

GÖRÜNTÜ İŞLEME - (3.Hafta)

GEOMETRİK DÖNÜŞÜMLER

Geometrik dönüşümler resim üzerindeki her pikselin bir konumdan (x_1, y_1) başka bir konuma (x_2, y_2) haritalanmasıdır. Bununla ilgili olarak aşağıdaki işlemler kullanılabilir.

- **Taşıma (Move):** Resim içeriğinin konumunu değiştirme. Kanvas (tuval) üzerinde pikseller hareket ettirilir.
- **Aynalama (Mirror):** Resmin aynadaki görüntüsüne benzer simetrisini alma, görüntüyü takla attırma.
- **Eğme ve Kaydırma (Skew):** Resmin bir kenarından tutup kaydırma işlemidir.
- **Ölçekleme (Zoom/Scale):** Resim boyutunu değiştirme yada yaklaştırma uzaklaştırma.
- **Döndürme (Rotate):** Resmi çevirme.

Bu temel operatörler için aşağıdaki genel fonksiyonu kullanabiliriz. Bu formülde A matrisi ölçekleme, döndürme ve aynalama işlemlerini formülüne ederken, B matrisi taşıma için kullanılır. Perspektif dönüşümlerde ise her iki matris de kullanılır.

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = [A] \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + [B]$$

Geometrik dönüşümler görüntünün belli bir bölgesini yaklaştırma, yada belli bir seri resmin birleştirme için kullanılabilir. Örneğin sırayla çekilmiş resimleri tek bir resim halinde panoramik (geniş açılı resimler) haline getirmek için kullanılabilir.

TAŞIMA (Translate)

Taşıma operatörü, giriş resmindeki her pikseli, çıkış resmindeki yeni bir konuma taşıma işlemidir. Orjinal resimdeki (x_1, y_1) koordinatındaki her piksel belli bir öteleme mesafesi (β_x, β_y) boyunca taşınarak yeni konumu olan (x_2, y_2) koordinatına yerleştirilir. Taşıma işlemi görüntü üzerinde iyileştirmeler yaparken yada başka uygulamaların alt işlemleri olarak kullanılabilir.

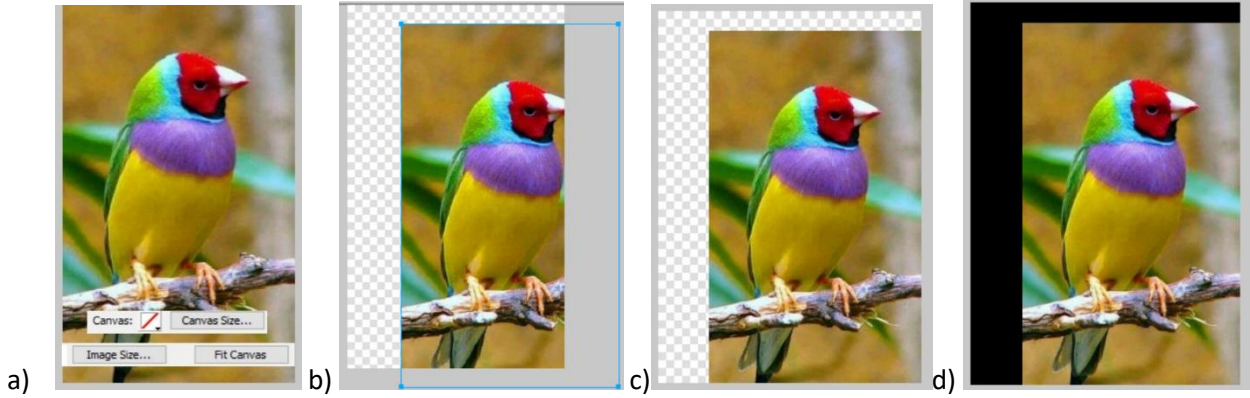
Taşıma operatörü aşağıdaki formüller kullanılarak gerçekleştirilir.

$$\begin{aligned} x_2 &= x_1 + T_x \\ y_2 &= y_1 + T_y \end{aligned}$$

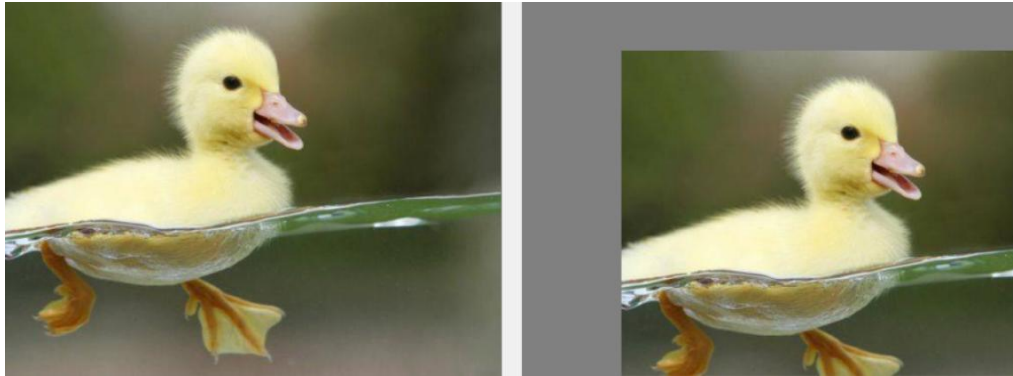
Matris formatında

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

Yeni oluşan koordinatlar (x_2, y_2) resmin sınırları dışına çıktıysa ya yok sayılır veya sınırlar genişletilerek, geride bırakılan boşluk alanları doldurulur. Bu durumda iki tane alandan bahsedebiliriz. Bir tanesi görüntünün bulunduğu resim (image) diğeri ise üzerinde kaydırıldığı zemin yani tuval (canvas) (bkz resim a). Burada resim kaydırıldığında altında kanvas da görünecektir (resim b). Kanvas yeni sınırları da kapsayacak şekilde büyütülürse resmin dışarı çıkan kısımları tekrar görünecektir fakat zeminde kanvas da görünecektir (resim c). Kanvas istenen renge boyanabilir yada renksiz düşünülebilir (resim d).



Programlama



```

public Bitmap Tasima()
{
    Color OkunanRenk;
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);
    int ResimGenisligi = GirisResmi.Width;
    int ResimYuksekligi = GirisResmi.Height;
    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    double x2 = 0, y2 = 0;

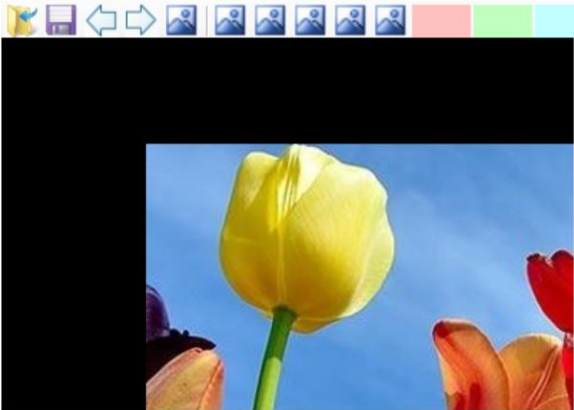
    //Taşıma mesafelerini atıyor.
    int Tx = 100;
    int Ty = 50;

    for (int x1 = 0; x1 < (ResimGenisligi); x1++)
    {
        for (int y1 = 0; y1 < (ResimYuksekligi); y1++)
        {
            OkunanRenk = GirisResmi.GetPixel(x1, y1);

            x2 = x1 + Tx;
            y2 = y1 + Ty;

            if (x2 > 0 && x2 < ResimGenisligi && y2 > 0 && y2 < ResimYuksekligi)
                CikisResmi.SetPixel((int)x2, (int)y2, OkunanRenk);
        }
    }
    pictureBox2.Image = CikisResmi;
}
    
```

Taşıma işlemini Mouse ile yaptırma



```

bool TasimaIslemi = false;

private void TASIMA_Click(object sender, EventArgs e)
{
    TasimaIslemi = true;

    ResmiDiziyeyukle();
}

ArrayList diziX = new ArrayList();
ArrayList diziY = new ArrayList();

ArrayList diziR = new ArrayList();
ArrayList diziG = new ArrayList();
ArrayList diziB = new ArrayList();

public void ResmiDiziyeyukle()
{
    Bitmap GirisResmi = new Bitmap(pictureBox1.Image);

    for (int x = 0; x < GirisResmi.Width; x++)
    {
        for (int y = 0; y < GirisResmi.Height; y++)
        {
            Color OkunanRenk = GirisResmi.GetPixel(x, y);

            diziR.Add(OkunanRenk.R);
            diziG.Add(OkunanRenk.G);
            diziB.Add(OkunanRenk.B);

            diziX.Add(x);
            diziY.Add(y);
        }
    }
}

int Tx1 = 0, Ty1 = 0, Tx2 = 0, Ty2 = 0;
private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
    if(TasimaIslemi==true)
    {
        Tx1 = e.X;
        Ty1 = e.Y;
    }
}

private void pictureBox1_MouseUp(object sender, MouseEventArgs e)

```

```

{
    if (TasimaIslemi == true)
    {
        Tx2 = e.X;
        Ty2 = e.Y;

        int Tx = Tx2 - Tx1;
        int Ty = Ty2 - Ty1;

        Tasima(Tx, Ty);
    }
}

public void Tasima(int Tx, int Ty)
{
    Bitmap GirisResmi = new Bitmap(pictureBox1.Image);
    Bitmap CikisResmi = new Bitmap(GirisResmi.Width, GirisResmi.Height); //Cikis resmi
    olusturuyor. Boyutları giriş resmi ile aynı olur. Tanımlaması globalde yapıldı.

    for (int i=0; i<diziX.Count; i++)
    {
        int x1 = Convert.ToInt32(diziX[i]);
        int y1 = Convert.ToInt32(diziY[i]);
        int R = Convert.ToInt32(diziR[i]);
        int G = Convert.ToInt32(diziG[i]);
        int B = Convert.ToInt32(diziB[i]);

        int x2 = x1 + Tx;
        int y2 = y1 + Ty;

        Color DonusenRenk = Color.FromArgb(R, G, B);

        if(x2>0 && x2<CikisResmi.Width && y2>0 && y2<CikisResmi.Height)
        {
            CikisResmi.SetPixel(x2, y2, DonusenRenk);
        }
    }
    pictureBox1.Image = CikisResmi;
}

```

AYNALAMA (Mirror/Reflect/Flip)

Yansıtma işlemi, görüntüyü orijinal resimdeki (x_1, y_1) konumundan alarak, belirlenen eksen veya bir nokta etrafında yansıtarak yeni bir konuma (x_2, y_2) yerleştirilmesidir.

Bununla ilgili yapılabilecek dönüşümler şu şekildedir.

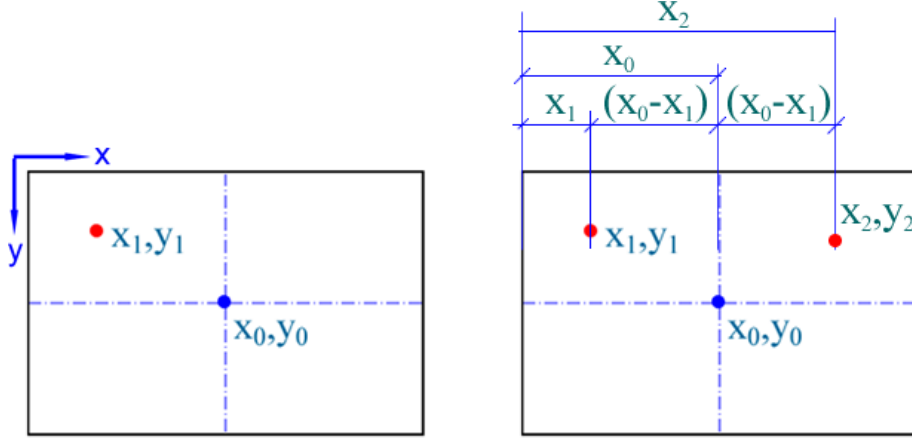
a) Ekranın orta noktasına (x_0, y_0) göre aynalama

Bunun için x_0 mesafesinden x_1 mesafesini çıkarırsak birinci koordinatın eksene uzaklığını buluruz. Bunun iki katını alıp x_1 mesafesini de eklersek x_2 koordinatının mesafesini bulmuş oluruz. y koordinatları değişmeyecektir. Ekranın ortası yerine tıklanan noktayı eksen kabul ederek aynalama da benzer şekilde yapılabilir.

$$x_2 = x_1 + 2(x_0 - x_1)$$

$$x_2 = -x_1 + 2x_0$$

$$y_2 = y_1$$



b) y_0 noktasından geçen yatay eksen etrafında aynalama

Bunun için de benzer mantık kullanılabilir. Bu durumda formüller aşağıdaki gibi olacaktır.

$$x_2 = x_1$$

$$y_2 = -y_1 + 2y_0$$

c) (x_0, y_0) noktasından geçen herhangi bir θ açısına sahip bir eksen etrafında aynalama.

$$\Delta = (x_1 - x_0) \sin\theta - (y_1 - y_0) \cos\theta$$

olmak üzere

$$x_2 = x_1 + 2 \Delta (-\sin\theta) \text{ yeni x koordinatını verir.}$$

$$y_2 = y_1 + 2 \Delta (\cos\theta) \text{ yeni y koordinatını verir.}$$

Eğer (x_0, y_0) noktası resmin merkezi olmazsa, yansıyan görüntü sınırların dışına çıkacaktır.

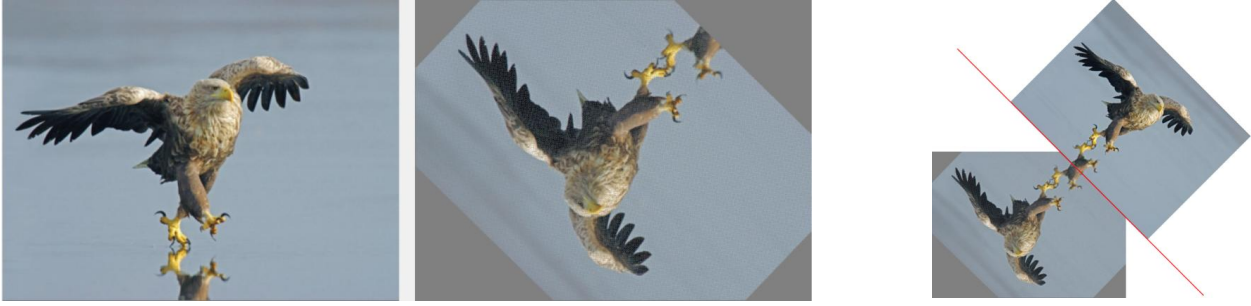
Programlama



Orta dikey eksen etrafında aynalama



Orta yatay eksen etrafında aynalama



45 derecelik eksen etrafında aynalama

```

public Bitmap Aynalama()
{
    Color OkunanRenk;
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width;
    int ResimYuksekligi = GirisResmi.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    double Aci = Convert.ToDouble(textBox1.Text);
    double RadyanAci = Aci * 2 * Math.PI / 360;

    double x2 = 0, y2 = 0;

    //Resim merkezini buluyor. Resim merkezi etrafında döndürecek.
    int x0 = ResimGenisligi / 2;
    int y0 = ResimYuksekligi / 2;

    for (int x1 = 0; x1 < (ResimGenisligi); x1++)
    {
        for (int y1 = 0; y1 < (ResimYuksekligi); y1++)
        {
            OkunanRenk = GirisResmi.GetPixel(x1, y1);

            //----A-Orta dikey eksen etrafında aynalama -----
            //x2 = Convert.ToInt16(-x1 + 2 * x0);
            //y2 = Convert.ToInt16(y1);

            //----B-Orta yatay eksen etrafında aynalama -----
            //x2 = Convert.ToInt16(x1);
            //y2 = Convert.ToInt16(-y1 + 2 * y0);

            //----C-Ortadan geçen 45 açılı çizgi etrafında aynalama-----
            double Delta = (x1 - x0) * Math.Sin(RadyanAci) - (y1 - y0) * Math.Cos(RadyanAci);

            x2 = Convert.ToInt16(x1 + 2 * Delta * (-Math.Sin(RadyanAci)));
            y2 = Convert.ToInt16(y1 + 2 * Delta * (Math.Cos(RadyanAci)));

            if (x2 > 0 && x2 < ResimGenisligi && y2 > 0 && y2 < ResimYuksekligi)
                CikisResmi.SetPixel((int)x2, (int)y2, OkunanRenk);
        }
    }
}

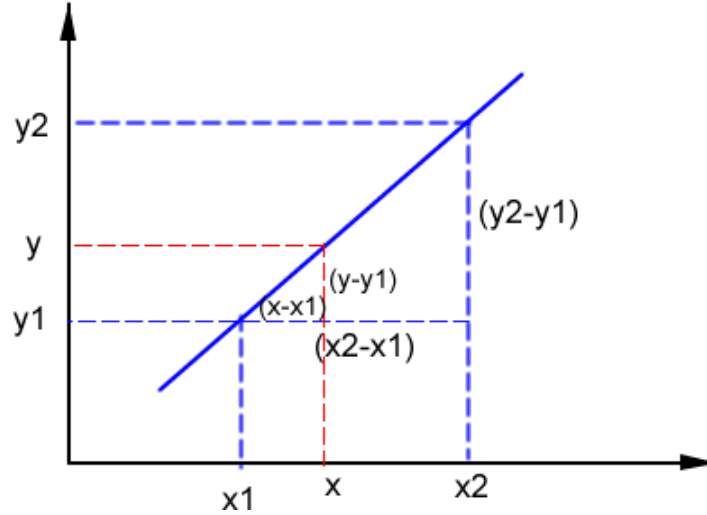
```

pictureBox2.Image = CikisResmi;

}

d) İki noktası verilen eksen etrafında aynalama: (DÜZENLE-KENDİ TARZINLA ANLAT!)

İki noktası bilinen bir doğrunun şekli aşağıda görülmektedir



Şekil. İki noktası bilinen doğrunun eğimi karşı kenarın komşu kenara oranı ile belirtilir.

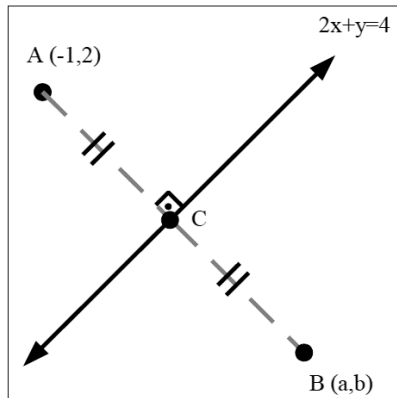
İki noktası bilinen doğrunun eğimin den hareket edersek x ve y değerleri arasındaki bağıntı bu doğrunun eğiminden bulunur. denklemini ifade eder.

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y - y_1}{x - x_1}$$

İçler dışlar çarpımı yaparak x değerini girdi y değerini çıktı olarak verirsek buna göre doğrunun denklemini aşağıdaki şekilde olacaktır. Bu denklemden sadece x ve y değerleri bilinmiyor. Diğerlerinin hepsi bilinen sayısal bir değerdir.

$$y = \frac{(y_2 - y_1)(x - x_1)}{(x_2 - x_1)} + y_1$$

Örnek: Noktanın doğruya göre simetriğinin bulunması için aşağıdaki şekilde verildiği şekliyle somut bir örnek üzerinde görelim. Buradaki doğrunun denklemini ve A noktasının koordinatları bilinmektedir. Bilinenler ışığında B noktasının koordinatlarını bulmaya çalışalım.



Bir noktanın doğruya göre simetriğinin görünümü

Doğru denkleminde doğrunun eğimi bulunmaktadır. Denklemden y yalnız bırakılır ve x 'in katsayısı eğimi ifade etmektedir. Doğrunun eğimi aşağıda verilmiştir.

$$y = 4 - 2x$$

$$m_d = -2$$

A noktasının eğime olan dik uzaklığı ile B noktasının eğime olan dik uzaklığı birbirine eşittir. Bu eşitlik yardımı ile koordinat bilgisine ulaşılabılır. A noktası ile B noktası arasındaki uzaklığın orta noktası yani C noktasının koordinatları aşağıda bulunmaktadır.

$$C\left(\frac{a-1}{2}, \frac{b+2}{2}\right)$$

C noktasının x ve y noktaları doğrunun denkleminde yerine yazılırsa aşağıdaki eşitlikler elde edilir.

$$2 \cdot \left(\frac{a-1}{2}\right) + \frac{b+2}{2} = 4$$

$$\frac{2a-2+b+2}{2} = 4$$

$$2a+b=8$$

Birbirine dik doğruların eğimleri çarpımı 1 eşittir. Bu eşitlik ile bir denklem daha türetilcektir. Öncelikle AB doğrusunun eğimi aşağıdaki gibi ifade edilmektedir.

$$m_{AB} = \frac{b-2}{a+2}$$

Doğrunun eğimi ve AB doğrusunun eğimlerinin çarpım ifadesinden aşağıdaki denklem elde edilir.

$$\frac{b-2}{a+2} \cdot -2 = -1$$

$$-2b+4 = -a-1$$

$$a-2b = -5$$

Yukarıdaki ($2a+b=8$) denklemini ile ($a-2b=-5$) eşitliği alt alta toplanarak denklemden bilinmeyen (a , b) değerleri bulunur (iki bilinmeyenin çözümü için iki denklem gerekir).

$$4a+2b=16$$

$$a-2b=-5$$

$$\hline 5a=11$$

$$a = \frac{11}{5} \rightarrow b = \frac{18}{5}$$

Bu noktalarda zaten aynalamanın yapılmış olduğu noktalar olmuş olur.

Ödev: Yukarıdaki formülleri kullanarak tıklanan iki noktayı eksen kabul eden çizginin etrafında aynalama yapan kodları geliştirin.

EĞME-KAYDIRMA (Shearing)

Resmin bir tarafı sabit dururken diğer tarafının x eksenini ya da y eksenini doğrultusunda kaydırmak için aşağıdaki matrisi kullanabiliriz.

x-eksenini doğrultusunda kaydırmak için;

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad \begin{aligned} x_2 &= x_1 + b y_1 \\ y_2 &= y_1 \end{aligned}$$



$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & -b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad \begin{aligned} x_2 &= x_1 - b y_1 \\ y_2 &= y_1 \end{aligned}$$



y-eksenini doğrultusunda kaydırmak için;

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad \begin{aligned} x_2 &= x_1 \\ y_2 &= b x_1 + y_1 \end{aligned}$$



$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -b & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad \begin{aligned} x_2 &= x_1 \\ y_2 &= -b x_1 + y_1 \end{aligned}$$



Ödev: Üst taraftan diğer yönlerde verilmeyen eğilme formüllerini çıkarın.

Programlama



```

{
    Color OkunanRenk;
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width;
    int ResimYuksekligi = GirisResmi.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    //Taşıma mesafelerini atıyor.
    double EgmeKatsayisi = 0.2;
    double x2 = 0, y2 = 0;

    for (int x1 = 0; x1 < (ResimGenisligi); x1++)
    {
        for (int y1 = 0; y1 < (ResimYuksekligi); y1++)
        {
            OkunanRenk = GirisResmi.GetPixel(x1, y1);

            // +X eksenini yönünde
            //x2 = x1 + EgmeKatsayisi * y1;
            //y2 = y1;

            // -X eksenini yönünde
            //x2 = x1 - EgmeKatsayisi * y1;
            //y2 = y1;

            // +Y eksenini yönünde
            //x2 = x1;
            //y2 = EgmeKatsayisi * x1 + y1;

            // -Y eksenini yönünde
            x2 = x1;
            y2 = -EgmeKatsayisi * x1 + y1;

            if (x2 > 0 && x2 < ResimGenisligi && y2 > 0 && y2 < ResimYuksekligi)
                CikisResmi.SetPixel((int)x2, (int)y2, OkunanRenk);
        }
    }
    pictureBox2.Image = CikisResmi;
}

```

ÖLÇEKLEME (Scaling)

Ölçekleme işlemi bir görüntünün boyutunu veya bir bölümünü küçültme yada büyültmek için kullanılır. Bu işlem görüntü üzerinde alınan bir grup matrisin pikselin değerini daha küçük yada büyük matrise dönüştürme ile gerçekleştirilir. Böylece matris büyüklüğüne göre ya pikseller çoğaltılır (yaklaştırma), yada azaltılır (uzaklaştırma).

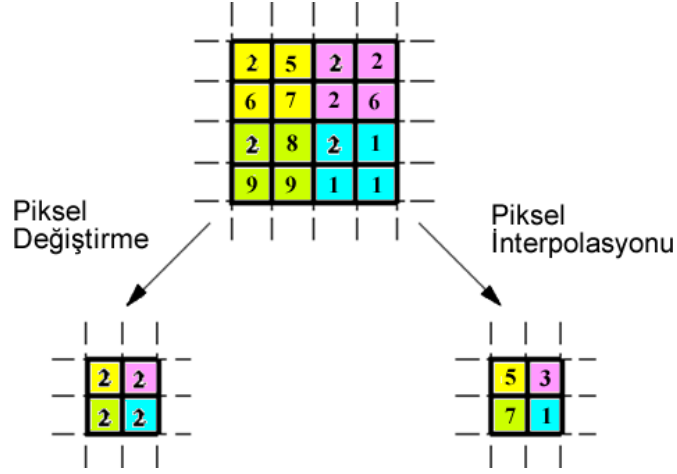
Ölçeklenen resimler büyük resimlerin küçültülerek daha küçük hafızalarda saklanması, yada büyütülerek belli bir bölgeye dikkat çekmek için kullanılabilir.

Uzaklaştırma (Küçültme)(Zoom out)

Yeni oluşturulacak görüntüdeki pikselin değeri için iki farklı teknik kullanılabilir.

a) **Piksel değiştirme:** Bir grup komşu pikselin değeri, tek bir piksel değerine dönüştürülürken içlerinden birinin değerine yada rastgele herhangi birinin değerine dönüştürülebilir. Bu metod işlemler açısından daha basittir fakat örneklenen pikseller arasındaki farklılık çok fazla ise zayıf sonuçlara yol açabilir.

b) **Piksel İnterpolasyonu:** İkinci metod ise komşu pikseller arasında istatistiksel bir örnekleme yaparak (örneğin ortalamasını alarak) oluşturulacak pikselin değerini belirler.



Şekil. Piksel değiştirmede dörtlü çerçevelerin sol üst köşelerindeki değerler kullanılmıştır. İnterpolasyonda ise dört tane pikselin ortalama değeri kullanılmıştır.

Programlama (Küçültme-Piksel Değiştirme Metodu)

Giriş resmini x ve y yönünde birer piksel atlayarak okursak ve okunan değerleri yeni resimde sırayla yerleştirirsek $1/2x$ lik bir küçültme sağlamış oluruz. Döngüdeki x ve y artış değeri küçültme katsayısı olacaktır.



```
public Bitmap Kucultme_DegistirmeMetodu()
{
    Color OkunanRenk;
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width;
    int ResimYuksekligi = GirisResmi.Height;
    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    int x2= 0, y2 = 0; //Çıkış resminin x ve y si olacak.
    int KucultmeKatsayisi = 2;
```

```

for (int x1 = 0; x1 < ResimGenisligi; x1=x1+KucultmeKatsayisi)
{
    y2 = 0;
    for (int y1 = 0; y1 < ResimYuksekligi; y1 = y1 + KucultmeKatsayisi)
    {
        OkunanRenk = GirisResmi.GetPixel(x1, y1);

        CikisResmi.SetPixel(x2, y2, OkunanRenk);
        y2++;
    }
    x2++;
}
pictureBox2.Image = CikisResmi;
}

```

Programlama (Küçültme-İnterpolasyon Metodu)

Giriş resmini x ve y yönünde küçültme katsayısı kadar atlayarak okursak ve her atladığında aradaki piksellerin ortalama değerini alırsak ve bu ortalama değerleri çıkış resmine aktarırsak 1/x lik bir küçültme sağlamış oluruz.



```

public Bitmap Kucultme_InterpolasyonMetodu()
{
    Color OkunanRenk, DonusenRenk;
    Bitmap GirisResmi, CikisResmi;
    int R = 0, G = 0, B = 0;

    GirisResmi = new Bitmap(pictureBox1.Image);
    int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı.
    int ResimYuksekligi = GirisResmi.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resminin boyutları

    int x2 = 0, y2 = 0; //Çıkış resminin x ve y si olacak.
    int KucultmeKatsayisi = 2;

    for (int x1 = 0; x1 < ResimGenisligi; x1 = x1 + KucultmeKatsayisi)
    {
        y2 = 0;
        for (int y1 = 0; y1 < ResimYuksekligi; y1 = y1 + KucultmeKatsayisi)
        {
            //x ve y de ilerlerken her atlanan pikselleri okuyacak ve ortalama değerini alacak.
            R = 0; G = 0; B = 0;
            try //resim sınırının dışına çıktığında hata vermesin diye
            {

```


Şekil. Yakınlaştırma yöntemleri. a) Piksel değeri ile çoğaltma, b) İnterpolasyon ile çoğaltma.

Programlama



```
private void button1_Click(object sender, EventArgs e)
{
    Color OkunanRenk;
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width;
    int ResimYuksekligi = GirisResmi.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    int BuyutmeKatsayisi = Convert.ToInt32(textBox1.Text);

    int x2 = 0, y2 = 0;

    for (int x1 = 0; x1 < ResimGenisligi; x1++)
    {
        for (int y1 = 0; y1 < ResimYuksekligi; y1++)
        {
            OkunanRenk = GirisResmi.GetPixel(x1 , y1 );

            for (int i = 0; i < BuyutmeKatsayisi; i++)
            {
                for (int j = 0; j < BuyutmeKatsayisi; j++)
                {
                    x2 = x1* BuyutmeKatsayisi + i;
                    y2 = y1 * BuyutmeKatsayisi + j;

                    if (x2 > 0 && x2 < ResimGenisligi && y2 > 0 && y2 < ResimYuksekligi)
                        CikisResmi.SetPixel(x2 , y2 , OkunanRenk);

                }
            }
        }
    }

    pictureBox2.Image = CikisResmi;
}
```

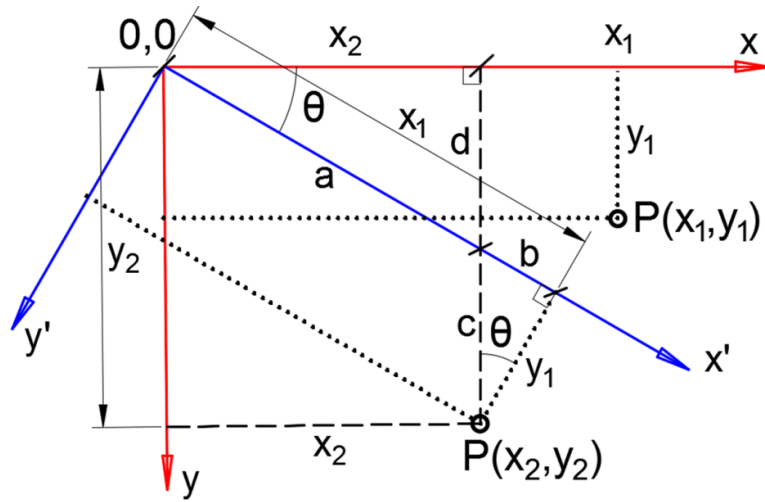
Ödev: Yukarıda kodlar Piksel değiştirme yöntemiyle yazılmıştır. Piksel interpolasyon yönteminin kodlarını siz kendiniz yazın.

DÖNDÜRME (Rotation)

Döndürme işlemi bir nokta etrafında belli bir açı (θ) değerinde çevirerek giriş resmindeki (x_1,y_1) koordinatını çıkış resmindeki (x_2,y_2) noktasına taşıma işlemidir. Çoğu döndürme işleminde sınırların dışına çıkan kısımlar yok sayılır.

Döndürme işlemi çoğunlukla resmin görünümü daha iyi hale getirmek için kullanılabilir. Afin işlemlerinde (perspektif bakış) de kullanılan bir işlem olacaktır.

Döndürme işlemi için formüllerimizi çıkaralım. Bir orijin $(0,0)$ etrafında x - y eksen takımını saat yönünde θ açısı kadar döndürdüğümüzde şekildeki gibi olacaktır. Burada x_1 ve y_1 değerleri ile θ açısı bilenen değerlerdir. x_2 ve y_2 değerlerini bulmaya çalışalım. θ açısının bulunduğu üçgenlerin kenarlarına a,b,c,d sembollerini atarsak aşağıdaki formüllerimiz çıkacaktır.



$$\tan\theta = \frac{d}{x_2} \rightarrow d = x_2 \tan\theta \rightarrow d = x_2 \frac{\sin\theta}{\cos\theta}$$

$$\tan\theta = \frac{b}{y_1} \rightarrow b = y_1 \tan\theta \rightarrow b = y_1 \frac{\sin\theta}{\cos\theta}$$

$$\sin\theta = \frac{d}{a} \rightarrow a = \frac{d}{\sin\theta} \rightarrow a = \frac{(x_2 \frac{\sin\theta}{\cos\theta})}{\sin\theta} \rightarrow$$

$$a = x_2 \frac{1}{\cos\theta}$$

$$\sin\theta = \frac{b}{c} \rightarrow c = \frac{b}{\sin\theta} \rightarrow c = \frac{(y_1 \frac{\sin\theta}{\cos\theta})}{\sin\theta} \rightarrow$$

$$c = y_1 \frac{1}{\cos\theta}$$

$$x_1 = a + b \rightarrow x_1 = x_2 \frac{1}{\cos\theta} + y_1 \frac{\sin\theta}{\cos\theta} \rightarrow$$

$$x_2 \frac{1}{\cos\theta} = x_1 - y_1 \frac{\sin\theta}{\cos\theta} \rightarrow$$

$$\boxed{x_2 = x_1 \cos\theta - y_1 \sin\theta}$$

$$y_2 = c + d \rightarrow y_2 = y_1 \frac{1}{\cos\theta} + x_2 \frac{\sin\theta}{\cos\theta} \rightarrow$$

$$y_2 = y_1 \frac{1}{\cos\theta} + (x_1 \cos\theta - y_1 \sin\theta) \frac{\sin\theta}{\cos\theta} \rightarrow$$

$$y_2 = y_1 \frac{1}{\cos\theta} + x_1 \sin\theta - y_1 \frac{\sin^2\theta}{\cos\theta} \rightarrow$$

$$y_2 = x_1 \sin\theta + y_1 \left(\frac{1}{\cos\theta} - \frac{\sin^2\theta}{\cos\theta} \right) \rightarrow$$

$$y_2 = x_1 \sin\theta + y_1 \left(\frac{\cos^2\theta}{\cos\theta} \right) \rightarrow$$

$$\boxed{y_2 = x_1 \sin\theta + y_1 \cos\theta}$$

Bu iki denklemi matris formunda yazarsak, orijin etrafında (resmin sol üst köşesi) bir noktanın dönüşü şu şekilde olur.

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

Pozitif θ açısı saatin yönünde dönmeyi ifade etmektedir. Resim alanı içerisindeki görüntü döndürülürken döndürme noktası sol üst köşe yerine resmin ortasını almak gerekir. Bu durumda koordinatlardan resmin orta noktasını gösteren x_0 ve y_0 değerlerini çıkarmak gerekir. Yani ilk işlemde orijin resmin ortasındaymiş gibi düşünüp, dönme işlemi gerçekleştikten sonra, resmi sol üst köşeye taşırsak x_0 ve y_0 değerlerini çıkarırız. Resimlerde y koordinat eksenini aşağıya doğru bakmaktadır. Buna göre kullanacağımız formülleri çıkarırsak, şu şekilde olacaktır.

$$x_2 = x_1 \cos\theta - y_1 \sin\theta$$

$$(x_2 - x_0) = (x_1 - x_0) \cos\theta - (y_1 - y_0) \sin\theta$$

$$\boxed{x_2 = (x_1 - x_0) \cos\theta - (y_1 - y_0) \sin\theta + x_0}$$

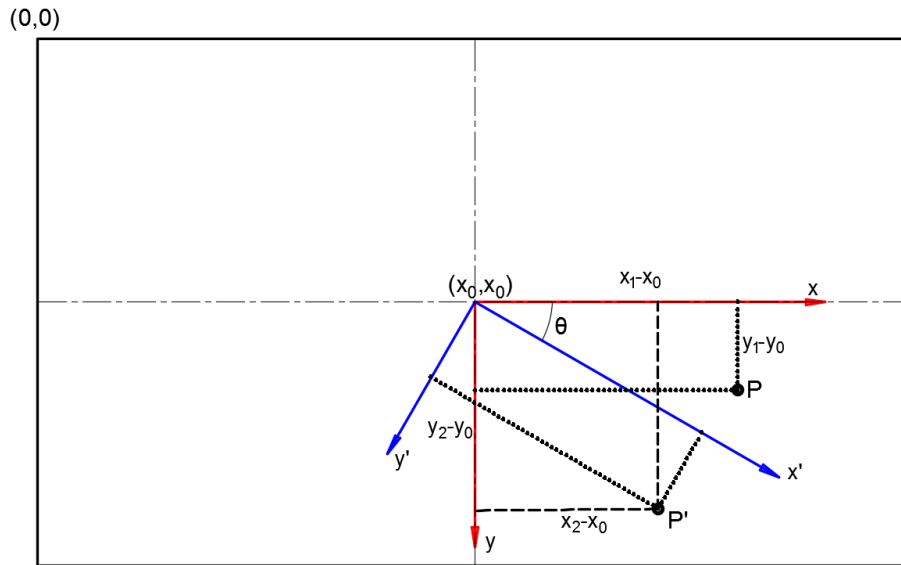
$$y_2 = x_1 \sin\theta + y_1 \cos\theta$$

$$(y_2 - y_0) = (x_1 - x_0) \sin\theta + (y_1 - y_0) \cos\theta$$

$$\boxed{y_2 = (x_1 - x_0) \sin\theta + (y_1 - y_0) \cos\theta + y_0}$$

Bu iki denklemi matris formunda yazarsak, resmin ortasındaki (x_0, y_0) noktası etrafında, (x_1, y_1) noktasını çevirdiğimizde (x_2, y_2) yeni konumuna şu formülle döndürülür.

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} (x_1 - x_0) \\ (y_1 - y_0) \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$



Döndürme işlemi nedeniyle resmin orijinal boyutunun dışına çıkan kısımlar yok sayılabilir yada bu kısımlar siyah renge dönüştürülebilir.

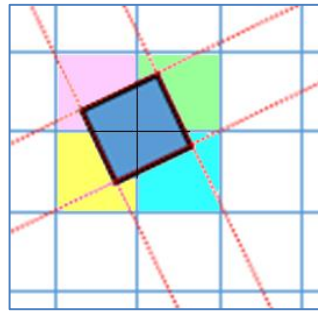
Bu formüller kullanıldığında x_2 ve y_2 koordinat değerleri, formül içindeki Sin ve Cos değerlerinden dolayı ondalık sayı olarak çıkacaktır. Bu durumda dönmüş olan piksel değeri, resim üzerindeki yatay ve dikey konumda olan piksel ızgarasının üzerine tam oturmayacaktır. Aynı anda birkaç pikselin üzerine basacaktır. Eğer en yakın tam sayı değerine yuvarlatılsa bile bazı noktalara renk ataması yapılamayacaktır. Bazı pikseller çift adreslenirken

bazıları kaçırılacaktır. Bu durumda resim üzerinde boşluklar gözükcektir. Bu bozucu duruma **alias** (aliasing) denir.



Bu sorunu çözmek için değişik yöntemler kullanılabilir. Bunlardan üç tanesi şu şekildedir. Başka yöntemlerde geliştirilebilir.

- Kaynak resim üzerindeki pikseller aşırı örneklenecek büyütülebilir. Yani her piksel aynı renkte olmak üzere $n \times n$ lik küçük bir matris şeklinde (örn: 2×2) **büyütülebilir**. Bu haliyle resim döndürülür ve **ardından küçültülürse** aradaki boşluklar kaybolacaktır.
- İkinci yöntemde ise sorunu tersine çevirebiliriz. Hedef piksel üzerine basan 4 tane kaynak pikselin hangisi ağırlıkça en fazla yer işgal ediyorsa onun rengi atanabilir. Ayrıca bu algoritmaya üzerine basan 4 tane pikselin alan ağırlığı ile doğru orantılı olacak şekilde ortalama bir renk değeri atanabilir. Bu durum daha pürüzsüz bir görüntünün oluşmasını sağlayacaktır fakat hesaplama zamanı artacaktır.



- Üçüncü yöntem ise daha ilginç bir yöntemdir. Bu yöntemde tüm pikseller önce yatay olarak sağa doğru kaydırılır. Daha sonra dikey olarak aşağı doğru kaydırılır. Sonra tekrar sağa doğru yatayda kaydırılırsa resim dönmüş olarak gözükcektir.



Sağa doğru kaydırma;

$$x_2 = (x_1 - x_0) - (y_1 - y_0) \tan(\theta/2) + x_0$$

$$y_2 = (y_1 - y_0) + y_0$$

Aşağı kaydırma;

$$x_2 = (x_1 - x_0) + x_0$$

$$y_2 = (x_1 - x_0) \sin\theta + (y_1 - y_0) + y_0$$

Tekrar Sağa doğru kaydırma;

$$x_2 = (x_1 - x_0) - (y_1 - y_0) \tan(\theta/2) + x_0$$

$$y_2 = (y_1 - y_0) + y_0$$

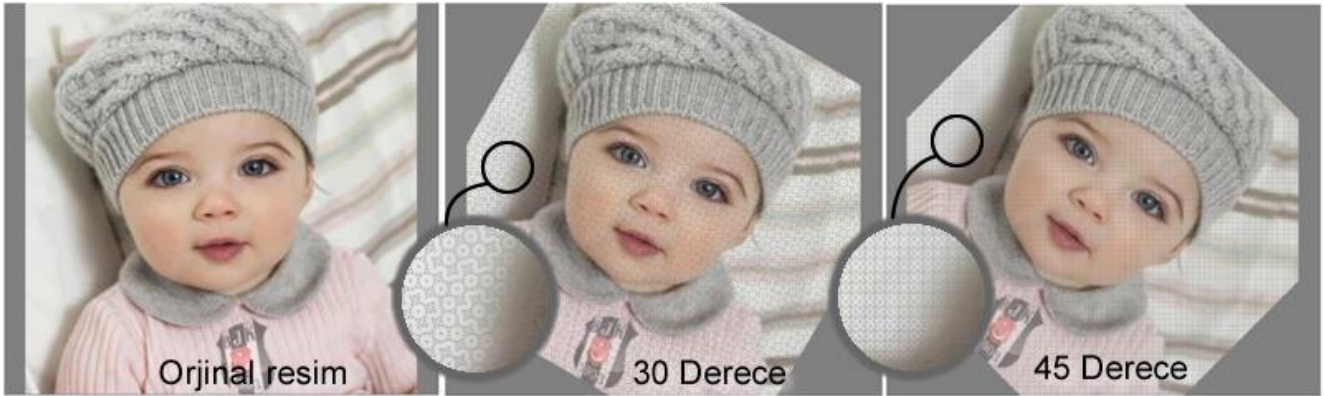
Bu işlemleri matris formatında gösterirsek topluca şu şekilde olacaktır.

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 - x_0 \\ y_1 - y_0 \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

Araştırma: Buradaki Kaydırma işlemi ile Döndürme işlemi arasındaki bağlantıların nasıl bulunduğunu çıkarın.

Programlama (Alias düzeltme yok)

Alias düzeltilmeden sadece resmi döndürmeye ait kodlar aşağıdadır. Resim üzerinde farklı açılardaki alias oluşumları gösterilmiştir.



```
public Bitmap Dondurme()
{
    Color OkunanRenk;
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width;
    int ResimYuksekligi = GirisResmi.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    int Aci = Convert.ToInt16(textBox1.Text);
    double RadyanAci = Aci * 2 * Math.PI / 360;

    double x2 = 0, y2 = 0;

    //Resim merkezini buluyor. Resim merkezi etrafında döndürecek.
    int x0 = ResimGenisligi / 2;
    int y0 = ResimYuksekligi / 2;

    for (int x1 = 0; x1 < (ResimGenisligi); x1++)
    {
        for (int y1 = 0; y1 < (ResimYuksekligi); y1++)
        {
            OkunanRenk = GirisResmi.GetPixel(x1, y1);

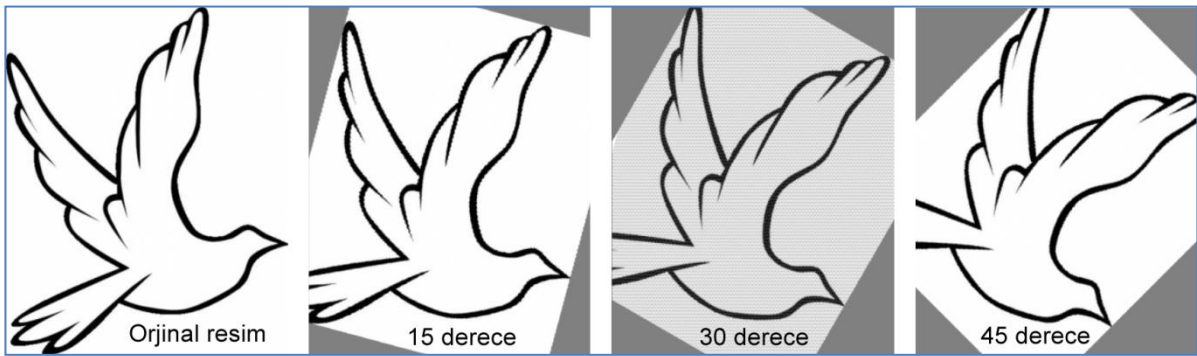
            //Döndürme Formülleri
```

```
x2 = Math.Cos(RadyanAci) * (x1 - x0) - Math.Sin(RadyanAci) * (y1 - y0) + x0;
y2 = Math.Sin(RadyanAci) * (x1 - x0) + Math.Cos(RadyanAci) * (y1 - y0) + y0;
```

```
if(x2>0 && x2<ResimGenisligi && y2>0 && y2<ResimYuksekligi )
    CikisResmi.SetPixel((int)x2, (int)y2, OkunanRenk);
}
}
pictureBox2.Image = CikisResmi;
}
```

Programlama (Alias düzeltme, kaydırma yöntemi ile)

Kaydırma yöntemi alias hatası düzelse de 30 derecede hala sorun devam etmektedir. Ondalık sayıların tam sayıya çevrilmesi bu derecede hataya neden olmaktadır.



```
public Bitmap Dondurme_Alias()
{
    Color OkunanRenk;
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox1.Image);

    int ResimGenisligi = GirisResmi.Width;
    int ResimYuksekligi = GirisResmi.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    int Aci = Convert.ToInt16(textBox1.Text);
    double RadyanAci = Aci * 2 * Math.PI / 360;

    double x2 = 0, y2 = 0;

    //Resim merkezini buluyor. Resim merkezi etrafında döndürecek.
    int x0 = ResimGenisligi / 2;
    int y0 = ResimYuksekligi / 2;

    for (int x1 = 0; x1 < (ResimGenisligi); x1++)
    {
        for (int y1 = 0; y1 < (ResimYuksekligi); y1++)
        {
            OkunanRenk = GirisResmi.GetPixel(x1, y1);

            //Aliaslı Döndürme -Sağa Kaydırma
            x2 = (x1 - x0) - Math.Tan(RadyanAci / 2) * (y1 - y0) + x0;
            y2 = (y1 - y0) + y0;
```

```

x2 = Convert.ToInt16(x2);
y2 = Convert.ToInt16(y2);

//Aliaslı Döndürme -Aşağı kaydırma
x2 = (x2 - x0) + x0;
y2 = Math.Sin(RadyanAcı) * (x2 - x0) + (y2 - y0) + y0;

x2 = Convert.ToInt16(x2);
y2 = Convert.ToInt16(y2);

//Aliaslı Döndürme -Sağa Kaydırma
x2 = (x2 - x0) - Math.Tan(RadyanAcı / 2) * (y2 - y0) + x0;
y2 = (y2 - y0) + y0;

x2 = Convert.ToInt16(x2);
y2 = Convert.ToInt16(y2);

if (x2 > 0 && x2 < ResimGenisligi && y2 > 0 && y2 < ResimYuksekligi)
    CikisResmi.SetPixel((int)x2, (int)y2, OkunanRenk);
}
}
pictureBox2.Image = CikisResmi;
}

```

Araştırma: Yukarıda Alias düzeltme yapıldığı halde 30 derece neden hala boşlukların çıktığını araştırın.

ALIAS DÜZELTME-Kendinden önceki pikselleri tarama

Not: Her pikselin kendinden önceki pikselleri kontrol et. Bunlar siyahsa kendini rengini ata. Her döngüde 2 şer adım ilerle..

```

private void ALIAS_DUZETME_Click(object sender, EventArgs e)
{
    Bitmap GirisResmi, CikisResmi;
    GirisResmi = new Bitmap(pictureBox2.Image);

    int ResimGenisligi = GirisResmi.Width;
    int ResimYuksekligi = GirisResmi.Height;

    CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

    Color DonusenRenk, OkunanRenk2, OkunanRenk1x, OkunanRenk1y, OkunanRenk1xy;

    for (int x = 1; x < (ResimGenisligi); x = x + 2)
    {
        for (int y = 1; y < (ResimYuksekligi); y = y + 2)
        {
            OkunanRenk2 = GirisResmi.GetPixel(x, y);
            OkunanRenk1x = GirisResmi.GetPixel(x - 1, y);
            OkunanRenk1y = GirisResmi.GetPixel(x, y - 1);
            OkunanRenk1xy = GirisResmi.GetPixel(x - 1, y - 1);

            if (OkunanRenk2.R != 0 && OkunanRenk2.G != 0 && OkunanRenk2.B != 0)
            {

                if (OkunanRenk1x.R == 0 && OkunanRenk1x.G == 0 && OkunanRenk1x.B == 0)
                {
                    DonusenRenk = OkunanRenk2;
                    CikisResmi.SetPixel(x - 1, y, DonusenRenk);
                }
            }
        }
    }
}

```

```

    }
    if (OkunanRenk1y.R == 0 && OkunanRenk1y.G == 0 && OkunanRenk1y.B == 0)
    {
        DonusenRenk = OkunanRenk2;
        CikisResmi.SetPixel(x, y - 1, DonusenRenk);
    }

    if (OkunanRenk1xy.R == 0 && OkunanRenk1xy.G == 0 && OkunanRenk1xy.B == 0)
    {
        DonusenRenk = OkunanRenk2;
        CikisResmi.SetPixel(x - 1, y - 1, DonusenRenk);
    }
}
}
pictureBox3.Image = CikisResmi;}

```

DÖNDÜRME İŞLEMİ-TERS DÖNÜŞÜM FORMÜLÜ

`CikisResmi.SetPixel((int)x2, (int)y2, OkunanRenk);` şeklinde hesaplandığında, (x_2, y_2) yuvarlandığında bazı koordinatlar atlanıyor. Sonuç resimde siyah delikler yani boşluklar oluşuyor. Çözüm Geriye doğru dönüşüm formülü (Backward Mapping). Doğru yöntem, hedef piksellerin (çıkış resminin) her birinin hangi kaynak piksele denk geldiğini hesaplamaktır. Yani `CikisResmi` pikselinden `GirisResmi` pikseline doğru dönüşüm yapılmalıdır. Şu şekildedir.

Matematiksel denklem formatında şu şekilde yazılır:

$$x_1 = \cos(\theta) (x_2 - x_0) + \sin(\theta) (y_2 - y_0) + x_0$$

$$y_1 = -\sin(\theta) (x_2 - x_0) + \cos(\theta) (y_2 - y_0) + y_0$$

Burada:

- (x_1, y_1) : Giriş (orijinal) resmindeki koordinat,
- (x_2, y_2) : Çıkış (döndürülmüş) resmindeki koordinat,
- (x_0, y_0) : Döndürme merkezinin koordinatı (genellikle resmin ortası),
- θ : Döndürme açısı (radyan cinsinden).

Bu ters dönüşüm, her hedef pikselin hangi kaynak pikselden geldiğini bulmak için kullanılır. İleri dönüşümün (forward mapping) tersi olup, boş piksel (siyah nokta) oluşmasını engeller.



30
0
0
0

```

private void btnDONDURME_Click(object sender, EventArgs e)
{
    Bitmap GirisResmi = new Bitmap(pictureBox1.Image);

```

```

int ResimGenisligi = GirişResmi.Width;
int ResimYuksekligi = GirişResmi.Height;
Bitmap CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi);

int Aci = Convert.ToInt16(textBox1.Text);
double RadyanAci = Aci * Math.PI / 180.0; // düzeltme: 2π/360 yerine π/180

int x0 = ResimGenisligi / 2;
int y0 = ResimYuksekligi / 2;

for (int x2 = 0; x2 < ResimGenisligi; x2++)
{
    for (int y2 = 0; y2 < ResimYuksekligi; y2++)
    {
        // Ters dönüşüm formülleri
        double x1 = Math.Cos(RadyanAci) * (x2 - x0) + Math.Sin(RadyanAci) * (y2 - y0)
+ x0;
        double y1 = -Math.Sin(RadyanAci) * (x2 - x0) + Math.Cos(RadyanAci) * (y2 - y0)
+ y0;

        if (x1 >= 0 && x1 < ResimGenisligi && y1 >= 0 && y1 < ResimYuksekligi)
        {
            Color OkunanRenk = GirişResmi.GetPixel((int)x1, (int)y1);
            CikisResmi.SetPixel(x2, y2, OkunanRenk);
        }
        else
        {
            // Arka planı beyaz yap (isteğe bağlı)
            CikisResmi.SetPixel(x2, y2, Color.White);
        }
    }
}

pictureBox2.Image = CikisResmi;
}

```

Yukarıdaki çözümde hâlâ hafif pikselleşme olabilir. Bunu düzeltmek için (int)x1 yerine komşu dört pikselin ortalamasını alabilirsin (Bilinear Interpolation (yumuşatma))

KIRPMA

Kırpma işleminin mantığını ve algoritmasını kendiniz geliştirin. Ödev olarak istenecektir.

----- 000 -----

Haftalık Ödev: Aşağıda verilen Ö1, Ö3 ve Ö4 ödevlerinin hepsinin uygulamasını yapınız.

----- 000 -----

Ödev 1: a) **Tıklayarak Resmi Taşıma:** Resim üzerine bir noktaya tıklansın. Ardından ikinci noktaya tıkladığında resmin o noktası yeni yerine gelecek şekilde resmi taşıyın. Tıklanan noktaların gözle görülebilmesi için resim üzerinde küçük bir kare çizdirin. (PictureBox üzerinde çizgi nasıl çizdirilir öğrenmek için İnt.Tab.Prg 4. Notlara bakın)

b) **Basılı Tutarak Resmi Taşıma:** Bu işlemin aynısı mouse basılı iken de yapılabilir. Mouse basılı iken her hareket ettiği nokta alınıp resim sürekli mouse takip edecek şekilde taşınabilir.

c) **Formül Çıkarma:** Aynalama komutlarından Resmin orta noktasına göre (x0,y0) belli bir açıda (θ) aynalarken kullanılan aşağıdaki formüllerin çıkarılışını bulunuz. (Bilenler x1, y1, x0, y0 dir. Bulunacak olan x2, y2) (Örnek

olarak döndürme işlemlerinin içindeki formül çıkarma yöntemini kullanabilirsiniz). Bu formüllerin doğruluğunu kontrol ediniz, nasıl çalıştığını gösteriniz.

(x_0, y_0) noktasından geçen herhangi bir θ açısına sahip bir eksen etrafında aynalama.


$$\Delta = (x_1 - x_0) \sin\theta - (y_1 - y_0) \cos\theta$$

olmak üzere

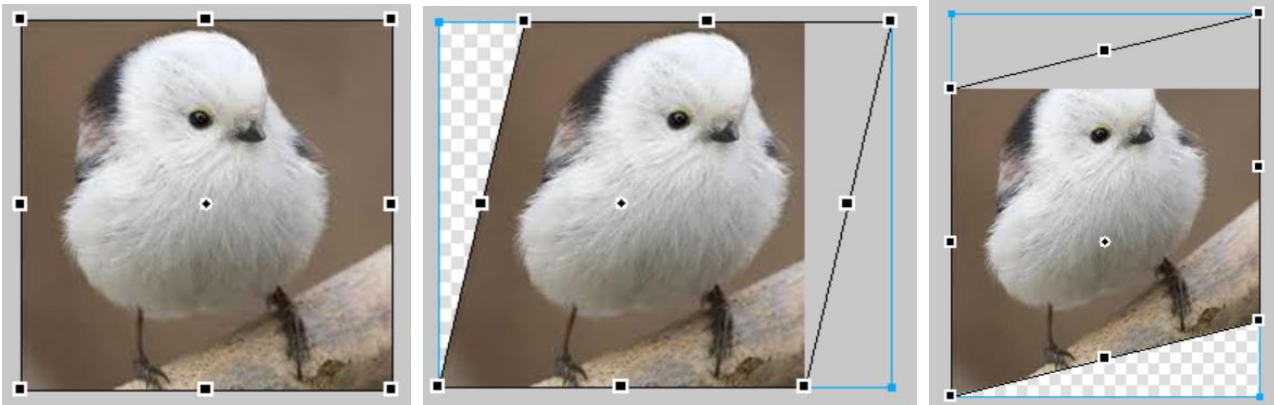
$$x_2 = x_1 + 2 \Delta (-\sin\theta) \text{ yeni x koordinatını verir.}$$

$$y_2 = y_1 + 2 \Delta (\cos\theta) \text{ yeni y koordinatını verir.}$$

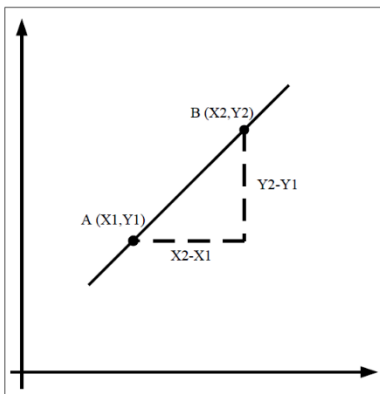
Eğer (x_0, y_0) noktası resmin merkezi olmazsa, yansıyan görüntü sınırların dışına çıkacaktır.

Ödev 2: Kaydırma komutuna  tıkladığında resmin köşelerinden tutulup çekebilecek şekilde kaydırma işlemi yapın. Örnek görünümü aşağıdakine benzer olsun. Burada olduğu gibi tüm köşelerden tutmak yerine üst kenara yakın bölgede mouse basılı iken yana doğru sürüklemeye yaparsak, resim sağa doğru kaysın. Yan kenara yakın bir bölgede mouse basılı iken aşağı doğru sürüklersem resim aşağı doğru kaysın. Bu şekilde 4 kenar üzerinde de bunu yapabilelim. Her iki yönde kayacak şekilde ayarlayabilelim.

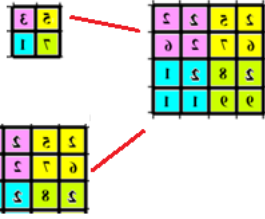
Bir resim yana kaydırıldıktan sonra o resim tekrar aynı picturebox üzerinde gözükmeli. Yani yan tarafta değil hepsi picturebox1 üzerinde olmalı. Böylece tekrardan eğik resim üzerinde ikinci bir kaydırma işlemi yapılabilir. Fakat bu esnada dışarı taşan piksellerin bilgisi kaybolacağından bunları kaybetmemek için tüm resim üzerindeki piksel değerlerinin hem koordinatları hemde renk bilgileri dizide tutulabilir.



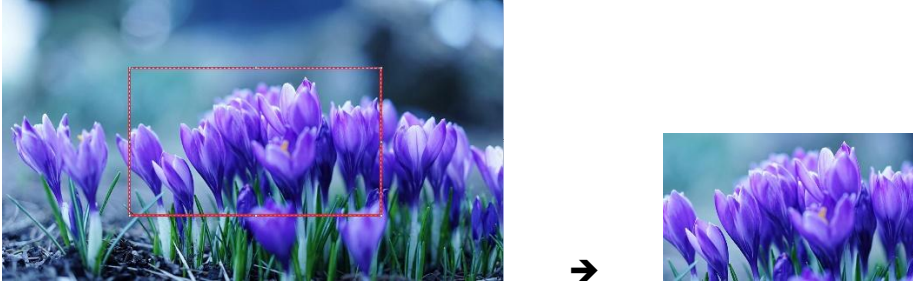
Ödev 3: Tıklanan iki noktayı eksen kabul eden çizginin etrafında aynalama yapan programı yazınız. Formüller ders notları içerisinde vardır.



Ödev 4: Küçük bir resmi belli oranlarda büyütebilen bir program yazınız. Resim büyütme esnasında aralarda oluşan piksel boşlukları (alias) için çözümde bulmaya çalışın.

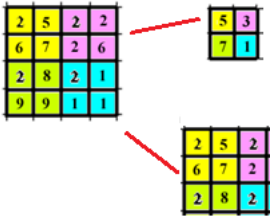


Ödev 5: Resim üzerinde mouse ile tıklanan iki noktanın arasında oluşan dikdörtgeni Kırpma aracı olarak kullanan programı yazın. Yani çizilen dikdörtgenin dışındaki kısımlar ikinci resimde olmayacak.




----- 000-----

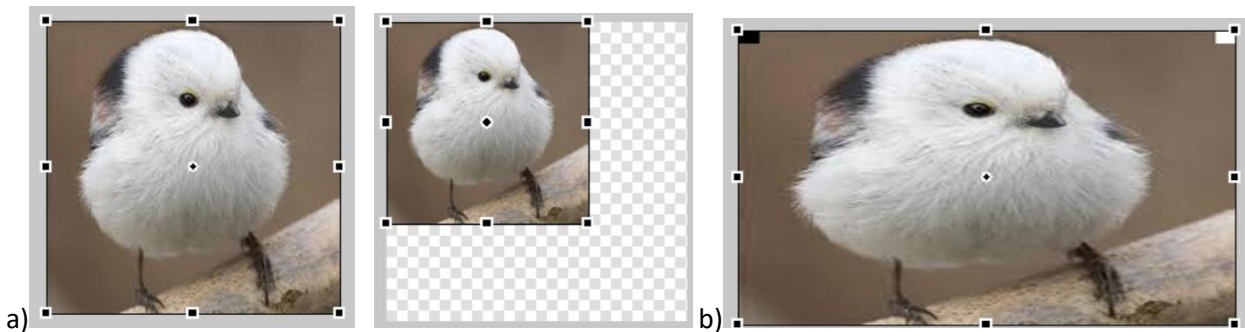
Ödev 6: Bir resmi yukarıda anlatıldığı gibi yarı yarıya küçültme değil belli oranlarda küçültme yapabilecek programı yazınız. %70, %30 gibi oranlarda da küçültme yapabilsin.



Ödev 7: Resmin ortasında tıklanan herhangi bir nokta etrafında textbox'tan girilen açı kadar çevirme yapan kodları yazın. Resmin ortasında tıklanan yerde küçük bir artı şeklinde işaret çizdirin. Dönme işleminden sonra bu işaret kalkmalı.

Ödev 8: Ölçekleme komutuna  tıkladığında resmin köşelerinde küçük kutucuklar oluşsun.

- Mouse ile bu noktadan basarak sürüklenip bırakıldığında, bırakılan nokta ile köşe arasında resim küçülmüş olsun. Bu esnada orantısı bozulmasın. Aynı işlemi büyütme içinde yapın.
- Köşe yerine kenarların ortasındaki noktalardan çekildiğinde resim yatay / dikey uzasın (orantısı bozulsun).



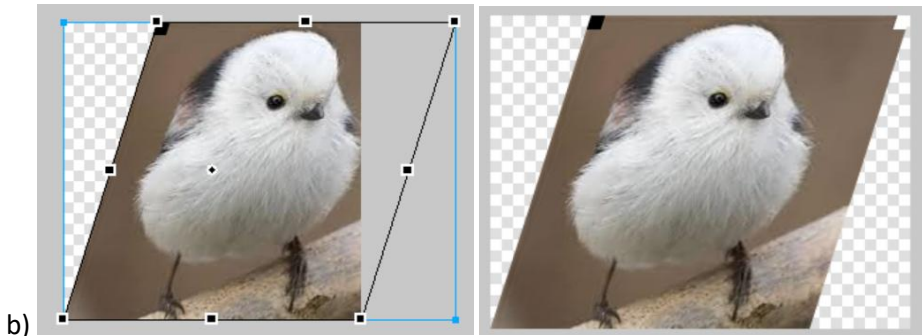
Ödev 9: Ölçekleme işlemlerinden büyültme işlemini hem "Piksel Değiştirme" metodu ile hem de "Piksel İnterpolasyon" metodu ile programlayın. Hangisinin daha iyi bir sonuç verdiğini görmek için print-screen ile alarak Paintte büyütüp inceleyin. Yanyana getirip kenarları karşılaştırın.

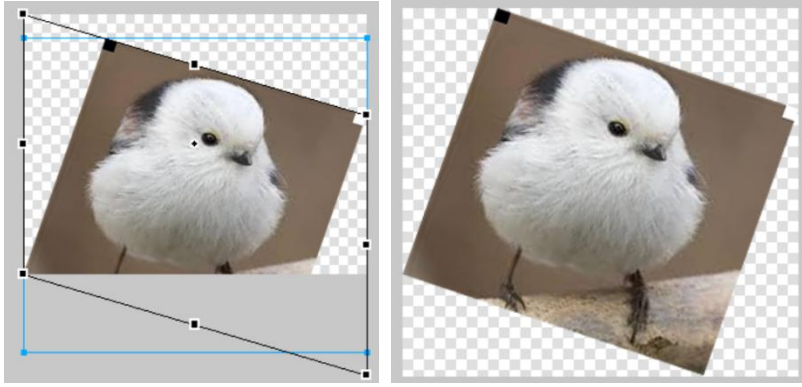
Ödev 10: Mouse ile bir resmi döndürme işlemi yaptırın. Resmin üzerinde mouse sol tuşuna basılı tutarken ve mouse ile döndürme hareketi yaparken resmin 4 kenarını çizgi şeklinde ve mouse takip edecek şekilde Orta nokta etrafında çizdirin. Mousdan elimizi çektiğimiz anda resim gösterilen en son dikdörgeğin içerisinde dönmüş olarak gözüksün. Bu esnada picturbox ın dışına taşan kısımlar gözükmecektir.

Ödev 11: Resim döndürme işlemini açı olmadan Kaydırma yöntemini kullanarak programlayın. (Köşelerden 50 piksel kaydırarak dönder)(Hem saat yönünde, hemde tersi yönde çalışan komutlar geliştirin)

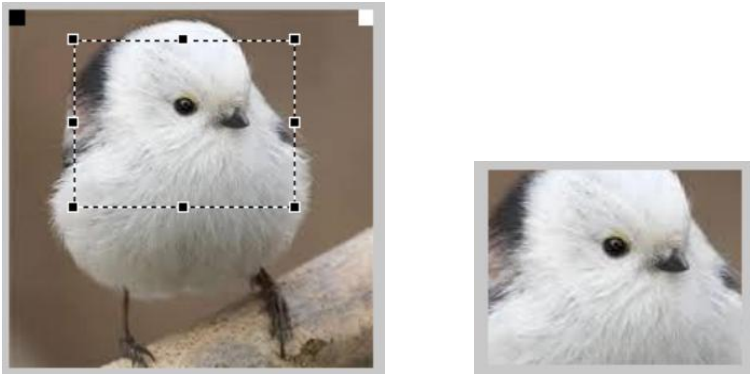
Ödev 12: a) Döndürme komutunu programlarken Geometrik dönüşümle programlayın. Oluşan Alias boşluklarını yok etmek için resim oluştuktan sonra siyah noktaların etrafındaki pikselleri okutun ve renk olarak etrafındaki piksellerin ortalamasını alacak şekilde programlayın. Bu şekilde döndürme nedeniyle oluşan hatayı yok etmeye çalışın. Eğer Aliasları yok etmek için iyi bir algoritma bulursanız büyük harflerle anlatın.

b) Döndürme işlemi Yana ve aşağı kaydırarak programlayın. Notlarda bu konuda hazır formül verilmiştir fakat bunlar resmin ortasına göre kaydırma yapıyor. Siz bunu kullanmayacaksınız. Resmin köşesinden eğerek döndürme yapacak. Bu konu notlar içinde anlatılmıştı. Burada yana ve aşağı kaydırırken oluşan açı yı geliştireceğiniz formüller içinde kullanın. Yani açı verildiği zaman köşeden itibaren eğerek tam o açı kadar dönmeyi sağlayacak yöntemi yada formülü bulun. Resmin dışına çıkan kısımların kaybolmamasına dikkat edin. O problemi de çözün. Geliştirdiğiniz önemli uygulamaları büyük harflerle anlatın.



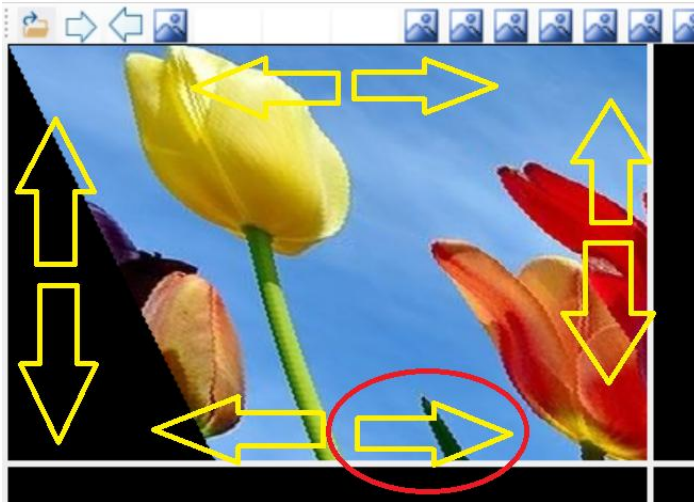


Ödev 13: Kırpma işlemini yapan kodları yazınız. Mouse takip eden çizgi çizdirerek yapın...



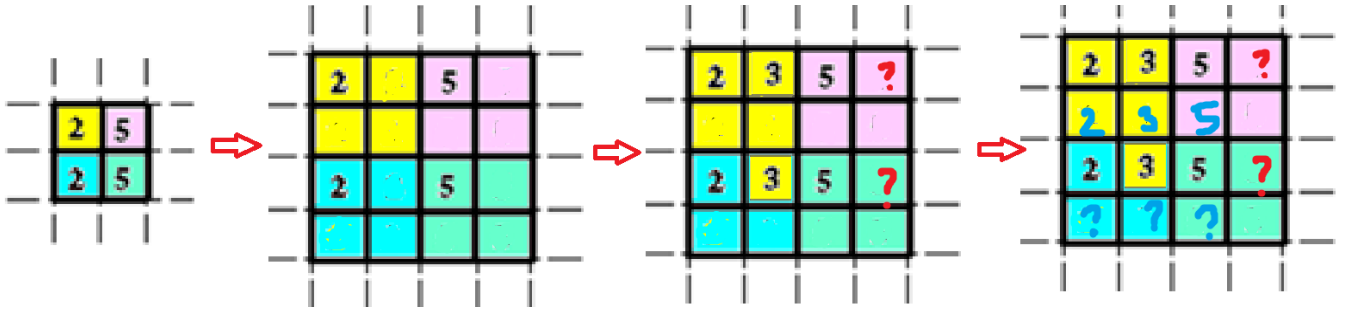
Ödev 14:

Aşağıdaki gibi bir resmin 4 kenarından tutup sağa-sola yada yukarı aşağı mouse ile sürüklemeye yaptığımızda, o kenar üzerinde ilgili kaydırma işlemini yaptırın. Bu uygulamada mouse ile aynı anda kaydırma işlemi oluşsun...



Ödev 15:

Aşağıdaki resimde gösterildiği şekilde bir resim büyültme algoritması geliştirin. Bu işlemde resim iki bütülürken her pikseli 4 lü gözlerin bir köşesine ekleyin. Aralardaki boşlukları sağa doğru tararken iki pikselin ortalamasını alarak, aşağı doğru tararken yine iki pikselin ortalamasını alarak algoritmayı geliştirin.



Ödev 16 (Mouse un bulunduğu nokta etrafında tekerle zoom yapma): Aşağıdaki resimde gösterildiği gibi Mouse resmin herhangi bir noktasında iken Mouse tekerini çevirdiğimizde bulunduğu noktanın etrafında zoom in/out (yaklaştırma /uzaklaştırma) işlemlerini yapan algoritmayı geliştirin.



Ödev 17 (Mouse bulunduğu nokta etrafında, tekerle döndürme): Aşağıdaki resimde gösterildiği gibi Mouse resmin üzerinde herhangi bir noktada tutalım. Döndürme komutunu aktif ettikten sonra bu işlemleri yapmalıyız. Odaklanılan nokta üzerinde iken Mouse tekeri geriye yada ileri dönderildiğinde resimde bulunan noktanın etrafında dönsün. Oluşan Alias (resim piksel boşlukları) olayı da bu esnada bulunabilir.



Ödev 18 (Alias Yok Etme): Resim pikselleri üzerinde hesaplama yaptığımızda double olan küsüratlı değerler resmin üzerine yerleştirilirken integer a çevrilmek zorundadır. Bu nedenle küsüratlar kaybolunca bazı piksellerin üzerine renk atanmamaktadır. Bu durum Alias dediğimiz boşlukların oluşmasına neden olmaktadır. Aşağıda resimde verildiği şekilde bir resmi değişik açılarda döndürdüğümüzde alias oluşan boşlukları kapatan bir algoritma geliştirin. Bunun için fikir olarak şu söylenebilir. Resmi öncelikle diziye atın. Döndürme işlemlerini dizi

üzerinden yapın. Yeni resmin çerçevesini de dizide tutun. Bu ikinci dizide her renk atanan koordinata bir eğer bir atama yapıldıysa bunu bir bilgide tutun. En tarama işlemi bittiğinde bu ikinci dizi bize hangi piksellerde renk atanmadığı bilgisini verecektir. Sonrada ikinci resmi bir daha taradığımızda renk atanmayan piksellerindeki etrafındaki 9 piksele bakıp bunların ortalama rengi, bu atanmayan piksele verilebilir. Bu şekilde algoritmayı geliştirin.

