CME 112- Programming Languages II
Lecture 11: Basic Graphic Operations

Assist.Prof. Dr. Ümit ATİLA

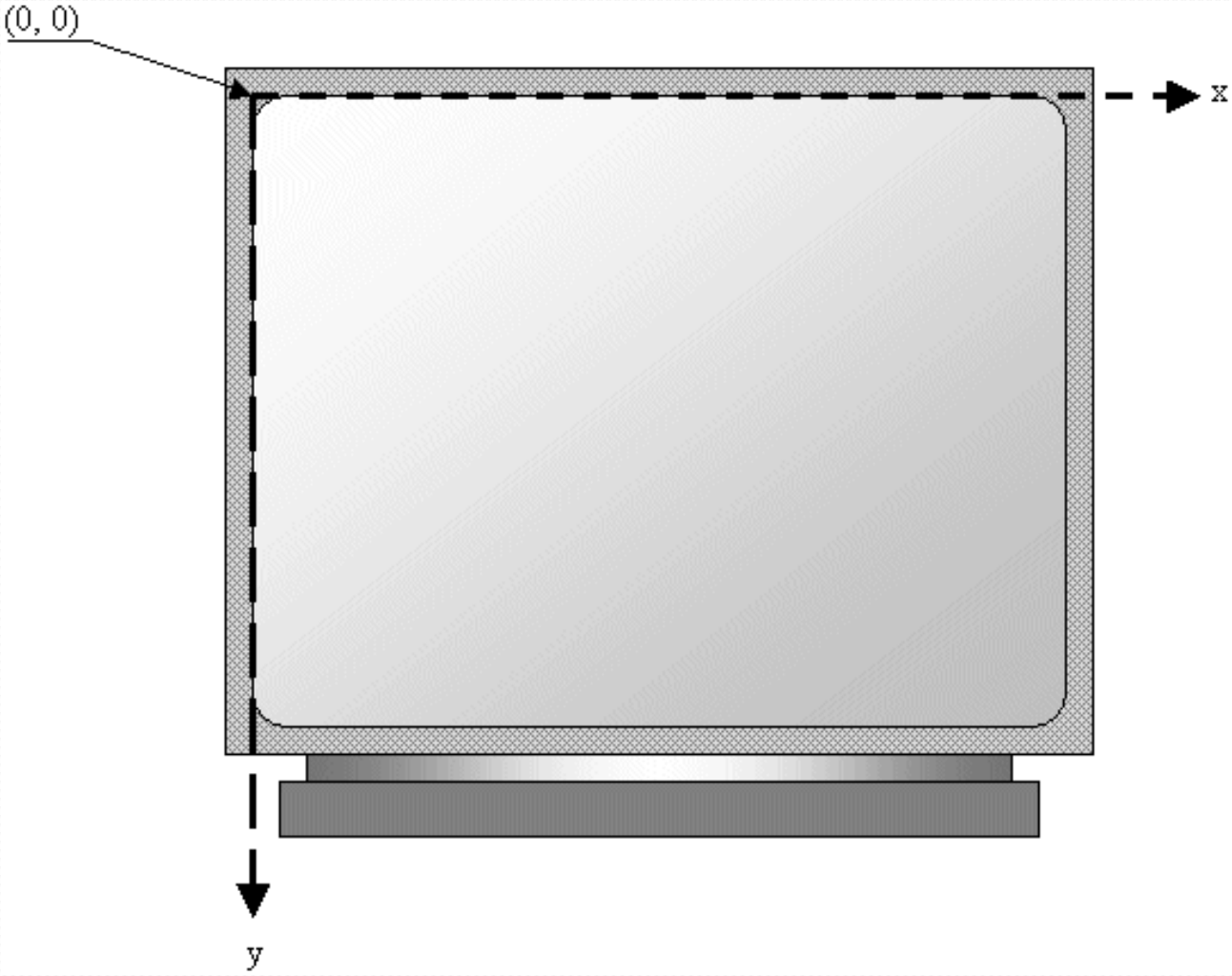
Grafik Programlama

- Bilgisayar kullanılırken monitörlerde iki tür ekran moduyla karşılaşılır. Bu ekran modları **Text modu** ve **Grafik modu'dur.**
  - Text modunda ekran 25 satır ve 80 sütundan oluşur.
  - Grafik modu, görüntülerin noktalarla oluşturulduğu moddur.
    - Monitör ekranı pixellerden (noktalardan) oluşur
    - Nokta sayısı yoğunluğuna çözünürlük denir.

Grafik Programlama

- **Grafik ekranı**, ekranın yatayda ve düşeyde eşit aralıklar bölünmesinden elde edilen matris elemanlarından oluşur.
  - Bu elemanların her birine piksel (pixel) denir.
  - Her pikselin yatay ve düşey olmak üzere bir koordinat numarası var
- Grafik sisteminde ekranın sol üst köşesi, piksel matris alanının başlangıç noktasıdır.
  - X koordinatı soldan sağa doğru, Y koordinatı yukarıdan aşağıya doğru artar.
  - Ekranda konumlandırılmış çizimler sol üst köşeye göre konumlandırılırlar.

Grafik Programlama



Renk Derinliği

- Piksellerin alabileceği renkler kırmızı, yeşil ve maviden türetilir
- **Renk derinliği** bu renklerin miktarını belirler
  - Renk derinliği ne kadar artarsa her pikselin alabileceği renk sayısı artar, renkler gerçeğe daha yakın olur.
- Renk derinliği bit cinsinden belirtilir
  - 8 bit kullanıldığında bu bitlerden $2^8 = 256$ kombinasyon üretilir
- True color, kullanılan üç rengin de (kırmızı, yeşil ve mavi) 256`şar tonu gereklidir
  - Renk başına 8 bitten 24 bit yapar
  - Ekran kartları pikselleri bu modda göstermek için 32 bite ihtiyaç duyarlar. Kalan 8 bit alpha kanalı (piksellerin saydamlık bilgisini tutar) için kullanılır.

Renk Derinliği

- High Colour (16 bit) modunda, yeşil için altı ve maviyle kırmızı için de beşer bit kullanılır.
  - Yeşil için 64, maviyle kırmızı için de renk başına 32 farklı yoğunluk var
  - Renk kalitesinde 32 bite göre çok az fark var
  - Piksel başına 4 yerine 2 byte (8 bit = 1 byte) hafıza gerekeceğinden 32 bite göre performans avantajı sağlar.
- Ekran kartı üretemediği renkler için ne yapar?
  - Sistemimizin 256 renge ayarlı olduğunu fakat 16 bitlik bir resim dosyası açtığımızı varsayalım.
  - Hazırdaki renklerin değişik kombinasyonları kullanılarak üretilemeyen renge yakın bir renk oluşturulur (dithering)

Görüntü Oluşumu

- Görüntü piksel denen noktalardan oluşur.
- Görüntü oluşumu için piksellerden oluşan bir matrise renk değeri atanır.
- Ekran çevre birimidir ve bilgisayar ile olan iletişimini bir bağdaştırıcı ile yapar. Bu bağdaştırıcı grafik kartıdır.
- Bilgisayarın Merkezi İşlem Birimi (MİB), görüntü kartının belleğine görüntüye ait sayısal verileri yerleştirir.
- Görüntü kartı bu sayısal görüntüyü kendisine bağlı ola n ekrana pikseller olarak yansıtır.

Görüntü Oluşumu

- Bilgisayarların grafik özellikleri sürekli gelişmiştir.
  - 1980'lerde CGA ekranlar
    - Yatayda en fazla 320, dikeyde en fazla 200 piksel
    - Her piksel en fazla 4 renkten birini alabilir
    - Ekran duyarlılığı (320*200*4)
  - VGA ekran duyarlılığı 640*480*16
  - SVGA ekran duyarlılığı 800*600*256
  - Günümüzde yatayda 1280, dikeyde 1024 piksel ve her piksel 16 milyondan fazla renkten biri olabilir.
  - HD ve Full HD (1920*1080)

RGB

- Renk sayısının artmasıyla renk belirlemek için değişik yöntemler uygulandı
  - RGB (Red Green Blue)
  - Bir renk kırımızı yeşil ve mavinin değişik tonlarının bir araya gelmesiyle tanımlanır
  - Kırmızı mavi ve yeşilin kaç tonu olduğu kullanılan görüntü kartının özelliğine bağlıdır.
    - Her temel renk için 256 farklı ton varsa 256*256*256 = 16.777.216 farklı renk elde edilir.

Resim İşleme Yazılımları

- Görüntü oluşturmak için temel nesneler
  - Çizgi, dikdörtgen, elips, eğri, yazı, boya vb. araçlar
  - Görüntüye eklenen her nesnenin görüntü içindeki göreli konumu ve biçim bilgisi dosya içine yazılarak dosyalama gerçekleşir
  - CorelDraw, PhotoShop vs.
  - Nesne tabanlı bu uygulamalar görüntü üstünde nesne ekleme, seçme, kesme, kopyalama ve yapıştırma yapabilir.
  - Görüntüyü oluşturan nesneler kendi aralarında gruplanabilir.
  - Bir görüntüye bir nesne eklendiğinde altta kalan nesneleri örtmesi veya şeffaf durması sağlanabilir.
  - Resim işleme yazılımları katmanlar halinde tutar.

Form Üzerinde Grafik Çizimi

- Öncelikle bir grafik bileşeni ve bir kaleme ihtiyaç duyulur.
- Grafik bileşeni tanımlamak için;

```
Graphics ^ g = this->CreateGraphics();
```

- Grafik bileşeninin ekrana çizim yapabilmesi için kaleme ihtiyaç duyulur.
  - Kalem çizgi kalınlığı, rengi, deseni vb özellikleri olan bir araç
  - Dinamik kalem renk değişkenine atanan renk ile oluşturulur

```
System::Drawing::Pen ^ kalem = gcnew System::Drawing::Pen(renk);
```

  - Kalemin nokta kalınlığı için;
    - Kalem->Width = 1;

Form Üzerinde Grafik Çizimi

- Kalem bileşeni oluştuktan sonra g bileşeni ile çizimler yapılır

Düz Çizgi :

```
g->DrawLine(kalem,x1,y1,x2,y2);
```

Dörtgen :

```
g->DrawRectangle(kalem,x1,y1,en,boy);
```

Çember :

```
g->DrawArc(kalem, x1, y1,en,boy,başlangıç, bitiş);
```

Elips :

```
g->DrawEllipse(kalem,x1,y1, en, boy);
```

Pasta Dilimi :

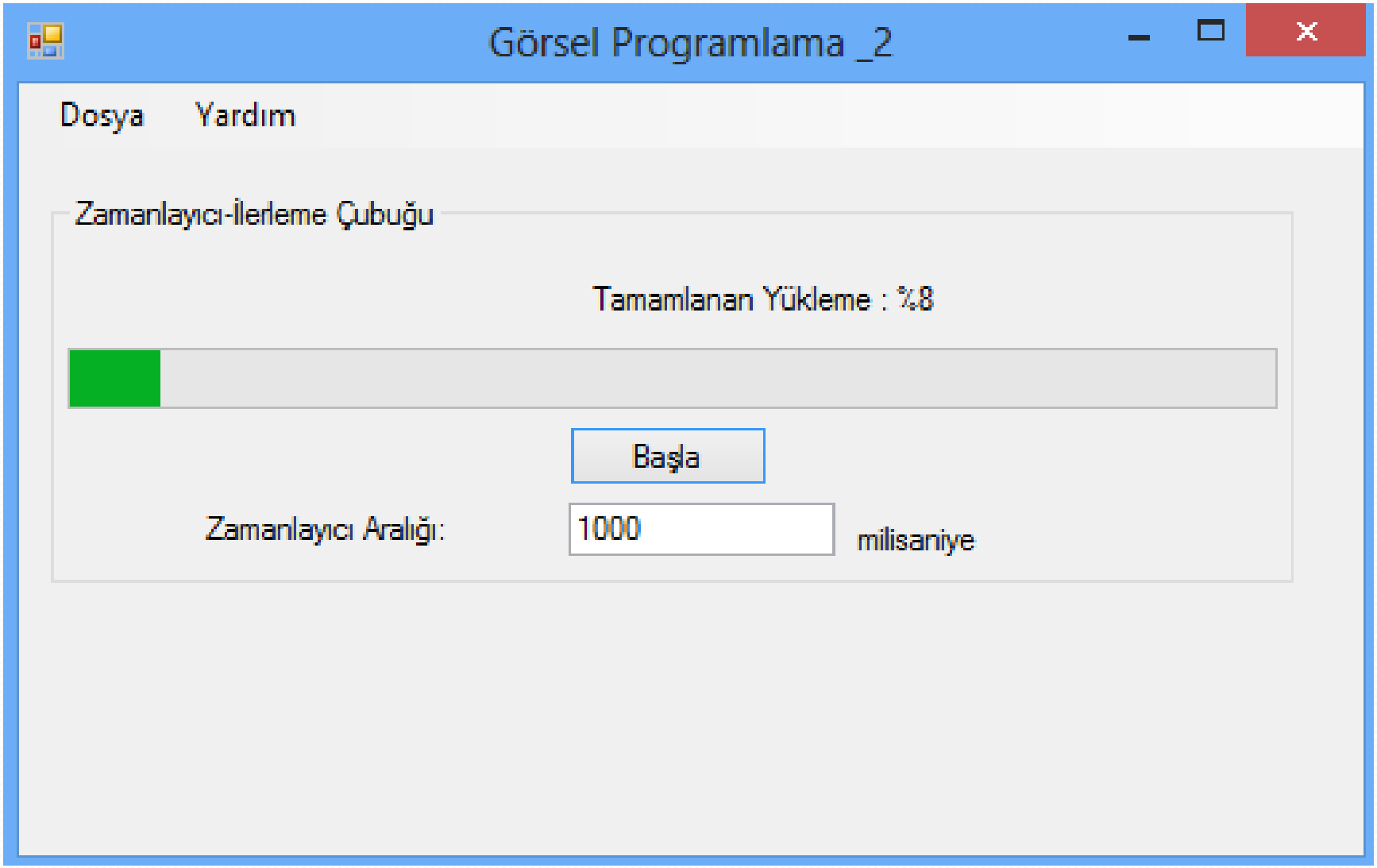
```
g->DrawPie(kalem, x1,y1,en,boy,başlangıç, bitiş);
```

*** başlangıç ve bitiş derece cinsindendir.**

Form Üzerinde Grafik Çizimi

- Çizim işlemleri bittikten sonra kalem ve grafik bileşeni için bellekten alınan yerler iade edilir. Bunun için delete fonksiyonu kullanılır.
  - delete g;
  - delete kalem;
- Form ekranının temizlenmesi için
  - Form1::Refresh();

Uygulama-1 (Form1)



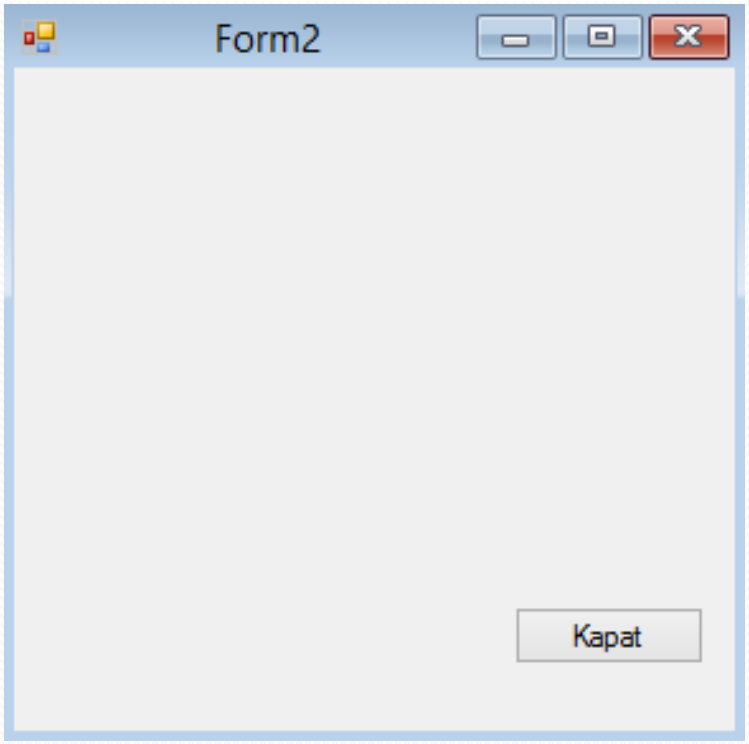
Uygulama-1 (Form1)

```
private: System::Void btnBasla_Click(System::Object^  sender, System::EventArgs^  e) {
             timer1->Enabled= true;
             progressBar1->Value = 0;
         }
private: System::Void txtZamanAraligi_TextChanged(System::Object^  sender, System::EventArgs^  e) {
                 if(txtZamanAraligi->Text != "")
                    timer1->Interval = Convert::ToInt32(txtZamanAraligi->Text);
         }
private: System::Void timer1_Tick(System::Object^  sender, System::EventArgs^  e) {
             if(progressBar1->Value == 100)
                 timer1->Enabled = false;
             else
                progressBar1->Value++;
             lblYukleniyor->Text = "Tamamlanan Yükleme : %"+ progressBar1->Value.ToString();
         }
```

Uygulama-1 (Form1)

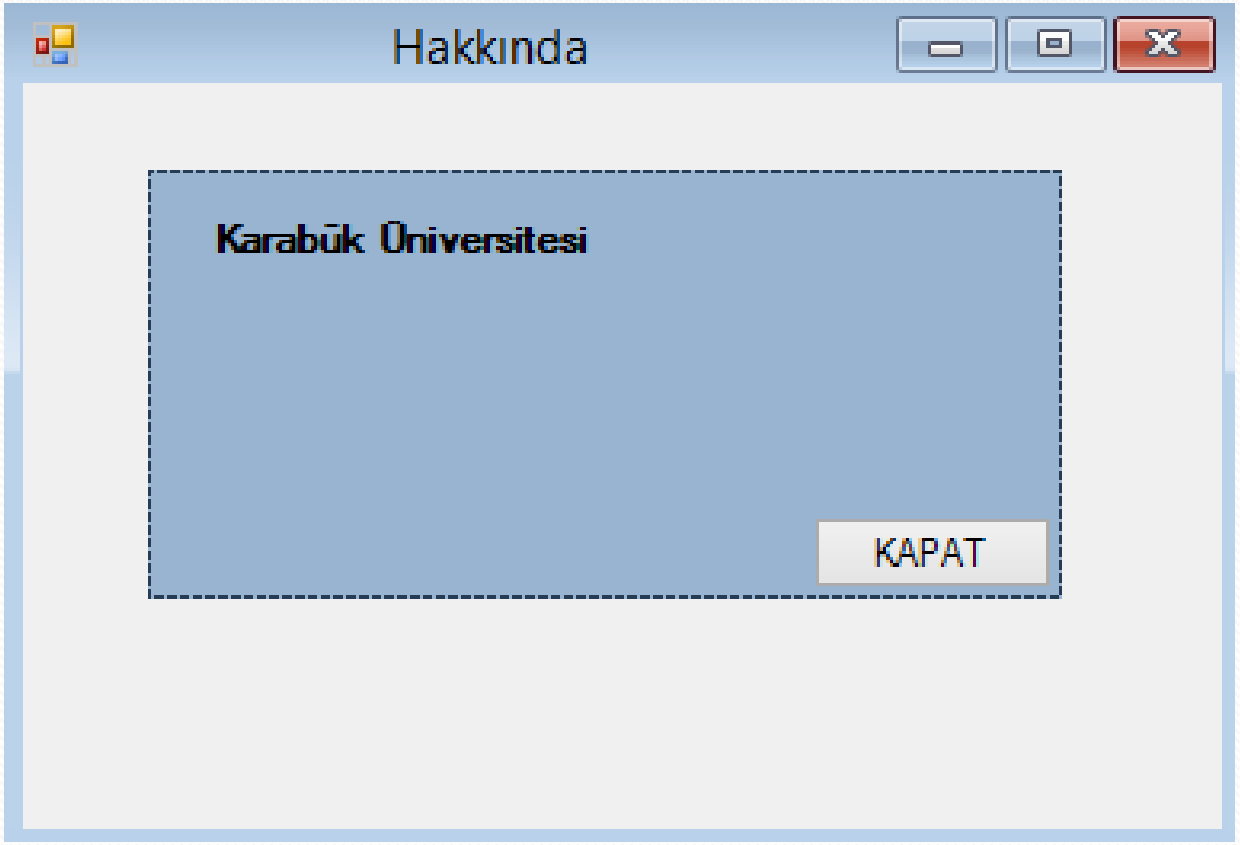
```
private: System::Void yeniToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
             Form2^ yeniForm2 = gcnew Form2();
             yeniForm2->Show();
         }
private: System::Void closeToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
             this->Close();
         }
private: System::Void hakkındaToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
             Form3^ yeniForm3 = gcnew Form3();
             yeniForm3->Show();
         }
private: System::Void graphicsToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) {
             Form4^ yeniForm4 = gcnew Form4();
             yeniForm4->Show();
         }
```

Uygulama1(Form2)



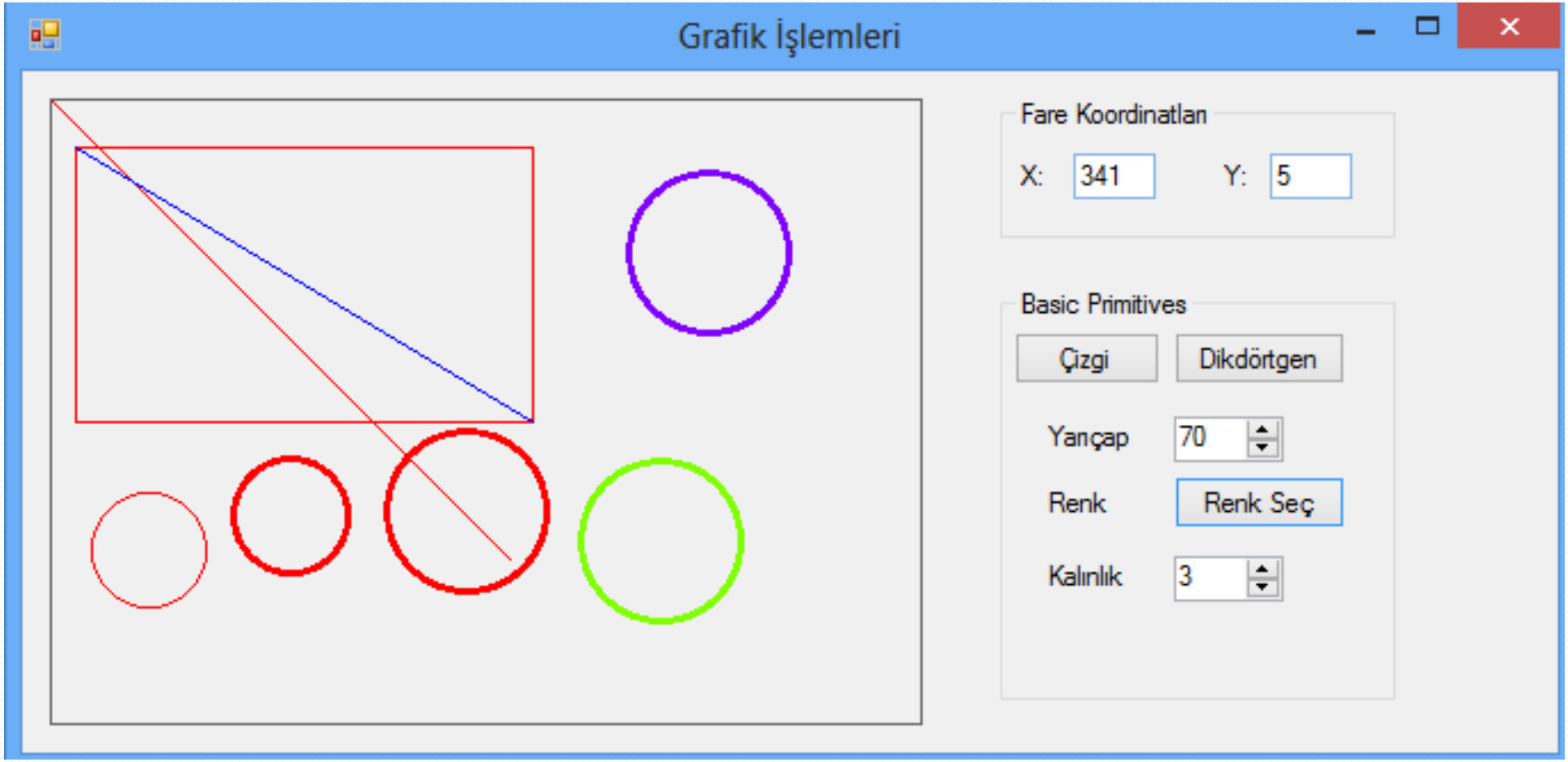

```
private: System::Void btnKapat_Click(System::Object^  sender, System::EventArgs^  e) {
             this->Close();
         }
};
```

Uygulama1(Form3)




```
private: System::Void btnKapat_Click(System::Object^  sender, System::EventArgs^  e) {
             this->Close();
         }
};
```

Uygulama1(Form4)



Uygulama1(Form4)

```
    private: System::Drawing::Graphics^ globalGraphics;
    private: System::Drawing::Pen^ globalPen;

private: System::Void pictureBox1_MouseMove(System::Object^  sender, System::Windows::Forms::MouseEventArgs^  e) {
             txtX->Text=e->Location.X.ToString();
             txtY->Text=e->Location.Y.ToString();
         }
private: System::Void Form4_Load(System::Object^  sender, System::EventArgs^  e) {
             globalGraphics=pictureBox1->CreateGraphics();
             globalPen = gcnew System::Drawing::Pen(System::Drawing::Color::Blue);
         }
```

```
private: System::Void btnCizgi_Click(System::Object^  sender, System::EventArgs^  e) {
             System::Drawing::Pen^ myPen =gcnew System::Drawing::Pen(System::Drawing::Color::Red);
             System::Drawing::Graphics^ pbGraphics;
             pbGraphics = pictureBox1->CreateGraphics();

             pbGraphics->DrawLine(myPen, 0, 0, 200, 200);

             delete myPen;
             delete pbGraphics;
         }
```

Uygulama1(Form4)

```
private: System::Void btnDikdortgen_Click(System::Object^  sender, System::EventArgs^  e) {
            System::Drawing::Pen^ myPen =gcnew System::Drawing::Pen(System::Drawing::Color::Red);
            System::Drawing::Graphics^ pbGraphics;
            pbGraphics = pictureBox1->CreateGraphics();

            pbGraphics->DrawRectangle(myPen,10,20,200,120);
            myPen->Color=System::Drawing::Color::Blue;
            pbGraphics->DrawLine(myPen,10,20,210,140);

            delete myPen;
            delete pbGraphics;
         }
```

Uygulama1(Form4)

```
private: System::Void pictureBox1_MouseClick(System::Object^  sender, System::Windows::Forms::MouseEventArgs^  e) {
             int radius=System::Convert::ToInt32(numericUDAci->Value);
             globalPen->Width=System::Convert::ToDouble(numericUDKalinlik->Value);
             globalGraphics->DrawEllipse(globalPen,e->Location.X-radius/2,e->Location.Y-radius/2,radius,radius);
         }
private: System::Void btnRenkSec_Click(System::Object^  sender, System::EventArgs^  e) {
              if ( colorDialog1->ShowDialog() == ::System::Windows::Forms::DialogResult::OK )
                 globalPen->Color = colorDialog1->Color;
         }
};
```