BİLGİSAYAR PROGRAMLAMA-II- 7.ders

3D GRAFİK ÇİZİM-(Yüzey Boyama ve Gizleme) Notlar daha düzenlenecek

Yapılacaklar (Mantık)

1. **Kenarlar yerine yüzeyler (faces) tanımlanacak**
2. **Her yüzey poligon olarak çizilecek (FillPolygon)**
3. **Hangi yüzey önde / arkada → Z-derinliğe göre sıralama (Painter's Algorithm)**
4. (Opsiyonel ama önemli) **Back-face culling** → arkadaki yüzeyleri çizmemek

Küp 6 yüzeyden oluşur, her yüzey 4 noktadan oluşur:

```
int[,] DiziYuzey = new int[,]
{
   {0,1,2,3}, // Arka
   {4,5,6,7}, // Ön
   {0,1,5,4}, // Alt
   {2,3,7,6}, // Üst
   {1,2,6,5}, // Sağ
   {0,3,7,4}  // Sol
};
```

2. OnPaint İçinde Solid Çizim

Mevcut OnPaint metodunu aşağıdaki gibi değiştir:

Wireframe → Solid dönüşümü

- DrawLine yerine → FillPolygon kullandık

Painter's Algorithm (ÇOK KRİTİK)

```
float zOrtalama = zToplam / 4f;
```

- Her yüzeyin ortalama Z'sini aldık
- **Uzak olan yüzey önce çizildi**
- Yakın olan üstüne çizildi → doğru görünüm

✅ **Neden gerekli?**

Eğer sıralamazsan:

- Arka yüz öne gelir
- Görüntü "bozulur"

4. İstersen Daha İleri Seviye

Bunu daha da geliştirip:

- 🔦 Işıklandırma (Lambert shading)
- 🧭 Back-face culling (görünmeyen yüzleri çizmemek)
- 🧱 Z-buffer (gerçek 3D motor mantığı)

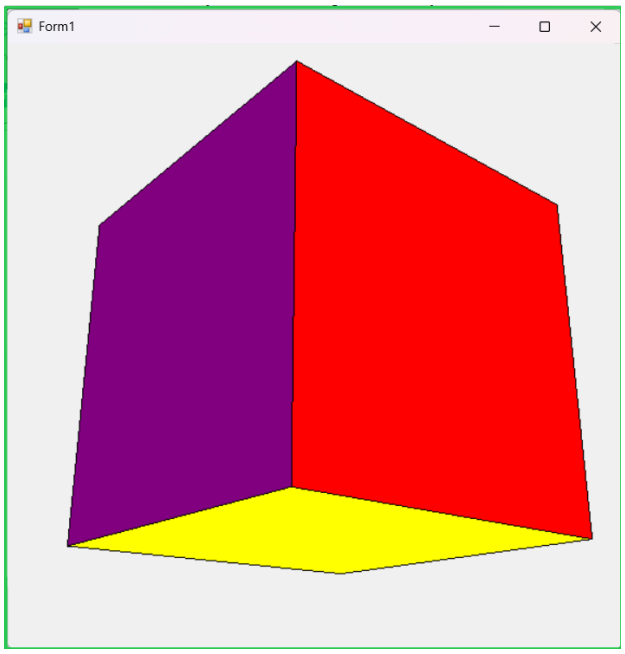
yapabiliriz.

Kritik Not

Bu sistem:

- Gerçek 3D motor DEĞİL
- Ama temel grafik pipeline'ı öğretir:
    - Model → Transform → Projection → Rasterization

Küpü **ışıklandırmalı (gölgelendirmeli)** hale getirelim
(çok güzel oluyor, direkt oyun motoru hissi verir)




```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;


namespace Uygulama_2
{
    public partial class Form1 : Form
    {
        Timer Zamanlayici = new Timer();
        float Aci = 0;

        float CisminBoyu = 1000; //Cismin en uzak iki noktası arasının uzaklığı, yani
ekrana sığacak uzaklık
```

```
Nokta3D[] DiziNokta3D = new Nokta3D[]
{
    //Orijin Küpün Tam Ortasında
    new Nokta3D(-500, -500, -500),
    new Nokta3D( 500, -500, -500),
    new Nokta3D( 500,  500, -500),
    new Nokta3D(-500,  500, -500),
    new Nokta3D(-500, -500,  500),
    new Nokta3D( 500, -500,  500),
    new Nokta3D( 500,  500,  500),
    new Nokta3D(-500,  500,  500)
};

int[,] DiziYuzey = new int[,]
{
    {0,1,2,3}, // Arka
    {4,5,6,7}, // Ön
    {0,1,5,4}, // Alt
    {2,3,7,6}, // Üst
    {1,2,6,5}, // Sağ
    {0,3,7,4}  // Sol
};

public Form1()
{
    InitializeComponent();
    this.DoubleBuffered = true;
    Zamanlayici.Interval = 30;

    Zamanlayici.Tick += (s, e) =>
    {
        Aci += 0.05f;

        this.Invalidate(); //Formun OnPaint olayı tetiklenir
    };

    Zamanlayici.Start();
}

//A-- ON PAINT OLAYI - ÇİZİMİN GERÇEKLEŞTİĞİ OLAY
protected override void OnPaint(PaintEventArgs e)
{
    Graphics g = e.Graphics;

    Nokta3D[] DonmusNoktalar = new Nokta3D[DiziNokta3D.Length];
    PointF[] DiziIzDusumNokta2D = new PointF[DiziNokta3D.Length];

    // 3D dönüş + projeksiyon
    for (int i = 0; i < DiziNokta3D.Length; i++)
    {
        var p = DiziNokta3D[i];

        p = DondurY(p, Aci);
        p = DondurX(p, Aci * 0.5f); // biraz X dönüşü de ekledik

        DonmusNoktalar[i] = p;
        DiziIzDusumNokta2D[i] = PerspektifIzDusur(p);
    }

    // Yüzeyleri Z derinliğine göre sırala (Painter's Algorithm)
    var yuzeyListesi = new List<(float z, int i)>();

    for (int i = 0; i < DiziYuzey.GetLength(0); i++)
    {
```

```
            float zToplam = 0;

            for (int j = 0; j < 4; j++)
            {
                zToplam += DonmusNoktalar[DiziYuzey[i, j]].Z;
            }

            float zOrtalama = zToplam / 4f;
            yuzeyListesi.Add((zOrtalama, i));
        }

        // Uzak olan önce çizilir
        yuzeyListesi.Sort((a, b) => b.z.CompareTo(a.z));

        // Renkler
        Brush[] fircalar = new Brush[]
        {
            Brushes.Red,
            Brushes.Green,
            Brushes.Blue,
            Brushes.Yellow,
            Brushes.Orange,
            Brushes.Purple
        };

        // Yüzeyleri çiz
        foreach (var yuzey in yuzeyListesi)
        {
            int i = yuzey.i;

            PointF[] poly = new PointF[4];

            for (int j = 0; j < 4; j++)
            {
                poly[j] = DiziIzDusumNokta2D[DiziYuzey[i, j]];
            }

            g.FillPolygon(fircalar[i], poly);
            g.DrawPolygon(Pens.Black, poly); // kenar çizgisi (isteğe bağlı)
        }
    }

    //*********** Y ETRAFINDA DÖNDÜR ***********
    Nokta3D DondurY(Nokta3D p3, float Aci)
    {
        float cos = (float)Math.Cos(Aci);
        float sin = (float)Math.Sin(Aci);

        float x = p3.X * cos + p3.Z * sin;
        float z = -p3.X * sin + p3.Z * cos;

        return new Nokta3D(x, p3.Y, z);
    }

    // X ETRAFINDA DÖNDÜR***************
    Nokta3D DondurX(Nokta3D p3, float Aci)
    {
        float cos = (float)Math.Cos(Aci);
        float sin = (float)Math.Sin(Aci);
        float y = p3.Y * cos - p3.Z * sin;
        float z = p3.Y * sin + p3.Z * cos;
        return new Nokta3D(p3.X, y, z);
    }

    PointF PerspektifIzDusur(Nokta3D p3)
```

```
        {
            // Eğer küp 0 noktasındaysa ve biz de 0 noktasındaysak düzgün göremeyiz.
            // Küpü kameradan biraz uzağa itmemiz lazım (Z ekseninde öteleme)
            float KameraUzakligi = 2500f;
            float z_derinlik = p3.Z + KameraUzakligi;

            // Odak uzaklığı (Ekranın derinlik hissi)
            float Odak = 800f;

            // Perspektif formülü: Koordinat / Derinlik
            float KucultmeFaktoru = Odak / z_derinlik;

            float x = p3.X * KucultmeFaktoru;
            float y = p3.Y * KucultmeFaktoru;

            // Ekran ortasına hizalama
            x = x + this.Width / 2;
            y = y + this.Height / 2;

            return new PointF(x, y);
        }
    }

    public class Nokta3D
    {
        public float X, Y, Z;

        public Nokta3D(float x, float y, float z)
        {
            X = x;
            Y = y;
            Z = z;
        }
    }

}
```