point3D** untuk merepresentasikan koordinat tiga dimensi (x, y, z) dan **Edge3D** untuk mendefinisikan tepi (edge) antara dua titik dalam kubus. Saat inisialisasi (init()), program membuat delapan titik untuk sudut kubus dan 12 tepi untuk menghubungkan titik-titik ini, menghasilkan kerangka kubus. Untuk menggambar kubus, metode **drawWireframe(Graphics g)** mengonversi koordinat 3D ke 2D menggunakan transformasi sudut azimuth (rotasi horizontal) dan elevation (rotasi vertikal) untuk simulasi perspektif sederhana.

Interaksi pengguna ditangani melalui **MouseListener** dan **MouseMotionListener**, memungkinkan pengguna untuk mengubah orientasi kubus dengan menyeret mouse. Rotasi dilakukan dengan mengubah nilai azimuth dan elevation berdasarkan pergerakan mouse, kemudian menggambar ulang kubus pada buffer gambar (backbuffer) untuk memperbarui tampilannya. Metode **update(Graphics g)** memastikan bahwa gambar yang diperbarui dari buffer ditampilkan dengan lancar, sementara **paint(Graphics g)** hanya memanggil update() untuk mencegah flickering. Program ini juga membatasi elevasi antara -89 dan 89 derajat untuk menghindari **"gimbal lock"**, yaitu kehilangan derajat kebebasan pada rotasi 3D.

BAB III

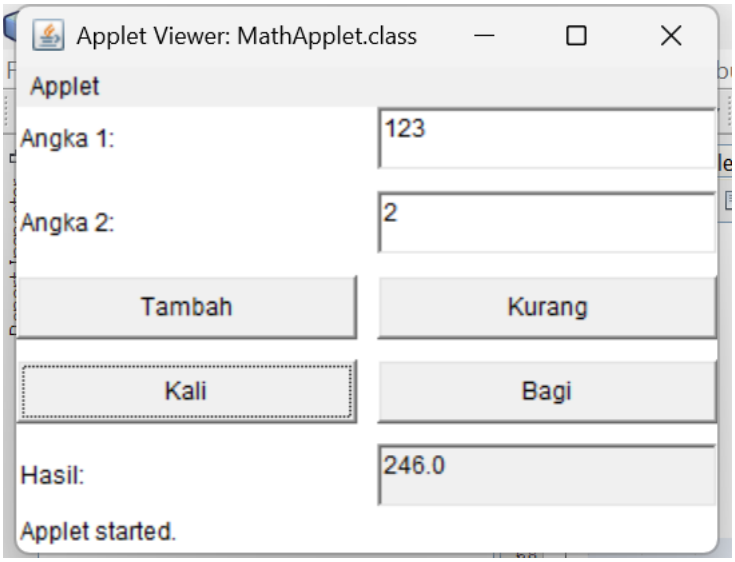
POSTTEST

Buatlah File Applet Perhitungan Matematika
(Penjumlahan/Perkalian/Pembagian/Pengurangan)

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

/**
 *
 * @author Muhammad Rizal Nurfirdaus
 */
public class MathApplet extends Applet implements ActionListener {
   TextField num1, num2, hasil;
   Button tombolTambah, tombolKurang, tombolKali, tombolBagi;

   @Override
   public void init() {
      // Atur tata letak untuk applet
      setLayout(new GridLayout(5, 2, 10, 10));

      // Buat kolom input
      num1 = new TextField();
      num2 = new TextField();
      hasil = new TextField();
      hasil.setEditable(false);

      // Buat tombol untuk setiap operasi
      tombolTambah = new Button("Tambah");
```

```
    tombolKurang = new Button("Kurang");
    tombolKali = new Button("Kali");
    tombolBagi = new Button("Bagi");

    // Tambahkan action listener ke tombol
    tombolTambah.addActionListener(this);
    tombolKurang.addActionListener(this);
    tombolKali.addActionListener(this);
    tombolBagi.addActionListener(this);

    // Tambahkan komponen ke applet
    add(new Label("Angka 1:"));
    add(num1);
    add(new Label("Angka 2:"));
    add(num2);
    add(tombolTambah);
    add(tombolKurang);
    add(tombolKali);
    add(tombolBagi);
    add(new Label("Hasil:"));
    add(hasil);
}

@Override
public void actionPerformed(ActionEvent e) {
    try {
        double angka1 = Double.parseDouble(num1.getText());
        double angka2 = Double.parseDouble(num2.getText());
        double hasilPerhitungan = 0;
```

```
            if (e.getSource() == tombolTambah) {
                hasilPerhitungan = angka1 + angka2;
            } else if (e.getSource() == tombolKurang) {
                hasilPerhitungan = angka1 - angka2;
            } else if (e.getSource() == tombolKali) {
                hasilPerhitungan = angka1 * angka2;
            } else if (e.getSource() == tombolBagi) {
                if (angka2 != 0) {
                    hasilPerhitungan = angka1 / angka2;
                } else {
                    hasil.setText("Error: Tidak bisa membagi dengan nol!");
                    return;
                }
            }
            hasil.setText(String.valueOf(hasilPerhitungan));
        } catch (NumberFormatException ex) {
            hasil.setText("Error: Masukkan angka yang valid!");
        }
    }
}
```



Analisis : Penambahan kode resetForm() dan penggunaan JOptionPane.showConfirmDialog pada method jButton1ActionPerformed merupakan langkah yang tepat untuk meningkatkan interaksi dan pengalaman pengguna. Method resetForm() berfungsi untuk mengosongkan seluruh input form, sehingga pengguna dapat langsung mengisi data baru setelah menekan tombol *Register*. Ini membuat alur pengisian data menjadi lebih efisien dan bersih setelah satu kali pendaftaran berhasil dilakukan.

Sementara itu, penggunaan JOptionPane memberikan lapisan konfirmasi tambahan sebelum data disimpan, yang dapat mencegah kesalahan akibat klik tidak disengaja. Jika pengguna memilih "Yes", maka data akan disimpan melalui method simpan() dan form langsung dikosongkan. Jika memilih "No", maka tidak ada aksi yang dilakukan. Kombinasi fitur ini tidak hanya meningkatkan keamanan input data, tetapi juga memperkuat prinsip usability dalam antarmuka aplikasi.

BAB IV

TUGAS

Buatlah File Applet Animasi dari File .gif

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Image;

/**
 *
 * @author Muhammad Rizal Nurfirdaus
 */
public class AppletAnimasiGif extends Applet {
    private Image gifImage;

    @Override
    public void init() {
        // Load .gif file from the applet code base (directory where the .class file is located)
        gifImage = getImage(getCodeBase(), "animasi.gif");
    }

    @Override
    public void paint(Graphics g) {
        // Gambar animasi di tengah applet
        int x = (getWidth() - gifImage.getWidth(this)) / 2;
        int y = (getHeight() - gifImage.getHeight(this)) / 2;
        g.drawImage(gifImage, x, y, this);
    }
```

}

Analisis : Program **AppletAnimasiGif** adalah applet Java sederhana yang digunakan untuk menampilkan animasi dari file **animasi.gif**. Program ini menggunakan kelas **Applet** dari paket **java.applet** dan kelas **Image** dari **java.awt** untuk memuat dan menggambar gambar GIF. Pada metode **init()**, file **animasi.gif** dimuat menggunakan **getImage()** yang mengambil URL dasar dari lokasi file **.class** dan menambahkan nama file **animasi.gif** sebagai parameter. Ini memungkinkan applet untuk mengambil gambar dari direktori yang sama dengan file **.class** saat dijalankan dalam konteks applet viewer atau browser.

Metode **paint(Graphics g)** digunakan untuk menggambar gambar GIF yang telah dimuat ke dalam applet. Posisi gambar dihitung untuk ditempatkan di tengah jendela applet menggunakan lebar dan tinggi komponen applet serta lebar dan tinggi gambar itu sendiri. Metode **g.drawImage()** kemudian digunakan untuk menggambar gambar pada posisi yang dihitung. Jika gambar berhasil dimuat, gambar akan ditampilkan di tengah layar, tetapi jika gagal, applet mungkin hanya akan menampilkan latar belakang kosong tanpa pesan kesalahan, karena tidak ada penanganan kesalahan eksplisit untuk kasus gagal memuat gambar dalam kode ini.

BAB V

KESIMPULAN

Dari beberapa contoh program applet Java di atas, kita dapat melihat bahwa applet adalah salah satu cara untuk membuat aplikasi grafis interaktif yang dapat dijalankan di dalam browser atau applet viewer. Program seperti **AppletAnimasiGif** digunakan untuk menampilkan animasi gambar GIF secara sederhana, dengan memanfaatkan metode **init()** untuk memuat gambar dan **paint()** untuk menampilkan gambar di layar. Pendekatan ini sangat berguna untuk aplikasi multimedia ringan.

Sementara itu, **AppletMouse** dan **Applet3D** menunjukkan penggunaan event handling untuk meningkatkan interaktivitas. **AppletMouse** memungkinkan pengguna menggambar jejak titik saat mouse bergerak, menggunakan koleksi **Vector<Point>** untuk menyimpan koordinat. Sedangkan **Applet3D** memperluas ide ini dengan merender model 3D sederhana berbentuk kubus, yang dapat diputar dengan mouse. Ini memanfaatkan matematika 3D untuk mengonversi koordinat ruang tiga dimensi ke layar dua dimensi, dan menggunakan buffering untuk meningkatkan performa grafis.

Secara keseluruhan, program-program ini menunjukkan bagaimana Java Applet dapat digunakan untuk membuat aplikasi grafis interaktif yang kompleks, meskipun teknologi ini kini kurang umum dibandingkan dengan teknologi web modern seperti HTML5, WebGL, atau Unity WebGL. Namun, pemahaman tentang applet ini tetap penting untuk memahami dasar-dasar pemrograman grafis dan interaksi pengguna dalam konteks Java.