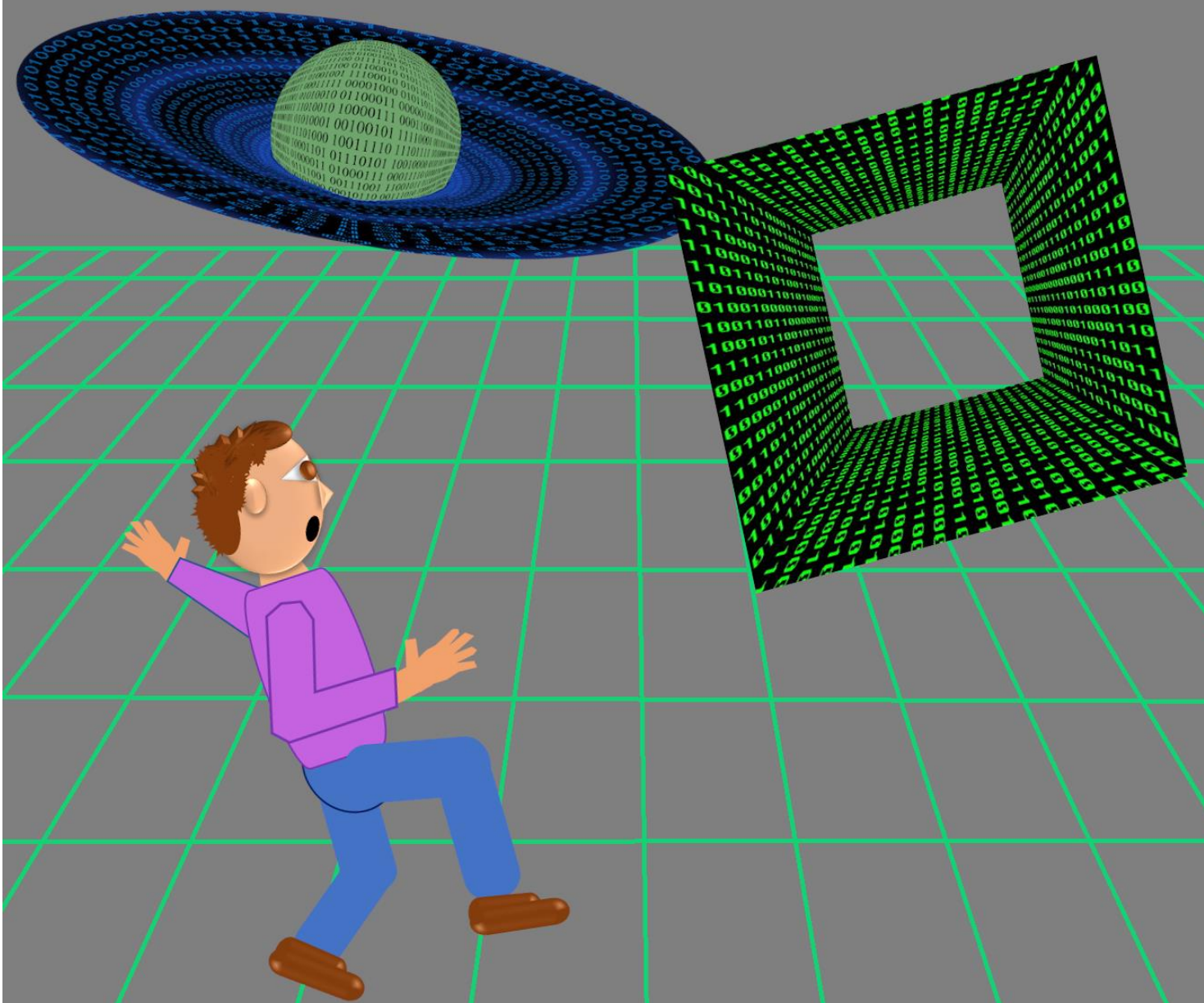
I SEE BYTES:
BASİTLEŞTİRİLMİŞ C++ KÜTÜPHANESİ



İbrahim Cem Baykal

1. BASKI

ICBYTES: BASİTLEŞTİRİLMİŞ C++ KÜTÜPHANESİ

İBRAHİM CEM BAYKAL

1. BASKI
Nisan 2023

- Bu kitabın bütün hakları İbrahim Cem Baykal'a aittir.
- T.C. Kültür ve Turizm Bakanlığı Telif Hakları Genel Müdürlüğü Kayıt-Tescil No: 2023/8358.
- Elektronik kopyalarını üzerinde hiçbir değişiklik yapmadan dağıtmak serbesttir.
- Basılı kopyası kişisel kullanım için tek adet üretilebilir ancak satılamaz.
- Kaynak göstermeden alıntı yapılamaz.

İÇİNDEKİLER

Önsöz

Bilgisayar programcılığı ile ileri düzeyde uğraşan herkes bir noktada yazılım teknolojilerinin uyumsuzluğunu fark etmişlerdir. Buna en basit ancak en çarpıcı örnek yalın karakter dosyasıdır. Windows işletim sisteminde "Not Defteri" (Notepad) isimli programı tarafından yaratılan dosyalar (.txt dosyaları) Linux ve MacOS işletim sistemlerine uymamaktadır. Bu dosyaları yazarken "Enter" tuşuna her bastığınızda satır sonuna eklenen görünmez karakterler olan "aşağı satıra geç" (Line Feed-LF) ve "satır başına dön" (Carriage Return-CR) komutları bu üç işletim sisteminde farklı şekilde yer almaktadır. Her ne kadar büyük bir sorun teşkil etmese de, uzun bir süredir gezegenimizde baskın olarak kullanılan bu üç işletim sisteminin bu kadar basit bir konuda dahi ortak bir format geliştirememiş olmaları bu problemin ciddiyetini göstermektedir.

Bir elektronik mühendisi olarak yazılım teknolojilerindeki bu uyumsuzluğu özellikle fark etmekteyim çünkü donanım teknolojileri bunun tam tersidir. Halk arasında toplama bilgisayar diye tarif edilen; masaüstü bilgisayar parçalarını farklı üreticilerden satın alıp, bir araya getirilmesiyle inşa edilen bilgisayarlar bunun en çarpıcı örneğidir. Yine çok başarılı bir iletişim arabirimi olan USB buna diğer bir örnektir. Televizyonlardan farelere kadar çok fazla sayıda cihazı birbirine sorunsuz olarak bağlayabilmenizi sağlar. Bir diğer standart da Ethernet teknolojisidir. Sayısız türde cihazın kablolu ve kablosuz olarak iletişimini sağlar. Üstelik bu iki teknolojiyi birbirine dönüştüren cihazlar bile bulunmaktadır.

Diğer taraftan Windows'a bulaşan bir bilgisayar virüsü aynı işlemci üzerinde çalışıyor olsa bile MacOs veya Linux bilgisayara bulaşamamaktadır. Bu durum her ne kadar yararlı olsa da aslında bu işletim sistemlerinin ne kadar uyumsuz olduğunu göstermektedir.

Daha derinlere indiğimizde bir programcının karşısına çıkan en bilindik problem başkasının yazdığı bir programı kendi yazdığı bir program ile bütünleştirmeye çalışırken ortaya çıkmaktadır. Kendi programı veriyi link-list olarak işlerken, diğer programcı ikili-ağaç (binary tree) veri yapısı kullanıyorsa bu iki programı bütünleştirmek için bazen o kadar uğraşması gerekir ki başkasının kaynak kodunu veya kütüphanesini kullanmak yerine aynı işi yapan kendi kodunu baştan yazmak zorunda kalır. Bunun kanıtı, en eski ve en uzun süredir popüler olan C/C++ dilinde aynı işi yapan muazzam sayıda kod ve kütüphane bulunmasıdır. İnanmıyorsanız bir konu seçin ve internette kodunu arayın. Mesela matris

hesaplamaları yapan bir kütüphane arayın. Aynı işi yapan üstelik te bedava olan bu kadar sayıda kütüphane olması sebebi nedir?

İşte ICBYTES (I see Bytes) kütüphanesini geliştirilme sebebi budur. İsmini bu kütüphanenin merkezinde bulunan veri yapısından alan bu kütüphane, fonksiyonlar arasında kullanılan argüman olarak tek tip bir değişken kullanmaktadır. **Böylelikle sınırsız bir bütünleşme sağlamaktadır.** ICBYTES değişken tipi içerisinde temel veri yapılarının hemen hepsini taşıyabilecek ve bunu dinamik olarak yapabilecek şekilde tasarlanmıştır. Dört boyuta kadar (x,y,z,w) matrisler, vektör, liste, ikili ağaç gibi birçok veri yapısını taşıyabilmektedir.

Doktora öğrencisi iken kullanmaya başladığım OpenCV kütüphanesindeki sinir bozucu sayıda farklı nesne ve yapı (structure) olması beni bu konu üzerinde düşünmeye itti ve yıllar içinde çok farklı sayıda değişken tipi ve veri yapısı tanımını sıkıştırılmış bir şekilde tutmanın yolunu tasarlamaya başladım. İşte I-See-Bytes kütüphanesi bu çabanın sonucu olarak ortaya çıktı.

Benzer bir çabaya sahip olan bazı kişiler de Python dilini çıkarttılar ki bu dilde de değişkenler çok farklı türde nesneleri tutabilmektedir. Matlab dili de Python kadar olmasa da farklı nesneleri tek bir değişkende tutabilmektedir. Son zamanlarda ABD'deki üniversiteler, mühendislik bölümlerinin programlamaya giriş derslerinde en çok bu iki dili öğretmeye başladılar. Ancak hem Matlab, hem de Python gerçek zamanlı sinyal, görüntü ve video işleme uygulamaları için uygun değiller çünkü bu diller bir yorumlayıcı üzerinde çalışmaktadır. Gerçek zamanlı (Real-Time) resim ve video işleme algoritmaları yüksek hızda çalışmak zorundadırlar ancak, yorumlayıcılar derleyicilere göre çok daha yavaş olduğu için bu diller resim işleme alanında ancak araştırma ya da eğitim amacıyla kullanılabilmektedir. Video işleme alanında endüstriyel yazılımların bu dillerle yazılması pek mümkün değildir.

Java dili de sahip olduğu geniş kütüphaneler yüzünden son derece popüler olmuştur ancak virtual machine üzerinde çalıştığı için C++ diline göre çok daha yavaştır. Bu nedenle, ilerleyen bölümlerde göreceğiniz üzere, "I See Bytes" kütüphanesi C++ dilinde yazılmış olmasına rağmen, farklı platformlarda sorunsuz çalışabilmesi için özel önlemler alınmıştır.

İbrahim Cem Baykal

Mart 2023

1. Tasarım ve Terminoloji

Önsöz'de belirtildiği üzere, I-See-Bytes kütüphanesinin yazılmasının en önemli sebeplerinden biri OpenCV kütüphanesine alternatif, öğrenmesi ve kullanması çok daha kolay bir kütüphane yaratmaktı. Matlab'ı kütüphane değil de bir dil olarak kabul edersek, OpenCV kütüphanesi, bu kitabın yazıldığı 2023 yılında, özellikle akademik çevrelerde dünyanın en fazla kabul görmüş Bilgisayarla Görü (BG) kütüphanesidir. Ancak bu kütüphaneyi öğrenmek ve kullanmak kolay değildir. Hem Matlab'ın hem de OpenCV'nin temel veri yapısı matristir. Bunun nedeni sayısal (dijital) resimleri hafızada tutmak ve manipüle etmek için en uygun veri yapısının matris olmasıdır. Ancak OpenCV'de kısaca "Mat" diye adlandırılan bu veri yapısı maalesef çok ilkeldir.

1999 yılında geliştirilmeye başlanan bu kütüphane, yüksek sayıda algoritma içerdiği ve bedava olduğu için özellikle akademide yaygın olarak kullanılmaktadır. Ancak 20 sene evvel geliştirilmeye başlandığı için çekirdeğinde hala 20 sene evvelinin tasarım anlayışını ve C++ yapılarını taşımaktadır. Bu nedenle kod yazarken kullanışsızdır. Oysa son 20 senede hem C++ diline çok sayıda ekleme yapılmış hem de tasarım teknikleri gelişmiştir. Diğer bir dezavantajı ise Windows işletim sistemi düşünülerek tasarlanmadığı için profesyonel görünümlü uygulamalar yaratmak zordur. Üstüne üstlük, üzerine profesyonel görünümlü grafik arabirimlerle inşa etmeye çalışıldığında hızı yavaşlamaktadır.

OpenCV'nin en ilkel kısmı, matris cinsinin koşma zamanında (run time) değil, derleme zamanında (compile time) belirlenmesidir. Buna bir de veri yapısının değişken olmamasını eklediğimizde, OpenCV'nin karşısına çıkan farklı veri yapıları ve tipleri ile başa çıkabilmek için çok sayıda veri tanımı ve makro tanımlamalarına başvurmaktan başka çaresi kalmamaktadır. Tüm bu tanımlamalar kullanıcıyı boğmakta, kod yazarken sürekli açıklama kitaplarına başvurma zorunda bırakmaktadır. Bütün bu zorlukların üzerine bir de

OpenCV'nin dökümantasyon eksiklikleri eklendiğinde, bu konuyu öğrenmeye yeni başlayan öğrenciler için bir kabus haline dönüşmektedir.

1.1. OpenCV Kod Örnekleri

OpenCV'nin temel veri yapısı olan matris yaratma yöntemlerini sıralarsak, kullanıcının ne kadar değişik tanımı aklında tutması gerektiği ortaya çıkmaktadır:

```
Mat M=Mat(3,3, CV_64F); //3x3 double matris yaratıyor

M.at<double>(1,3)=5.5;   //yukarıdaki matrisin ikinci satırın 4. elemanını 5.5 yapıyor
```

Yukarıdaki örnekte, 3x3 double matris yaratmak için kullanıcının "CV_64F" adlı makro tanımını bilmesi gerekiyor. C++'da 32 bitlik kayan nokta sayılara "float," 64 bitlik kayan sayılara "double" dendiğini biliyorsanız bu isimlendirmede bir mantık olduğunu düşünebilirsiniz. Ancak 4. sınıf bilgisayar mühendisliği öğrencilerinin bile çok azının float ve double'ın kaç bit olduğunu bilmediğini, bilseniz, elektronik, endüstri ve inşaat gibi bölümlerin öğrencilerinin hiç şansı olmadığını anlardınız. Ayrıca C++'daki "template" yapılarını bilmeyen öğrenciler için "<double>" ifadesinin ne olduğunu anlamaya çalışmak da ayrıca kafa karışıklığı yaratıyor.

Şimdi daha da karışık bir örnek:

```
Mat M(400,300, CV_8UC3); //300x400 24 bitlik resim yaratıyor
```

Bu örnekte 32 bit renk çözünürlüğü olan bir resim yaratılıyor. Bilgisayarda renkler Kırmızı, Yeşil ve Mavinin (KYM) tonlarının karışımı olarak üretilir. Dolayısı ile her renkli piksel bu 3 birleşene, her renkli resim de bu üç kanala sahip olmalıdır. Burada da bu 3 kanalı yaratmak için "CV_8UC3" kelimesi kullanılıyor. Baştaki "CV_" OpenCV'ye ait demek "8UC3" ise 8 bitlik (U→"Unsigned") yani pozitif, "C3"is 3 kanal manasına geliyor. Burada dikkat edilmesi gereken bir başka husus resim boyutlarının tanım sırası. Resim boyutları veya çözünürlüğünden bahsederken önce hep yatay yani X → yönündeki çözünürlüğü sonra da Y yönündeki çözünürlüğü yazarız. Resmin boyu 400x300 dersek yatay olarak 400 pikselden oluşuyor demektir. Ancak OpenCV'de önce hep dikey çözünürlük yazılır. Bu da yeni kullanıcıları hep ters köşeye yatırır. Gelelim matris elemanlarına ulaşım yöntemine. Bu matrisin x:100 y:150 noktasındaki pikselin yeşil tonunu 255 yapmak istersek şöyle karışık bir ifade kullanmamız gerekir:

```
M.at<cv::Vec3b>(150, 100)[1]=255;
```

Yeni başlayanlar çoğunlukla gri tonlamalı (8 bit), renkli 24 bit ve 32 bitlik resimlerle uğraşırlar. Ancak ileri seviye resim işleme ile uğraşanlar 1 bitten 64 bite kadar herhangi bir renk çözünürlüğü ile uğraşabilirler. Bunlara kayan noktalı matrisleri de eklediğimizde ezberlenmesi gereken bazı makro tanımları listesi aşağıdaki gibidir:

CV_8UC1	CV_16UC1	CV_32SC1	CV_64FC1
CV_8UC2	CV_16UC2	CV_32SC2	CV_64FC2
CV_8UC3	CV_16UC3	CV_32SC3	CV_64FC3
CV_8UC4	CV_16UC4	CV_32SC4	CV_64FC4
CV_8UC(n)	CV_16UC(n)	CV_32SC(n)	CV_64FC(n)
CV_8SC1	CV_16SC1	CV_32FC1	CV_16FC1
CV_8SC2	CV_16SC2	CV_32FC2	CV_16FC2
CV_8SC3	CV_16SC3	CV_32FC3	CV_16FC3
CV_8SC4	CV_16SC4	CV_32FC4	CV_16FC4
CV_8SC(n)	CV_16SC(n)	CV_32FC(n)	CV_16FC(n)

Oysa ICBYTES kütüphanesinde resim yaratmak için tek yapılması gereken:

```
ICBYTES M;
CreateMatrix(M,400,300, ICB_UCHAR); //400x300 gri tonlamalı
```

Veya,

```
CreateMatrix(M,400,300,3, ICB_UCHAR); //3x8=24 bit renkli
```

Veya,

```
CreateMatrix(M,400,300,4, ICB_UCHAR); //4x8=32 bit renkli
```

Gri tonlamalı resimde x:100 y:150 noktasındaki pikselin rengini beyaz yapmak için:

```
M.C(100,150)=255;
```

Hem 24 hem de 32 bit resimde ise piksel yeşil tonunu 200 yapmak istesek:

```
M.C(100,150,2)=200;
```

Yeterli olacaktır. Ayrıca ICBYTES kütüphanesi, matrisin tipini bilmeksizin ona erişmenizi veya değiştirmenizi sağlayan mekanizmalar da sunmaktadır. Sadece parantez kullanırsanız matrisin tipi ne olursa olsun o elemanı double cinsinden okuyabilirsiniz:

```
double d= M(100,150)=255;
```

Eğer amacınız yazmak ise:

```
Set(100,150,M,5.5);
```

M matrisinin tipi ne olursa olsun bu komut (100,150) elemanına 5.5 yazmaya çalışır. Tam sayı tipi ise 5 yazar. Float veya double ise 5.5.

```
Set(100,150,M,5);
```

Son argümanı tam sayı verirseniz M matrisi float veya double bile olsa içerisine yazar. Yani, ICBYTES kütüphanesi kullanıyorsanız, matris tipini bilmek zorunda değilsiniz.

1.2. OpenCV Dökümantasyonu

Maalesef OpenCV'nin en felaket yönlerinden biri kullanıcıların öğrenmek için başvurduğu internet dökümanlarıdır. Örneğin Çok kullanılan Shi-Tomasi metodunu uygulayan fonksiyon tanımı internet sayfasında aşağıdaki gibi verilmiştir:

```
void cv::goodFeaturesToTrack(InputArray image,
      OutputArray     corners,
      int   maxCorners,
      double     qualityLevel,
      double     minDistance,
      InputArray      mask = noArray(),
      int   blockSize = 3,
      bool       useHarrisDetector = false,
      double     k = 0.04
 )
```

Burada açıkça görüldüğü şekilde, cinsi "InputArray" diye tarif edilmiş bir resim fonksiyona girdi olarak sunulmaktadır. InputArray yazan yer bir link olduğu için üzerine tıkladığınızda web sitesi sizi hiçbir mana veremediğiniz bir açıklamaya götürür. Ancak biz işi biraz kaptıysak bunun gri tonlamalı, yani 8 bit renk çözünürlüğüne sahip "Mat" cinsi bir matris olduğunu tahmin edebiliriz. Bu durumda doğal olarak "OutputArray" yazan "corners" isimli çıktının da yine "Mat" cinsi bir matris olmasını beklersiniz değil mi? Çok yanılırsınız. Oraya OpenCV içerisinde tanımlanmış olan "Point2f" cinsinden bir yapının STL (Standard Template Library) vektörünü yollamanız gerekmektedir. Yani "image" ve "corners" argümanlarını aşağıdaki gibi yaratıp ondan sonra bu fonksiyona argüman olarak yollamalısınız:

```
Mat image(Y,X, CV_8UC1);
vector<Point2f> corners;
```

Maalesef bu bilgiyi sadece, eğer varsa, örnek program kodlamalarından öğrenebilirsiniz. Yoksa işiniz zor. Oysa I-See-Bytes kütüphanesinde sadece tek çeşit veri yapısı kullanılmaktadır o da ICBYTES. İçindeki baytları dinamik olarak yorumlayıp farklı veri yapılarına büründürebildiği için ismi de oradan gelmektedir: **Sadece baytlar görüyorum**. Aşağıda OpenCV'de sıklıkla kullanılan veri yapılarının bazıları listelenmiştir:

Point2i	Point3i	Point2d	Size2f
Point2l	Point3f	Size2i	Size2d
Point2f	Point3d	Size2l	DataType
Rect2i	Rect2d	Keypoint	Complex
Rect2f	RotatedRect	Range	Scalar

Bir de **enum** sabitleri var. Bunların aslında makro değişkenlerden kullanım olarak bir farkı yok. Ancak onlar daha çok özel metotlar ve fonksiyonlar içinde kullanmak amacıyla tanımlanmış oluyorlar ve sayıları muazzam miktarda. Aşağıdaki tabloda sadece piksel formatı dönüşümlerinde kullanılan 200 civarındaki **enum** sabitlerinden 18 tanesi gösterilmiştir.

COLOR_BGR2BGRA	COLOR_BGR2XYZ	COLOR_YUV2RGB_NV12
COLOR_RGB2RGBA	COLOR_BGR2YCrCb	COLOR_RGB2HLS_FULL
COLOR_BGRA2RGBA	COLOR_BGR2HSV	COLOR_YUV2RGBA_NV12
COLOR_BGRA2RGBA	COLOR_RGB2Lab	COLOR_YUV2BGR_I420
COLOR_GRAY2RGBA	COLOR_RGB2Luv	COLOR_BayerRGGB2RGBA
COLOR_BGR5552RGBA	COLOR_YUV2BGR	COLOR_BayerRGGB2RGB_EA

Yukarıdaki **enum** sabitleri özellikle matrislerin renk çözünürlüğü veya renk uzayı dönüşümleri için kullanılmaktadır. Aşağıda bu sabitleri kullanarak renk çözünürlüğü dönüşümünün OpenCV'de nasıl yapıldığı gösterilmiştir:

```
cvtColor(girdi,cıktı,COLOR_RGB2ARGB); //24 bit çözünürlükten 32 bit
                                      çözünürlüğe

cvtColor(girdi,cıktı,COLOR_ARGB2GRAY);//32 bit çözünürlükten 8 bit
                                      çözünürlüğe
```

Yani sadece çıktı türünü değil, girdi matrisinin renk yapısını da sizin bilip parametreyi ona göre seçmeniz gerekmektedir. Diyelim ki girdi matrisi size başka bir OpenCV fonksiyonu tarafından yollandı. Kötü dokümantasyon yüzünden de bu fonksiyonun ne renk türünde matris yolladığını da öğrenemediniz. İşiniz zor. Girdi matrisinin türünü yanlış girdiyseniz karşılaşacağınız sonuçlar felaket olabilir. Girdi matrisinin türünü neden bilmek zorunda olalım ki? Bu "cvtColor()" isimli

fonksiyon neden girdi matrisinin türünü kendi algılayamıyor? Hani bilgisayarlar zeki idi?

Aynısı I-See-Bytes kütüphanesinde ise aşağıdaki gibi kolayca hallediliyor:

```
ToRGB32(girdi,cıktı); //Herhangi bit çözünürlükten 32 bit çözünürlüğe
Luma(girdi,cıktı);    //Herhangi bit çözünürlükten 8 bit gri tonlamalı
                      çözünürlüğe yani ekranda direk gri tonlama ile
                      gösterebilirsiniz.
```

Burada OpenCV'nin jurasik dönemden kalma hantal ve karmaşık yapısıyla ilgili sadece birkaç örnek vermekle yetindik. Maalesef OpenCV'nin aslında büyük bir kısmı "katkı" (contrib) (https://github.com/opencv/opencv_contrib) adı verilen başka insanlar tarafından yazılmış daha minik kütüphanelerden oluşur. Bunlar tamamen kafalarına göre veri yapıları kullanır ve üstelik bunun dokümantasyonunu çok az ya da hiç sağlamazlar. OpenCV'de uzmanlaşmış kişilerden bu konunun zorluklarına dair örneklerini öğrenebilirsiniz.

1.3. Terminoloji

Şu ana kadar bahsettiğimiz konularda çoğu zaman bilişim alanında bir terminoloji kullandığımızda yanında İngilizce karşılıklarını yazmak zorunda kaldık. Bilişim terimleri maalesef Türkiye'de sürekli bir tartışma konusudur. Bunun en büyük sebebi bilgisayar teknolojilerinin çok hızlı ve çoğunlukla da ülkemiz dışında geliştirilmesi ve orada isimlendirilmesidir. Bu dezavantajlar üzerine bir de ülkemizde bilgisayar mühendisliği eğitiminin üniversitelerde çoğunlukla İngilizce veriliyor olmasını eklersek, bilişim terimlerinin neden bir türlü yaygın kabul görmediğini anlayabiliriz.

Ancak kimsenin fazla değinmediği önemli bir sebep, bu terimlerin üzerinde fazla düşünülmeden, yaratıcılık içermeyen, başka anlamlarla çakışmalara yol açan veya anlamını tam göbeğinden vuramayan kelimelerden seçilmiş olmasıdır. Bunun sorumlusu maalesef biz akademisyenleriz. Yeterince Türkçe yayın yapmıyor, yaptığımız zamanda bu yeni çıkan teknolojileri birebir Türkçeye çevirip geçiyoruz. Bu tavrımız da uzun vadede problemler çıkarıp sonunda günlük dilimizde yabancı terimlerin tercih edilmesine yol açıyor.

Bunlara örnek olarak "structure" kelimesinin Türkçeye "yapı" olarak geçirilmesidir. "Data structures" kelimesinin Türkçeye birebir "veri yapıları" olarak çevrilmesi başarıyla kabul görmüş olsa da C++ (ve kardeşleri Java, C#, vs.)

dilinde sıklıkla kullanılan “struct” kelimesiyle çakışmaktadır. Mesela şöyle bir cümleyi Türkçeye çevirdiğimizde:

“We will use a struct named “mystruct” to implement that data structure.”

“O veri yapısını inşa etmek için ismi “yapım” olan bir yapı kullanacağız.”

Bu tercümenin edebi açıdan ne kadar çirkin olduğunu görüyorsunuz. Buradaki problem “yapı” kelimesinin Türkçe’de çok geniş bir anlam ifade etmesidir. Bu sebeple bu kitaptaki önerimiz C++’daki “struct” için “ardışık” kelimesinin kullanılmasıdır. “Data structure” terimi için “veri yapısı” kullanılmaya devam edilecektir çünkü bu zaten son derece yerleşik bir terim haline gelmiştir. C++’daki struct, aslında arka arkaya yazılmış değişkenlerden ibarettir. Bunlar bilgisayar hafızasında da arka arkaya durdukları için “ardışık” kelimesi “struct” kelimesini İngilizcesinden bile daha iyi ifade etmektedir. Ayrıca Türkçede çok kullanılmayan bir kelime olduğu için çakışma olasılığı da azdır.

Kötü tercüme edilmiş terimlere en güzel örneklerden biri de İngilizce ‘deki “pixel” kelimesidir. Bu kelime “picture” yani resim kelimesi ile “element” kelimesinin yaratıcı bir şekilde harmanlanmasıyla ortaya çıkarılmış tamamen suni bir terimdir. Bu terim Türkçeye maalesef “benek” olarak çevrilmiştir. “Resim elemanı” bir pikselin manasını ne kadar iyi açıklıyorsa “benek” kelimesi de o kadar kötü açıklamaktadır. Bu nedenle günlük yaşamımızda hiç kabul görmemiştir. Web sitelerindeki bilişim konusunda yazılı teknik veya ticari yazıların %99’u hala piksel kelimesini kullanmaktadır. Biz de bu kitapta piksel kelimesini kullanmaya devam edeceğiz.

1.4. Yeni Terimler

1.4.1. Akışkan

Bilişim terimlerinden sıklıkla kullanılan “değişken” kelimesi programlamadaki “variable” kelimesinin karşılığı olarak başarı ile kullanılmaktadır. Yine programlama dillerindeki “object” kelimesi yerine “nesne” kelimesi de yerleşmiş şekilde kullanılmaktadır. Ancak I-See-Bytes kütüphanesinde kullanılan ICBYTES türündeki yapı ne değişken ne de nesnedir. İçerisinde matris, ikili ağaç, STL vektörü, küme gibi yapıların yanında, C++’daki temel veri tipleri olan int, float, long long gibi değişken türlerini de tutabilir. Asıl önemlisi koşma zamanında (run time) dinamik olarak bu yapıların birinden diğerine dönüşebilir. Sayılan nedenler yüzünden bu kitapta, ICBYTES cinsinden

nesnelere “akışkan” diye hitap edilecektir. Böylelikle koşma zamanında sabit tip tutan veri tipleri ve nesnelerden ayrılması sağlanacaktır.

1.4.2. Tip Kaydırma ve Tip Dönüştürme

Yeni uygulanacak terimlerden biri de “type casting” kelimesinin karşılığı olarak tip kaydırmadır. Type casting, birbirine uyumlu olan, mesela bir tür sayı tipinin başka bir tür sayı tipine çevrilmesidir. Maalesef bu terim “tip dönüştürme” olarak çevrilmektedir. Tip dönüştürme uyumlu olmayan tipler arasındaki dönüştürme için, yani “type conversion” için kullanılacaktır. Birbirine yakın olan tipleri dönüştürmek için daha yumuşak manası olan “kaydırma” kelimesi kullanılması daha mantıklıdır.

1.4.3. Dosyol

İç içe klasörlerden oluşan bir dosya sisteminde bir dosya veya dizin’in tam yerini tarif etmek için İngilizcede “path” kelimesi kullanılmaktadır. Maalesef bu terim Türkçeye son derece tembel bir şekilde “yol” veya “dosya yolu” olarak çevrilmiştir. (Bir de “yolak” diye bir kelime kullanılıyor ama onun ne olduğunu bilen ya da duyan yoktur.) İngilizce bir sözlüğe bakarsanız “yol” kelimesine karşılık gelen en az 20 değişik kelime bulabilirsiniz. Bu sebeple onlar için “path” kelimesine ikinci bir anlam yüklemek çakışmalara yol açmaz. Oysa bizde yol kelimesinin buradaki manasına karşılık gelen sadece “hat” (Arapça kökenlidir) ve “güzergah” (Farsça kökenlidir) kelimeleri vardır. Üstelik “yol” kelimesinin üzerine başka birçok anlam yüklenmiştir:

“Bu problemi bu yolla çözemeyiz.” → yöntem.

“Bu gittiğin yol, yol değil!”→ yaşam stili.

“Bir yola çıktık ki sorma.” → proje.

Daha önce belirttiğimiz gibi dilinize yeni bir terim kazandıracaksanız, şu üç kuralı yerine getirmeye çalışmalısınız:

- Var olan bir kelime kullanacaksanız günlük konuşmada veya terimin kullanılacağı bilim alanında zaten çok sık kullanılmakta olan bir kelime olmamalı. Aksi halde çakışmalara yol açarsınız.
- Terimi açıklayıcı bir kelime olmalı. Yani mantıksal bir geçiş veya bağ olmalı. Kelime zihninizde o anlamı çağrıştırmalı. Dosyol gibi.
- Mümkünse estetik ve yaratıcı olmalı. Aksi takdirde toplumda kabul görmez. (Piksel yerine Benek kelimesinin tutmayışı gibi.)

Bu sıraladığımız kurallara en güzel örneklerden biri “bilgisayar” kelimesidir. İngilizcedeki “computer” kelimesinin yerine kullanılan bu birleşik sözcük, gerçekten de bir bilgisayarın yaptığı işi “hesaplayıcı” kelimesinden çok daha iyi özetlemektedir. Yeni bir kelime olduğu için çakışma yaratmamıştır. Hal böyle olunca da toplumun geneli tarafından hızla kabul görmüştür. Bilgisayar kelimesi ilk defa Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyesi Prof. Dr. Aydın Köksal tarafından 1969 yılında gazeteye verilen bir ilanda kullanılmıştır.

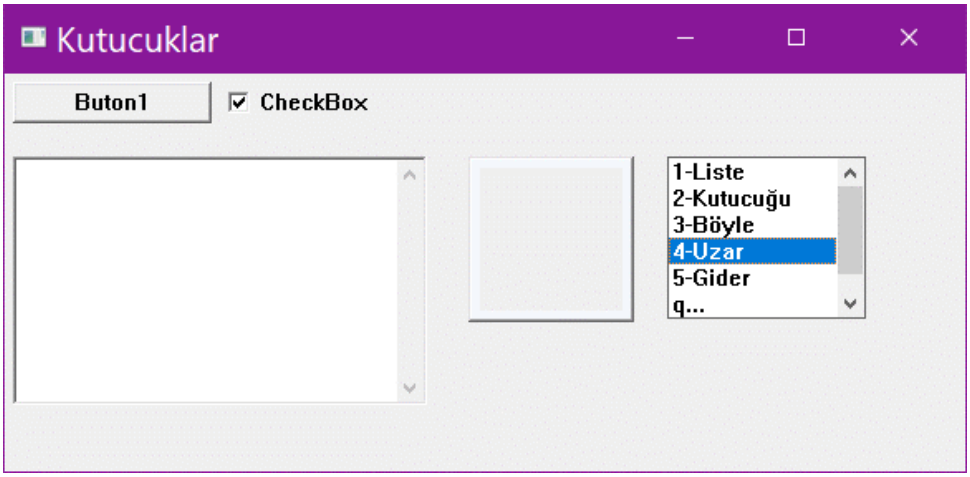
Bu kitapta “path” kelimesinin karşılığı olarak dosya yolunun kısaltması olan “dosyol” kelimesi kullanılacaktır. Aslında iç içe her klasörü birbirinin çocuğu ve ebeveyni olarak düşünürsek, soy ağacı manasına gelen “şecere” kelimesi daha uygun olacaktır ancak bu kelime Arapça kökenlidir.

Soru: Şu cümleyi saf Türkçeye çeviriniz: That approach is not the correct way to find the path of a file.

Cevap: O yol, bir dosyanın yolunu bulmak için doğru yol değildir.

1.4.4. Kutucuk

Aşağıdaki resimde Buton, metin kutusu, onay kutusu, panel ve benzeri araçlar gösterilmiştir. Windows işletim sisteminde bu araçların hepsi “Window” yani pencere denilen bir nesneden (object) türetilir. Zaten bu işletim sisteminin ismi de oradan gelir. Diğer işletim sistemleri bu tür grafik kullanıcı arayüzü nesnelerine “widget” (cihaz) diye isimlendirir. Bu kelime genelde daha yaygın kullanılmaktadır. Bildiğimiz kadarı ile bu kelimeye henüz Türkçede bir karşılık atanmamıştır. Bu nedenle “widget” kelimesi karşılığı olarak bu kitapta “kutucuk” kelimesi kullanılacaktır.



1.4.5. Kulp

İşletim sistemleri programlara ayırdıkları kaynakları betimlemek için "handle" adı verilen değişkenler kullanır. Bu değişkenler genelde tam sayı ya da işaretçi olabilirler. Mesela bir dosya açıldığında işletim sistemi genelde buna tamsayı bir "handle" atar. Windows işletim sistemi her kutucuk, sicim ve aklınıza gelebilecek her nesne için işaretçi türünden bir "handle" atar. Bu terime karşılık olarak Türkçede farklı farklı kelimeler atanmaktadır. Bu kitapta "handle" kelimesinin karşılığı olarak "kulp" terimi kullanılacaktır.

1.4.6. Demirbaş

Bir sistemde ilk açılış anında veya değiştirilmesi için özel emir verilmediği takdirde kullanılan ayarlar, nesneler, renkler, yazı stilleri ve benzeri sıfatlar İngilizcede "default" kelimesiyle adlandırılmaktadır. Bu kelime Türkçeye "varsayılan" diye son derece bariz, yaratıcılıktan uzak ve çakışma olasılığı çok yüksek bir kelimeyle tercüme edilmiştir. Biz bunun yerine "demirbaş" kelimesini kullanacağız.

Soru: Şu iki cümleyi Türkçeye çeviriniz: Without counting, you assumed that there were only 10 default settings. As a result, your program tried to change the nonexistent default settings.

Cevap: Saymadan sadece 10 tane varsayılan ayar olduğunu varsaydın. Sonuç olarak, programın var olmayan varsayılan ayarlarını değiştirmeye çalıştı.

1.4.7. Yaratık

1980'ler ve 90'larda popüler olan, iki boyutlu basit grafiklere sahip, diğer adı arcade denilen oyun türleri özellikle çok sicimli (multithreaded) ve oyun programcılığın eğitiminde sıklıkla kullanılmaktadır. I-See-Bytes kütüphanesi, çok sicimli programlama veya oyun programlamaya giriş dersleri için rahatlıkla kullanılabilir çünkü içerisinde oyun programcılığında kullanılan ve İngilizce "sprite" ya da "moving object block" (MOB) olarak adlandırılan basit iki boyutlu grafik nesneleri yaratabilmek için gerekli olan birçok fonksiyon bulunmaktadır. Bu fonksiyonlar anlatılırken İngilizce'deki "sprite" (peri, cin hayalet vb.) kelimesi karşılığı olarak "yaratık" kelimesi kullanılacaktır.

1.4.8. Sicim

Her ne kadar bilişim sözlüğünde "thread" terimi "iş parçacığı" olarak çevrilmiş olsa da bu kitapta eskiden birçok yazarın kullanmış olduğu "sicim" kelimesi kullanılacaktır. Çünkü "iş parçacığı" iki kelimeden oluşan uzun ve anlam

çakışmalara yol açabilecek kötü bir tercihtir. Yani 1.4.3'te sıralanan özelliklerin hiçbirine uymamaktadır.

1.5. Bu Kitaptaki Terimlerin İngilizce Karşılıkları

Aşağıdaki tabloda bu kitapta kullanılan yeni oluşturulmuş Türkçe terimlerin İngilizce karşılıkları verilmiştir.

Türkçe	İngilizce
Ardaşık (veya Ardışık)	struct (C++)
Demirbaş	Default
Dosyol	Path
Kulp	Handle
Kutucuk	Widget
Tip Kaydırma	Type casting
Tip Dönüştürme	Type conversion
Çerçeve (resim gösteren kutucuk)	Frame
Buton	Button
Yaratık	Sprite (Game Programming)
Sicim	Thread
Piksel	Pixel

1.6. Fonksiyon İsimlendirme Kuralları

I-See-Bytes kütüphanesi içerisinde iki temel akışkan vardır. Bunlar:

1. ICBYTES: Veri ve veri yapıları tutar.
2. ICDEVICE: Bilgisayara bağlı veri alışverişi yapılabilen (mesela kamera) her türlü cihaza erişim sağlar.

Eğer fonksiyon girdilerinden en az biri kütüphanemize özgü olan bu akışkanlardan biri ise, fonksiyon isim çakışması olmayacağı için fonksiyon ismi manasına özgü olarak direk yazılır. Örneğin:

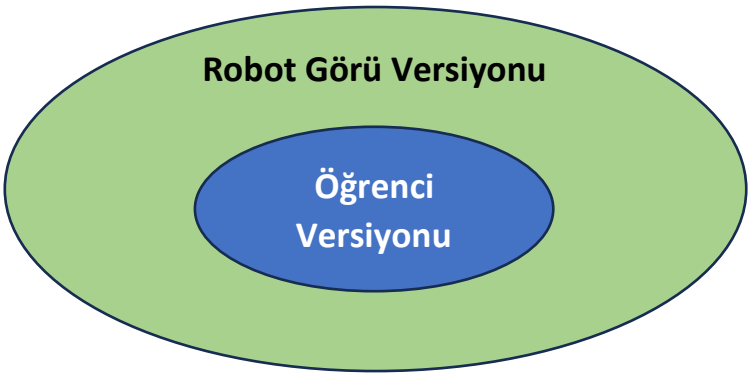
CreateMatrix()

Ancak durum böyle değilse, fonksiyonun başına şu eklemelerden biri yapılmıştır:

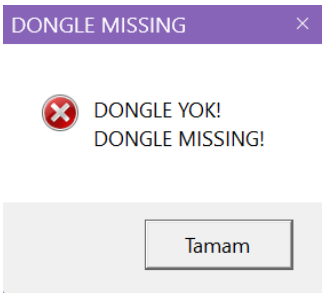
- ICB_
- ICG_

1.7. I-See-Bytes Versiyonları

I-See Bytes Kütüphanesinin temelde iki farklı versiyonu bulunmaktadır. Bunlar "Öğrenci" (Student) ve Robot Görü (Robot Vision) versiyonlarıdır. Ancak bu versiyonlar tamamen uyumludur. Sadece daha üst versiyonların özellikleri daha fazladır. Bu demek oluyor ki, eğer yazdığınız program öğrenci versiyonunda çalışıyorsa, robot görü versiyonunda da sorunsuz çalışacaktır. Bu uyumluluğu aşağıdaki şekli kullanarak daha iyi tarif edebiliriz:



Robot görü versiyonu, öğrenci versiyonunun tüm özelliklerini kapsar. Aradaki diğer önemli fark doğal olarak fiyatlarıdır. Öğrenci versiyonu çok daha ucuzdur. Satın alındığında bu kütüphaneler bir USB dongle ile çalışır. Bu dongle, USB hafıza çubuğuna benzeyen bir cihazdır ve kütüphanenin korsan olarak çoğaltılmasını önler. Öğrenci versiyonu dongle olmadan 15 dakika çalışabilir. Robot görü kütüphanesi ise dongle olmadan 5 dakika çalışır. Bu süreler yazılan programların çalışma süresi içindir. Microsoft Visual Studio'da program yazmak için değil. Yani .exe programlarınızın çalışma süresidir. Çoğu öğrenci projesi için 15 dakika çalışma süresi yeterlidir. Kütüphaneyi satın almadan evvel geliştirme yapabilirsiniz. Ancak bu kütüphaneyi kullanarak bir yazılım satacaksanız o zaman almanız gerekir. Süre sonunda ekranda dongle olmadığına dair bir mesaj görürsünüz. Sonra da programınız aniden kapanır.



Elinizdeki bu kitap sadece ve sadece öğrenci versiyonunun özelliklerini anlatır. Robot görü versiyonu "I-See-Bytes ile Uygulamalı Robot Görü" kitabında anlatılmaktadır.

2. Grafik Arabirimi Alt Kütüphanesi: ICGUI

Grafik arabirimi bilgisayar programcılığının vazgeçilmez bir parçası haline gelmiştir. Piyasada kimse konsol programı yazmadığına göre programcılara konsol üzerinde programlama öğretme konusunda ısrar etmenin gereği kalmamıştır. I-See-Bytes kütüphanesi içinde, grafik arabirimi programlamasını sağlamak üzere çok sayıda fonksiyon bir alt kütüphane olarak sunulmuştur. Bu kütüphaneneye de I-See-GUI ismi verilmiştir. Kısaca ICGUI olarak yazılmaktadır. Bu kütüphane WIN32 API üzerine kurulmuştur. WIN32 API Windows işletim sisteminin en alt düzeydeki, en temel programlama arabirimidir. Bunun sonucu olarak herhangi ekstra bir DLL vs. gibi dosyalara ihtiyaç duymadan her Windows makinada çalışabilmektedir.

WIN32 API, Windows'un en temel programlama arabirimi olması nedeniyle son derece hızlı çalışır ki bu I-See-Bytes kütüphanesinin en önemli hedeflerinden biridir. En popüler C++ derleyicileri olan Microsoft Visual Studio ve GNU C++ (MinGW) derleyicileri tarafından desteklenir.

Ancak WIN32 API kullanılmasının çok önemli bir avantajı daha vardır. WINE (Wine Is Not an Emulator) Emulatörü kullanarak Linux ve MacOs işletim sistemleri üzerinde sorunsuz olarak ve hemen hemen aynı hızda çalışabilmektedir. WINE, açık kaynak kodlu bir uyumluluk katmanıdır. Yani kodunuz Java'da olduğu gibi sanal işlemci üzerinde değil, direk olarak gerçek işlemci üzerinde çalışır. Dolayısı ile çok hızlıdır. Bunun sonucu olarak I-See-Bytes kütüphanesi kullanarak yaratacağınız EXE dosyalar WINE yüklenmiş Linux üzerinde sorunsuzca çalışır. Bu da Java'ya olan ihtiyacı önemli ölçüde azaltır.

2.1. Merhaba Dünya

Aşağıdaki kod I-See-GUI kütüphanesinin basitliğini göstermektedir. Bilindiği gibi her C++ programı "main()" fonksiyonu ile başlar. ICGUI kütüphanesinde ise programı başlatan iki fonksiyon vardır:

void ICGUI_Create(){}

void ICGUI_main(){}

Bu iki fonksiyon üstte gösterildiği üzere boş halde çağrılsa (daha doğrusu tanımlansa) bile 800x600 piksel büyüklüğünde bir pencere yaratırlar. Ancak tabii ki bu pencere üzerinde başka bir kutucuk (widget) olmaz. Kutucuklar yaratmak için ICGUI_main() içerisinde kutucuk yaratan fonksiyonların çağrılması gerekir.

```
#include"icb_gui.h"

int MLE; //Çok Satırlı metin düzenleyici penceresi kulpu (handle)

void ButonFonksiyonu(); //Buton fonksiyonu deklare edildi

void ICGUI_Create()//Ana pencere yaratıldı
{ }

void ICGUI_main()//C++ olmazsa olmazı main
{
    ICG_Button(5, 5, 120, 25, "BUTON", ButonFonksiyonu);
    MLE = ICG_MLEditSunken(5, 80, 400, 400, "", SCROLLBAR_V);
}

void ButonFonksiyonu()//Buton'a basınca olacaklar
{
    for(int x=1;x<11;x++)
        ICG_printf(MLE, "%d-Merhaba Dünya!\n",x);
}
```

Yukarıdaki kodda bir adet buton (ICG_Button)ve bir adet metin düzenleyici (ICG_MultilineEditSunken) kutucukları yaratılmıştır. Kutucuk yaratan tüm fonksiyonlar bir tam sayı döndürürler. Bu tamsayı o kutucuğun kulpu (handle'ı) olmaktadır. Bu tanımlayıcı kullanılarak programın başka bir köşesinde o kutucuğa erişmek mümkün olur. Genelde bu tanımlayıcılar global olarak tanımlanır ve böylelikle her fonksiyon tarafından erişebilinirler.

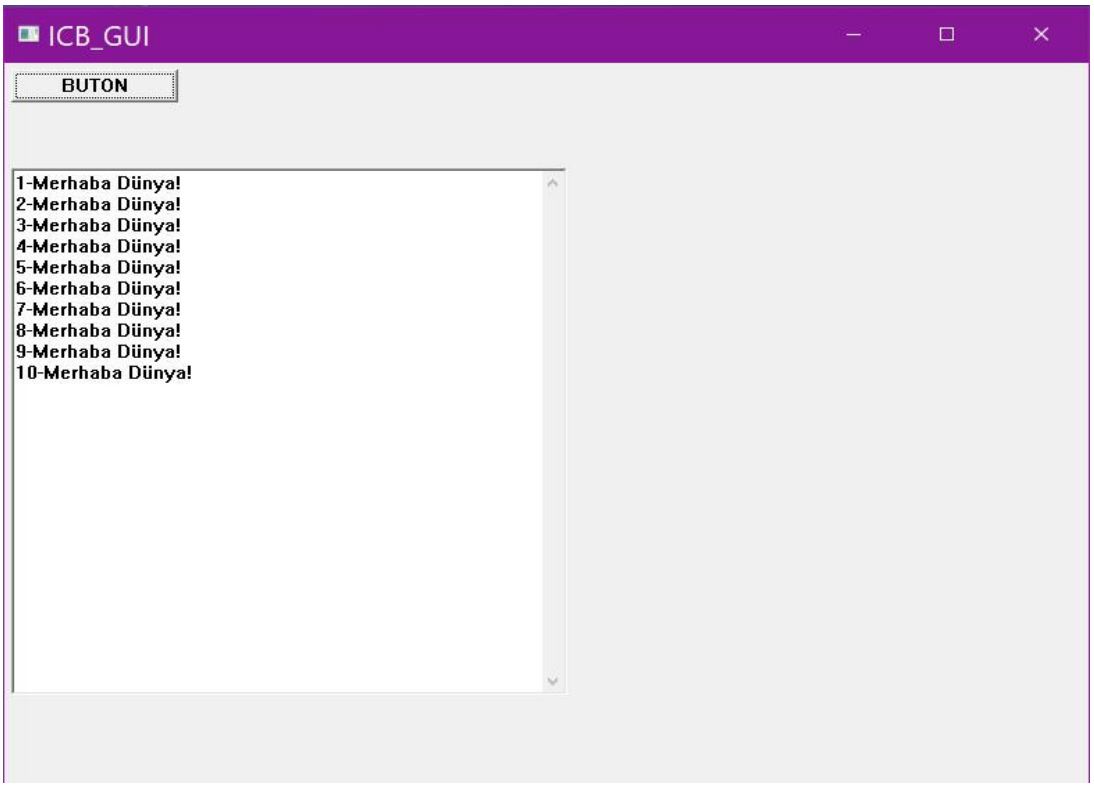
```
ICG_Button(5, 5, 120, 25, "BUTON", ButonFonksiyonu);
```

Butonların tanımlayıcısına genelde ihtiyaç duyulmaz çünkü yaratıldıktan sonra program sonlanıncaya kadar üzerlerine bir değişiklik meydana getirmek gerekmez. Kutucuk yaratan fonksiyonlarının ilk 2 argümanı her zaman koordinat bilgisidir. Kutucuğun sol üst köşesinin penceremizin neresinde yer alacağının piksel cinsinden koordinatlarıdır. Sonraki iki argüman ise yine piksel cinsinden eni ve boyudur. Beşinci argüman, ilk yaratıldığında kutucuğun üzerinde yahut içerisinde yazacak yazıdır ve tırnak içerisinde yazılır ("BUTON") gibi veya char cinsinden pointer olabilir. Son argüman ise butona basıldığında çağrılacak olan fonksiyonun ismidir.

```
MLE = ICG_MLEditSunken(5, 80, 400, 400, "", SCROLLBAR_V);
```

Bu fonksiyonu çağırarak kenarları içe göçükmüş gibi görünen bir "çok satırlı" (Multi Line Edit(MLEdit)) metin düzenleme kutucuğu yaratıyoruz. İlk yaratıldığında içerisinde bir şey yazmasını istemiyorsak beşinci argümanı arka arkaya boşluksuz iki tırnak ("") olarak belirliyoruz. Son argüman ise dikey bir sürgüsü (vertical scrollbar) olmasını istediğimizi belirtiyor.

Buton'a basıldığında "Butonfonksiyonu()" isimli fonksiyon çağrılıyor. Bu fonksiyon önce ilan edilmiş (declared) sonra da tanımlanmıştır. Basit bir for döngüsü ile ekrana10 kez "Merhaba Dünya" yazdırılıyor. Yazdırmak için kullanılan fonksiyon:



Şekil 1.1. Buton'a basılınca 10 kere "Merhaba Dünya!" yazan program.

```
    ICG_printf(MLE, "%d-Merhaba Dünya!\n",x);
```

C++ programcılarının çok iyi bildikler "printf()" fonksiyonunun hemen hemen aynısıdır. Tek farkı ilk argümanının metin düzenleme kutucuğunun kulpu (handle'ı) olmasıdır. Pencerenizde birden fazla metin düzenleme kutucuğu olabilir. Kulpunu girdi olarak vererek hangi kutucuğa yazmasını istediğinizi belirtmiş oluyorsunuz. Aşağıda bu programın sonucu gösterilmiştir.

2.2. En, Boy, Renk ve Stil

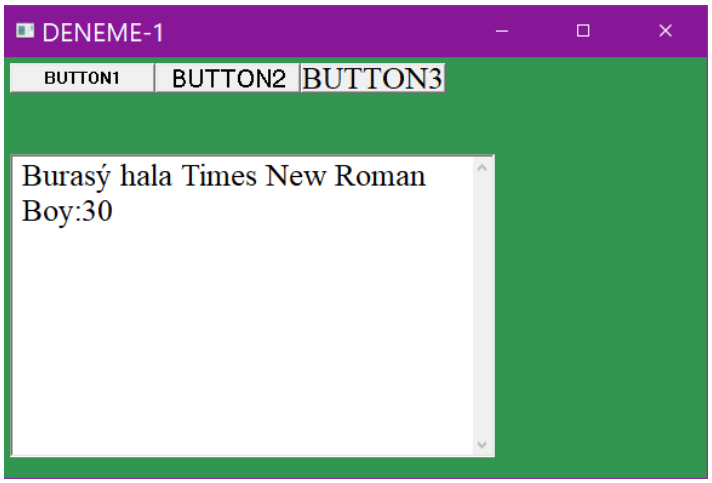
Aşağıdaki kod örneğinde, ana penceremizi yaratan "ICG_GUICreate()" fonksiyonu içerisinde üç tane fonksiyonun çağrıldığını görüyoruz:

```
#include"icb_gui.h"
int MLE;
void ButonFonksiyonu();

void ICGUI_Create()
{
 ICG_MWSize(600, 400);
 ICG_MWTitle("DENEME-1");
 ICG_MWColor(50, 150, 80);
}

void ICGUI_main()
{
 ICG_Button(5, 5, 120, 25, "BUTTON1", ButonFonksiyonu);
 ICG_SetFont(25, 0, "Ariel");
 ICG_Button(125, 5, 120, 25, "BUTTON2", ButonFonksiyonu);
 ICG_SetFont(30, 0, "Times New Roman");
 ICG_Button(245, 5, 130, 25, "BUTTON3", ButonFonksiyonu);
 MLE = ICG_MLEditSunken(5, 80, 400, 250, "Burası hala Times \\
New Roman Boy:30", SCROLLBAR_V);
}

void ButonFonksiyonu()//Buton'a basınca olacaklar
{}
```



Yukarıda görüldüğü üzere, uygulamamızın görünümünde belirgin değişiklikler meydana gelmiştir. Bu değişikliklerden en göze çarpanı pencere renginin yeşil olmasıdır. Ayrıca pencerenin boyu da küçülmüştür. İşte bu değişiklikleri ortaya çıkaran fonksiyonlar "ICGUI_Create()" fonksiyonu içerisindeki üç adet fonksiyon tarafından gerçekleştirilmiştir. Şimdi bunları sırayla inceleyelim:

ICG_MWSize(600, 400)

Bu fonksiyon, uygulamamızın ana penceresinin 600x400 piksel boyunda olmasını sağlamaktadır. Bu fonksiyonu kullanmadığımızda ön tanımlı (default) olarak uygulamamızın boyutu 800x600 piksel olmaktadır.

ICG_MWTitle("DENEME-1")

Bu fonksiyon ana penceremizin ismini değiştirmek için kullanılır. Bu fonksiyonu kullanmazsak, bir önceki örnekte olduğu gibi ön tanımlı olan "ICB_GUI" kullanılır.

ICG_MWColor(50, 150, 80)

Ana penceremizin renginin değişmesini sağlayan fonksiyon. Bilgisayarda tüm renkler kırmızı, yeşil ve mavi renklerin karışımı olarak yaratılır. Bu fonksiyonun girdileri, pencere renginin bu üç rengin hangi parlaklıkta karışımı olacağını belirtmektedir. Kırmızı için 50, yeşil için 180 ve mavi için de 80. Bu değerler 0 ile 255 arasında olmak zorundadır. Hepsi 0 veya 0'a yakın olduğunda siyah renk ortaya çıkar. Hepsi aynı olursa grinin tonları oluşur. Hepsi 255 olduğunda ise beyaz renk elde edilir.

Bu uygulamada göze çarpan diğer değişiklik butonlarda farklı font tipi ve boyut kullanılmış olmasıdır. Button1 için demirbaş sistem fontu kullanılmış iken, Button2 için boyu 25 olan Ariel fontu, Button3 için ise 30 boyutunda Times New Roman fontu kullanılmıştır. Bu font değişikliklerini yaratan:

```
ICG_SetFont(25, 0, "Ariel");
```

fonksiyonudur. Bu fonksiyonun ilk iki girdisi fontun yüksekliği ve genişliğidir. Ancak genişlik sıfır verilirse yükseklik değeri her ikisi için de kullanılır. Üçüncü girdi ise fontun ismini belirtmektedir. Dikkat edilirse, bu fonksiyon sadece kendinden sonra yaratılan kutucukları etkilediği görülecektir. Yani bu fonksiyonu bir kere çağırdıktan sonra yaratacağınız buton veya metin kutusu gibi tüm kutucuklar aynı font kullanacaklardır. Bunun örneği Button3 yaratıldıktan sonra metin kutucuğu yarattığımızda görülmektedir. Bu metin kutucuğu hala Times New Roman fontu kullanmaktadır.

Çok satırlı metin kutucuğunda yer alan "Burasý hala Times New Roman Boy:30" yazısının hatalı yazılmış olmasının sebepleri ileriki bölümlerde anlatılacaktır.

2.3. ICGUI ve ICBYTES

Bu kısımda grafik arabirimi kütüphanesi olan ICGUI ile akışkan yaratan ICBYTES kütüphanesinin birlikte nasıl çalıştığına dair örnekler işleyeceğiz. Bu örnekler ile ICBYTES tipi ile yaratılan akışkanların ne kadar değişik amaçlarla kullanılabileceğini de öğrenmiş olacağız.İlk örneğimizde ICBYTES'ın temel kullanım amaçlarından biri olan matris yapısı ile ilgili fonksiyon ve arabirimleri göreceğiz. Aşağıdaki kod örneğinde iki adet buton tanımlanmaktadır. Bu butonlara basıldığında çağrılan fonksiyonlar ile iki farklı şekilde matris yaratma yöntemi gösterilmiştir.

```
#include"icb_gui.h"
int MLE,FRM;
void MatrisYaz(); //Matris Butonu fonksiyonu deklare edildi
void Resim(); //Resim Butonu fonksiyonu deklare edildi

void ICGUI_Create()
{
 ICG_MWSize(600, 400);
 ICG_MWTitle("ICGUI ve ICBYTES");
}

void ICGUI_main()
{
 ICG_SetFont(25, 0, "Consolas");
 ICG_Button(5, 5, 120, 25, "Matris", MatrisYaz);
 ICG_Button(125, 5, 120, 25, "Resim", Resim);
 MLE = ICG_MLEditSunken(5, 80, 250, 250, "", SCROLLBAR_V);
 FRM = ICG_FramePanel(260, 80, 250, 250);
}

void MatrisYaz()//Matris'e basınca olacaklar
{
 ICBYTES A = {{1,2,3},{4,5,6},{7,8,9}};
 DisplayMatrix(MLE, A);
 A= "ABCDEF";
 DisplayMatrix(MLE, A);
 Print(MLE, A);
}

void Resim()//Resim'e basınca olacaklar
{
 ICBYTES A;
 CreateMatrix(A, 250, 250, 3, ICB_UCHAR);
 A = 100;
 for (int x = 50; x < 200; x++)
   A.B(x, 100, 2) = 255;
 DisplayImage(FRM, A);
}
```

Bu örnekte, "ICGUI_Main()" fonksiyonu içerisinde yeni bir tür kutucuk yaratılmaktadır. Bu kutucuk resim göstermek için kullanılıp ismi Frame'dir. Frame, İngilizcede çerçeve manasına gelir. Yani resmi gösteren bir çerçeve yaratıyoruz. Bu fonksiyonun isminin "FramePanel" olması onun şeklinin bir panel gibi dışa doğru olmasından kaynaklanmaktadır. ICGUI içerisinde farklı stilde çerçeveler bulunmaktadır. Bunlardan ileriki bölümlerde bahsedilecektir.

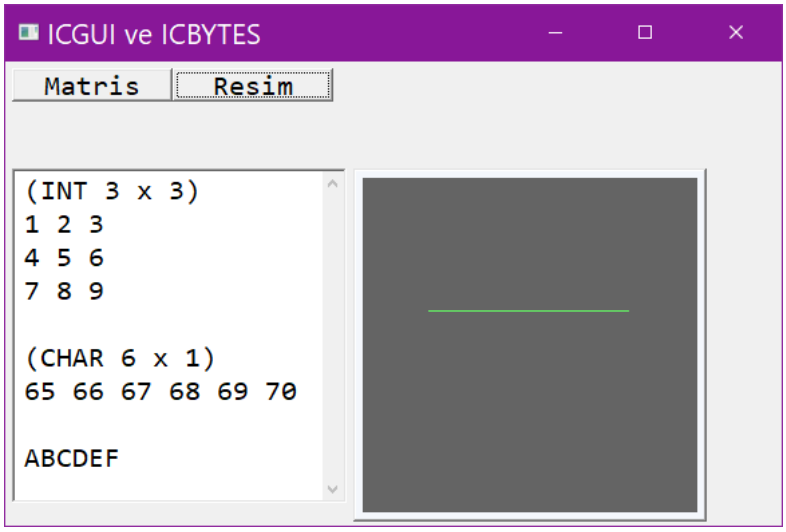
Birinci butona gelirsek, bu butonun kodunda ICBYTES cinsinden bir akışkan yaratılmakta, ve içerisinde 3x3 boyutunda tam sayılardan oluşan bir matris yaratılmaktadır. Bunu yapan satır aşağıda tekrar gösterilmiştir:

```
ICBYTES A = {{1,2,3},{4,5,6},{7,8,9}};
```

Daha sonra bu matrisi çok satırlı metin düzenleyici içerisinde yazdırmak için "DisplayMatrix()" isimli fonksiyon kullanılmıştır. Hemen arkasından, A akışkanı içerisine "ABCDEF" karakterleri yazılmıştır:

```
A= "ABCDEF";
```

Yani A akışkanı iki boyutlu bir dizi taşırken aniden bir karakter dizisi (dizgi) taşımaya başlamıştır. Daha sonra A akışkanı iki farklı şekilde çok satırlı metin kutucuğuna aşağıda görüldüğü şekilde yazdırılmıştır.



İkinci butona ait olan "Resim()" adlı fonksiyon içerisinde ise farklı bir yöntemle matris yaratılmaktadır:

```
ICBYTES A;
CreateMatrix(A, 250, 250, 3, ICB_UCHAR);
```

Önce ismi "A" olan bir akışkan yaratılmakta, daha sonra bu akışkanın 250x250x3 boyutlarında ve "unsigned char" cinsinden bir matris taşıması komutu

verilmektedir. Daha evvelki bölümde, bilgisayarda tüm renklerin kırmızı, yeşil ve mavi tonlarının karışımından yaratıldığını söylemiştik. Bu nedenle eğer 250x250 piksel boyutlarında renkli bir resim yaratmak istersek, matrisimizin X ve Y boyutlarının 250 olması gerekir. Her bir renk için de 3 tane değer (kırmızı, yeşil ve mavi için) olması gerektiğine göre matrisimizin boyutunun neden 250x250x3 olması gerektiği anlaşılır.

ICBYTES içerisindeki matrisler en fazla dört boyutlu olabilir ve her bir boyutu (X,Y,Z,W) olarak adlandırılır. Bu sıralama şekli Matlab ve OpenCV'den farklıdır ve bu yazılımların kullanıcıları bu konuda dikkatli olmalıdırlar. C/C++ kullanıcıları da matris indekslerinin array'lerin aksine sıfırdan değil birden başladığını unutmamalıdırlar.

Daha sonraki satırda:

```
A = 100;
```

Eşitlemesi ile matrisin içerisindeki tüm elemanlar 100'e eşitlenmektedir. Yani bu tek satırda, 250x250x3=187.500 elemanın içerisine 100 yazılmaktadır. Bunun sonucu olarak resmimizin tamamı koyu gri tonuna sahip olmaktadır. Daha evvel belirmiş olduğumuz üzere, eğer kırmızı, yeşil ve mavi (bundan böyle KYM olarak kısaltılacaktır) tonlarına aynı değerleri yüklersek grinin tonlarını elde etmiş oluruz.

Daha sonra bu gri renk üzerine yatay yeşil bir çizgi çizen satırları inceleyelim:

```
for (int x = 50; x < 200; x++)
   A.B(x, 100, 2) = 255;
```

Burada bir "for" döngüsü ile x değerinin 50'den 199'a kadar arttırıldığını ve her seferinde A matrisinin 100'üncü satırının 50, 51, 52, … yatay piksellerine ulaşılıp ikinci, yani yeşil tonunun maksimum değer olan 255'e yükseltildiğini görmekteyiz. Bu da (50,100) den (199,100)'e kadar olan piksel renginin koyu griden yeşile dönüşmesine yol açıp birbirine bitişik olan bu minik piksellerin gözümüze bir çizgi gibi gözükmesine yol açmaktadır. Burada unutmadan değinmemiz gereken önemli bir nokta matris elemanlarına ulaşırken kullandığımız:

```
A.B()
```

ifadesidir. "A" akışkanı arkasından nokta koyup B() yazmamız, A matrisine (B)yte yani "unsigned char" olarak erişmek istediğimizi göstermektedir. Bu şekilde A matrisinin istediğimiz elemanlarını okuyabilir veya değerlerini değiştirebiliriz. Kodumuzun son satırındaki:

```
DisplayImage(FRM, A);
```

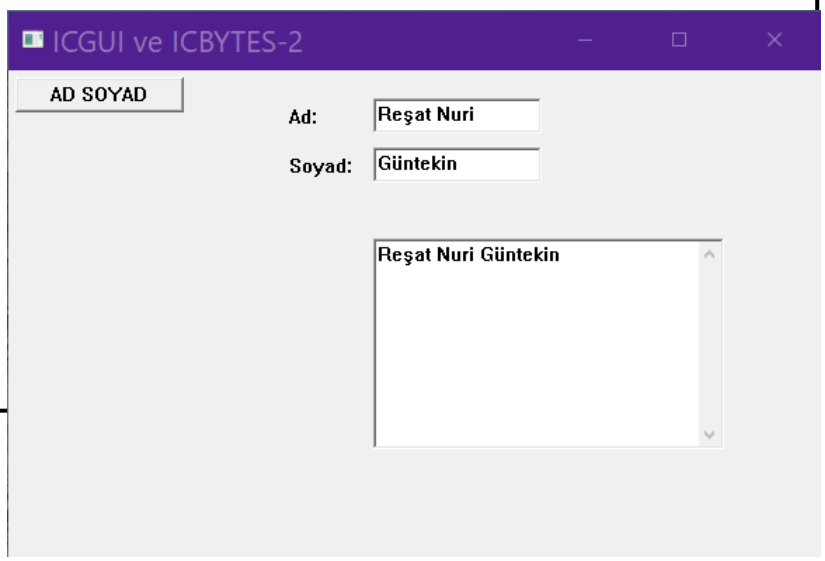
satırı ise “A” matrisini bir resim (imaj) olarak “FRM” çerçevesi içinde gösterilmesi komutunu vermektedir. Bu çerçeve, ICGUI_Main() içerisinde “ICG_FramePanel()” fonksiyonunu çağırarak yarattığımız çerçeve olduğunu hatırlayınız.

2.4. Ekrandan Yazı Okuma (Girdi)

Hemen her program kullanıcıdan bilgi alma ihtiyacındadır. ICBYTES kütüphanesi ekrandan bilgi almayı da kolaylaştırmıştır. Herhangi bir metin kutucuğundaki karakter dizini ICBYES akışkanı tarafından kolaylıkla kopyalanabilir. Aşağıdaki örnekte kullanıcı adını ve soyadını tek satırlı metin kutucuklarına girmektedir. Daha sonra programımız “AD SOYAD” butonuna basıldığında bunları çok satırlı bir metin kutucuğuna aralarında bir boşluk bırakıp yazmaktadır.

```
#include"icb_gui.h"
int SLE1,SLE2,MLE;
void adsoyad(); //AD SOYAD butonu fonksiyonu deklare edildi
void ICGUI_Create()
{
  ICG_MWSize(600, 400);
  ICG_MWTitle("ICGUI ve ICBYTES-2");
}
void ICGUI_main()
{
  ICG_Button(5, 5, 120, 25, "AD SOYAD", adsoyad);
  ICG_Static(200, 25, 40, 25, "Ad:");
  ICG_Static(200, 60, 60, 25, "Soyad:");
  SLE1 = ICG_SLEditSunken(260, 20, 120, 25, "");
  SLE2 = ICG_SLEditSunken(260, 55, 120, 25, "");
  MLE = ICG_MLEditSunken(260, 120, 250, 150, "", SCROLLBAR_V);
}

void adsoyad()
{
  ICBYTES ad, soyad;
  GetText(SLE1, ad);
  GetText(SLE2, soyad);
  Print(MLE, ad);
  ICG_printf(MLE," ");
  Print(MLE, soyad);
}
```



Programımızı incelediğimizde ICGUI_main() içerisinde yeni bir fonksiyon ile karşılaşıyoruz:

```
ICG_Static(200, 25, 40, 25, "Ad:");
ICG_Static(200, 60, 60, 25, "Soyad:");
```

Bu fonksiyon penceremizin gri arka planı üzerinde "Ad:" ve altına da "Soyad:" diye yazdıran fonksiyonlardır. Aslında görünmeyen bir pencere içerisine statik, yani kullanıcının değiştiremeyeceği (programcının değil) bir yazı yazmayı sağladığı için bu tür kutucuğa "statik" kutucuğu denmektedir. Pencerenin koordinatları, eni ve boyu fonksiyonun ilk dört argümanı tarafından belirlenmektedir. Bunların hemen yanında yer alan tek satırlı (Single Line Edit (SLEdit)) metin kutucukları kullanıcıya adlarını ve soyadlarını hangi kutucuklara yazmaları gerektiğini işaret etmektedir. Pencere görüntüsüne bakıldığında Türkçe karakterlerin sorunsuz biçimde gösterildiği gözükmektedir. Bunun sebebi Windows'un halihazır (default) fontunu kullanıyor olmamızdır.

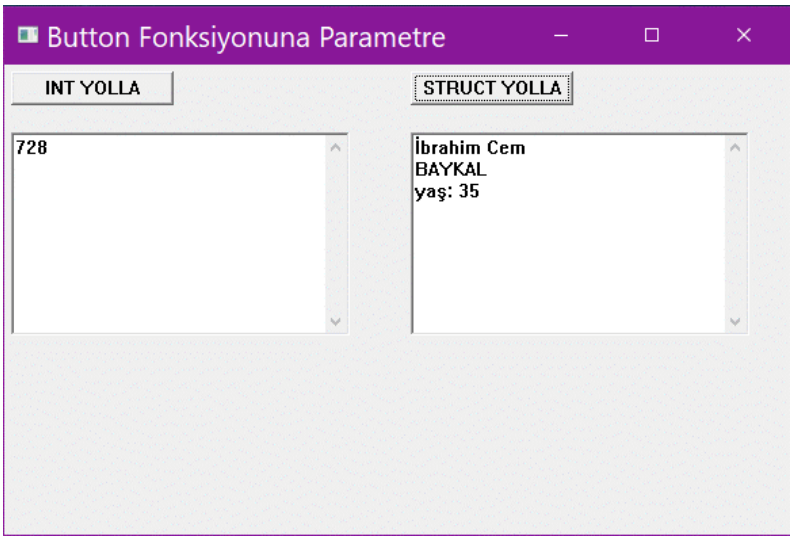
Butona basıldığında çağrılan "adsoyad()" isimli fonksiyona gelirsek, burada da yeni bir fonksiyonla karşılaşıyoruz:

```
GetText(SLE1, ad);
```

Bu fonksiyon içerisinde kullanıcı tarafından "Reşat Nuri" yazılı olan tek satırlı metin kutucuğundaki yazıyı, yani "Reşat Nuri" yazısını "ad" isimli akışkana kopyalar. Bundan sonraki satırda da aynı fonksiyon ikinci tek satırlı metin kutucuğundaki "Güntekin" yazısını "soyad" isimli akışkana kopyalar. Daha sonra gelen üç satırdaki fonksiyonlar da bu akışkanları çok satırlı metin kutucuğuna (Multi Line Edit (MLEdit)) aralarına bir adet boşluk bırakarak yazdırır.

2.5. Buton Fonksiyonuna Parametre Yollama

Bu ana dek anlatılan programlarda bir buton'a basıldığında çağrılan fonksiyonun nasıl tanımlandığı gösterildi ancak bu fonksiyona ICGUI_main() içerisinden nasıl parametre yollanacağından bahsedilmedi. İstendiği takdirde buton fonksiyonuna sadece bir parametre yollanabilir ve bu parametre void (boş) cinsi bir işaretçi (pointer) olmak zorundadır. İşaretçi, herhangi bir değişken veya nesnenin hafızadaki adresidir. Yani değişken veya akışkan yollamak yerine onların adresi yollanmaktadır. Kullanıcılar fonksiyonlara genelde birden fazla parametre yollarlar. Üstelik de bunlar genelde int, float veya double türü olur. Hal böyleyken ICGUI kütüphanesi neden sadece bir parametreye izin vermekte ve üstelik te bunun bir işaretçi olmasını istemektedir? Çünkü işaretçi hafızada bir adrestir ve o adreste sıralı bir şekilde duran sayısız değişken bu yöntemle fonksiyona iletilebilir.



Yukarıda ekran görüntüsü verilmiş olan uygulamada iki tane buton bulunmaktadır. Bu butonlardan üzerinde "INT YOLLA" olanına bastığınızda sol taraftaki metin kutusuna 728 yazmaktadır. Bu sayı aslında "int" cinsinden bir değişken olarak ICGUI_main() içerisinde yaratılmakta, sonra da bu butonun fonksiyonuna yollanmakta ve bu fonksiyon tarafından soldaki metin kutucuğuna yazdırılmaktadır. Sağdaki "STRUCT YOLLA" butonu ise iki adet karakter dizini ve bir adet tamsayı yollamakta, aynı şekilde onun fonksiyonu da bunları sağdaki metin kutucuğuna yazdırmaktadır. Öyleyse bu nasıl gerçekleşmektedir? Aşağıdaki kod örneği bunun nasıl yapıldığını adım adım anlatmaktadır.

```
#include"icb_gui.h"
#include<string.h>

int MLE1, MLE2;

struct insan
{
  char ad[32];
  char soyad[32];
  unsigned int yas;
};

void intfonksiyon(void*p);
void structfonksiyon(void* p);

void ICGUI_Create()
{
  ICG_MWSize(600, 400);
  ICG_MWTitle("Button Fonksiyonuna Parametre");
}
```

Programımızın başında çok satırlı metin kutularının kulpları (MLE1,MLE2) tanımlanmıştır. Daha sonra bir ardışık (struct) tanımlandığını görmekteyiz. Bu struct'ın tip tanımı "insan" olarak belirlenmiştir ve üç adet üyesi bulunmaktadır. Birinci üye ismi "ad" olan 32 harf uzunluğunda bir karakter dizinidir. İkincisi yine aynı özelliklere sahip "soyad" isimli bir dizin, üçüncüsü ise "yas" adlı bir pozitif tam sayıdır.

Bu ardaşık'ın tanımından sonra buton fonksiyonlarının tanımlarını görmekteyiz. Bu seferki tanımlarda fonksiyonların girdi olarak void türünde bir işaretçi aldığını görebiliriz. Bu tanımlardan sonra klasik içeriğe sahip bir ICGUI_Create() fonksiyonu gelmektedir. Şimdi gelelim programımızın ana fonksiyonuna:

```
void ICGUI_main()
{
  static int i=728;
  static insan C;
  strcpy_s(C.ad, "İbrahim Cem");
  strcpy_s(C.soyad, "BAYKAL");
  C.yas = 35;
  ICG_Button(5, 5, 120, 25, "INT YOLLA",intfonksiyon,(void*)&i);
  ICG_Button(300,5,120,25,"STRUCT YOLLA",structfonksiyon,(void*)&C);
  MLE1 = ICG_MLEditSunken(5, 50, 250, 150, "", SCROLLBAR_V);
  MLE2 = ICG_MLEditSunken(300, 50, 250, 150, "", SCROLLBAR_V);
}
```

Ana fonksiyon içerisinde "i" adında bir tam sayı yaratıldığını görüyoruz. Buton fonksiyonlarına girdi olarak yollanacak her türlü değişkenin "static" cinsinden tanımlanması gerekiyor. Bunun sebebi "ICGUI_main()" ana fonksiyonu sona erdikten sonra dahi bu değişkenlerin varlıklarını sürdürme zorunluluğudur. Bu örnekte olduğu gibi ana fonksiyon, kutucukları yarattıktan sonra sona erebilir. Bu programın sona ereceği manasına gelmez. Burada da "i" tam sayısını yarattıktan sonra içerisine 728 yazıyor, ve onun adresini buton fonksiyonuna kopyalıyoruz. Kendisini değil.

Bundan sonra ismi "C" olan "insan" tipinde bir struct yaratıyoruz. Bunu yaratırken "static" kelimesini kullanmayı unutmuyoruz. Daha sonra C++ diline ait tümce kopyalama fonksiyonunu kullanarak C'nin "ad" dizini içerisine içerisine bir isim, "soyad" dizinine de soy isim yazıyoruz. C'nin yaş elemanına da 35 yazınca bu struct button fonksiyonuna yollanmaya hazır oluyor.

Buton fonksiyonlarını incelediğimizde iki farklı yöntemle tip kaydırma (type casting) uygulandığını görüyoruz:

```
void intfonksiyon(void *p)
{
  ICG_printf(MLE1, "%d ", *(int*)p);
}

void structfonksiyon(void* p)
{
  insan *i = (insan*)p;
  ICG_printf(MLE2, "%s\n", i->ad);
  ICG_printf(MLE2, "%s\n", i->soyad);
  ICG_printf(MLE2, "yaş: %d\n", i->yas);
}
```

Birinci fonksiyonda tip kaydırma için şu ifade kullanılıyor:

```
*(int*)p
```

Bu ifadede "p" işaretçisinin önündeki parantez içindeki "`(int*)`" ifadesi p'yi void türünde işaretçiden tam sayı türünde işaretçiye kaydırıyor. Daha sonra ise "`(int*)`" ifadesinin önündeki "*" yıldız, bu işaretçinin gösterdiği tam sayı değerini bize veriyor ve bu değer de soldaki metin kutucuğuna yazdırılıyor.

İkinci fonksiyonda ise void türünden işaretçimiz p, tip kaydırma kullanılarak "`insan`" cinsinden bir ardışık işaretçisine dönüştürülüp aynı cinsten fonksiyonun içinde yaratılmış olan "i" işaretçisine kopyalanıyor. Bundan sonra "`insan`" ardışığın elemanlarına "i" üzerinden ulaşılarak sağdaki metin kutusu içerisine birer birer yazdırılıyorlar.

2.6. Resim Yükleme

I-See-Bytes öğrenci versiyonu jpeg ve bitmap resimleri okuyup matrsi haline dönüştürebilmektedir. Aşağıdaki kod örneğinde bunun nasıl yapıldığı gösterilmiştir. Bu örnekte farklı olarak "icbimage.h" başlık dosyası da eklenmiştir. Bu dosyada resimi yüklememizi sağlayacak fonksiyonların tanımları bulunmaktadır. İki tane kulp tanımlanmıştır.

```
#include"icb_gui.h"
#include "icbimage.h"

int MLE1,FRM1;

void butonfonk();
```

Kodun geri kalan kısmında çoğu bilindik fonksiyonlar vardır. Sadece buton fonksiyonunda farklı komutlar görmekteyiz.

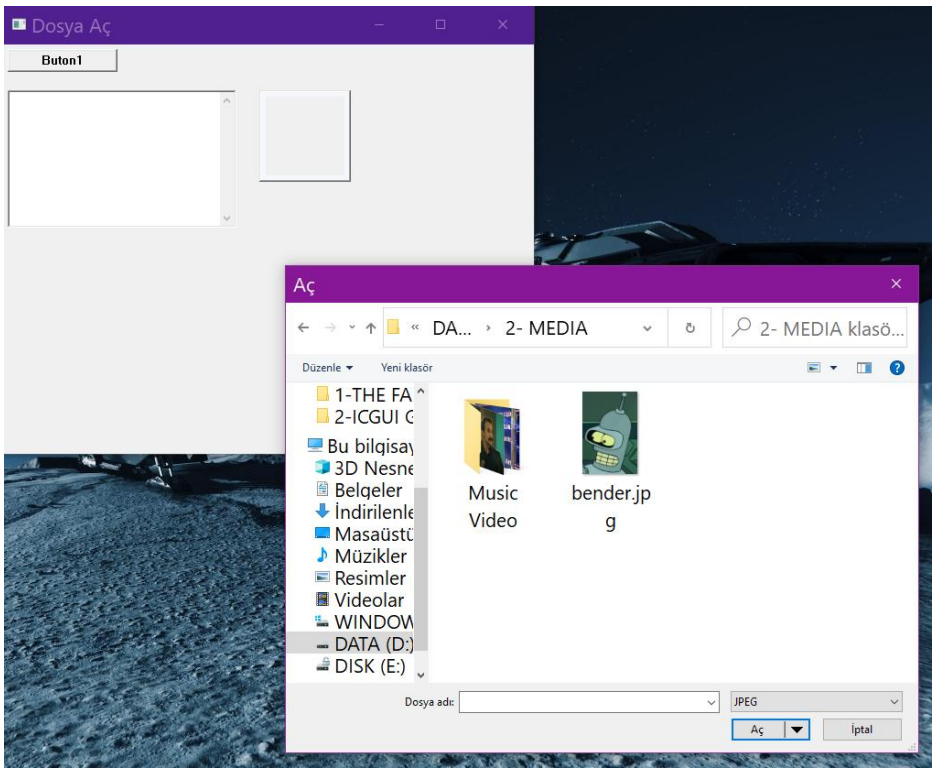
```
void ICGUI_main()
{
  ICG_Button(5, 5, 120, 25, "Buton1", butonfonk);
  MLE1 = ICG_MLEditSunken(5, 50, 250, 150, "", SCROLLBAR_V);
  FRM1=ICG_FramePanel(280, 50, 100, 100);
}

void butonfonk()
{
  ICBYTES dosyol,resim;
  ReadImage(OpenFileMenu(dosyol, "JPEG\0*.JPG\0"), resim);
  Print(MLE1, dosyol);
  DisplayImage(FRM1, resim);
}
```

Buton fonksiyonunda iki adet akışkan yaratılmıştır. Bunların ne amaçla kullanılacağı zaten isimlerinden belli olmaktadır. Daha sonra iki adet iç içe kullanılmış fonksiyon görüyoruz. Önce içerdeki fonksiyon çalıştırılacağı için "`icb_gui.h`" içerisindeki tanımına bakalım:

```
char* OpenFileMenu(ICBYTES& path, const char* type);
```

Bu fonksiyon girdi olarak bir akışkan ve bir de karakter işaretçi almaktadır. Bu komut çalıştırıldığında aşağıdaki dosya seçme penceresi açılır:



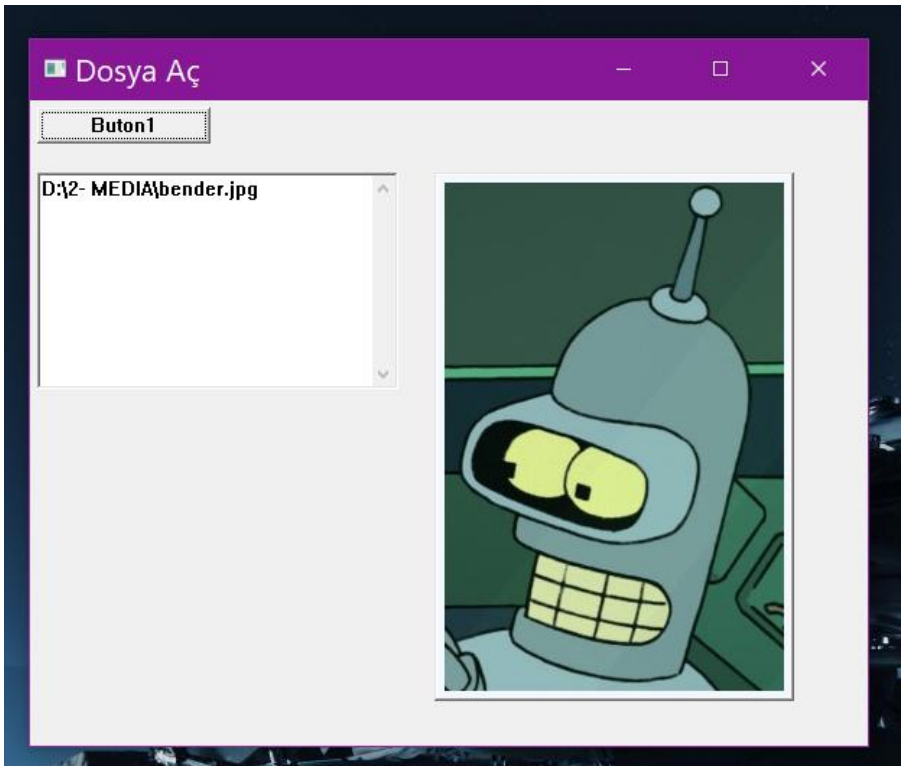
Bu pencerenin alt köşesine dikkat ederseniz, sadece jpeg türündeki resim dosyalarını seçebilmenize izin verdiğini görürsünüz. Bunun sebebi bu fonksiyona ikinci argüman olarak "JPEG\0*.JPG\0" karakter dizgisinin verilmiş olmasıdır. Bu dizgi, sadece "*.JPG" uzantılı dosyaların seçilebileceğini belirtir. Birinci argüman olan akışkana ise seçilen dosyanın ismi ve dosyolu yerleştirilir. Bu dosyol, ayrıca bir karakter dizgisine işaretçi olarak fonksiyon tarafından döndürülür. İşte bu işaretçi, dışardaki "ICB_ReadImage()" fonksiyonuna girdi olarak kullanılır. Bu fonksiyonun "icbimage.h" başlık dosyasındaki tanımını incelersek,

```
bool ICB_ReadImage(const char * filepath, ICBYTES& i);
```

ilk girdisinin karakter bir işaretçi, ikincisinin ise ICBYTES akışkanı cinsinden olduğunu görebiliriz. İngilizce isminden de anlaşıldığı gibi, darakter işaretçi, bir dosyanın ismi ve tam dosyolunu içeren bir dizgiyi göstermektedir. İşte bu işaretçi, "OpenFileMenu()" fonksiyonunun döndürdüğü işaretçidir. Eğer dosyol geçerli ise ve jpeg yada bitmap bir dosyanın dosyolu ise "ReadImage()" fonksiyonu bu dosyayı açıp bir matrise yükler ve bu matrisi de "resim" isimli akışkana yükler. Arkadan gelen şu iki satır ise aşağıdaki ekran görüntüsünü oluşturur.

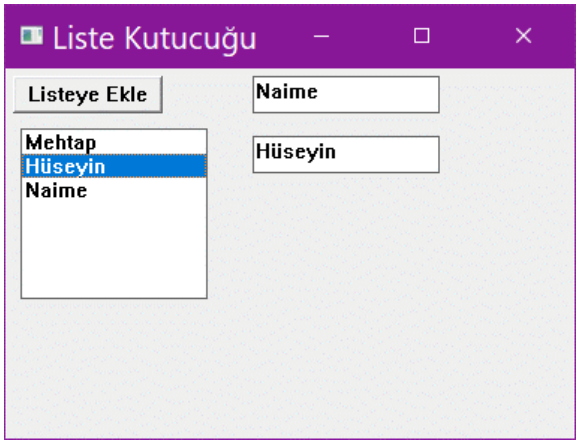
```
Print(MLE1, dosyol);
DisplayImage(FRM1, resim);
```

Bu örnekte D: diskinde "2-MEDIA" dizini içindeki "bender.jpg" isimli resim dosyası yüklenip çerçeve içerisinde gösterilmiştir.



2.7. Liste Kutucuğu

Kelime yada cümlelerden oluşan maddeleri seçilebilir bir liste halinde göstermenizi sağlayan grafik arabirimi yapısını liste kutucuğu adını veriyoruz. Aşağıda bu kutucuğu kullanan bir uygulamanın ekran görüntüsü verilmiştir. Bu uygulamada sağ üst taraftaki tek satırlı metin kutucuğuna yazdığınız dizgi "Listeye Ekle" butonuna her bastığınızda liste kutucuğuna farklı bir madde olarak eklenir. Örneğin aşağıdaki resimde görülüyor ki, sırayla "Mehtap," "Hüseyin," ve "Naime" üstteki metin kutucuğuna yazılmış (Naime yazan kutucuk) ve sonra her seferinde butona basılmıştır.



Bu üç isim listeye eklendikten sonra liste kutucuğunda ikinci sırada olan "Hüseyin" kelimesi üzerine fare imleci ile gelip sol tuşa çifte tıklama yaptığımızda seçilmiş olan madde (ki bu durumda Hüseyin'dir) sol alttaki ikinci metin kutusu içerisine yazılmıştır. Aşağıda bu uygulamanın C++ kodu verilmiştir.

```
#include"icb_gui.h"

int LB1,SLE1,SLE2;

void ICGUI_Create()
{
  ICG_MWTitle("Liste Kutucuğu");
  ICG_MWSize(400, 300);
}

void ListboxFonk(int indeks)
{
  ICBYTES yaz;
  ICG_GetListItem(LB1, indeks,yaz);
  SetText(SLE2, yaz);
}
```

```
void ButonFonk()
{
  ICBYTES oku;
  GetText(SLE1, oku);
  ICG_AddToList(LB1, &oku.C(1));
}

void ICGUI_main()
{
  ICG_Button(5, 5, 100, 25, "Listeye Ekle", ButonFonk);
  SLE1= ICG_SLEditBorder(165, 5, 125, 25, "");
  SLE2 = ICG_SLEditBorder(165, 45, 125, 25, "");
  LB1 = ICG_ListBox(10, 40, 125, 125,ListboxFonk);
}
```

ICGUI_main içerisinde ilk dikkatimizi çeken liste kutucuğu yaratan ICG_ListBox fonksiyonudur. Tüm diğer kutucuk yaratma fonksiyonlarında olduğu gibi bu fonksiyonun da ilk dört argümanı, pozisyonunu ve boyutlarını belirler. Beşinci argüman ise tamsayı (int) cinsinden bir girdisi olan "ListBoxFonk" isimli fonksiyondur ve yukarıda tanımlanmıştır. Bu fonksiyon, siz liste kutucuğu üzerindeki bir maddeye çifte tıklama yapıldığında çağrılır. ICG_ListBox'ın tanımını incelediğimizde aslında isteğe bağlı bir argümanı daha olduğunu görürüz:

```
int ICG_ListBox(int x, int y, int width, int height, void (*callback)(int),
                bool sort=false);
```

Sondaki bool cinsinden "sort" isimli argüman isteğe bağlı olarak "true" verilirse, liste kutucuğun içerisine yazılan maddeleri yollanma sıralarına göre değil, alfabetik olarak sıralar. Bizim de yaptığımız gibi bu argüman verilmezse demirbaş değeri olan false kullanılır.

Buton fonksiyonunu incelediğimizde "oku" isimli bir akışkana SL1 kulpu kullanılarak üstteki metin kutucuğunun içinde yazanların aktarıldığını, daha sonra da bu akışkanın liste kutucuğuna eklenmesi için ICG_AddToList() fonksiyonunun çağrıldığını görmekteyiz. Bu fonksiyonun ikinci argümanı char cinsinden bir işaretçi olması gerektiği için akışkanın içindeki vektörün birinci elemanının adresi yollanmaktadır.

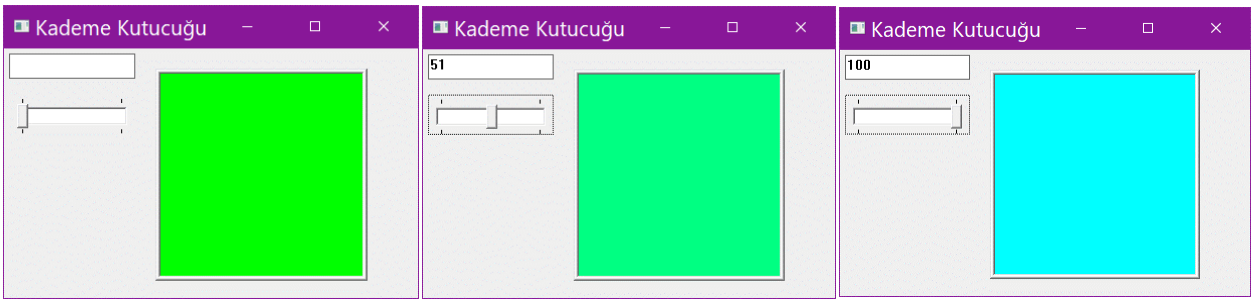
Bu kod örneğinde ilk kez tam sayı (int) girdi alan bir kutucuk fonksiyonu ile karşılaşıyoruz:

```
void ListboxFonk(int indeks)
```

Buradaki girdi "indeks" çift tıkladığınız liste maddesinin numarasıdır. Bu numara sıfırdan başlar. Programın sonraki adımında ICG_GetListItem() fonksiyonuna bu indeks numarası verilerek o indeks sırasındaki maddenin "yaz" isimli bir akışkana kopyalandığını görüyoruz. Son adımda ise bu akışkan, SLE2 kulpu ile ulaşılan metin kutucuğuna yazdırılıyor. Yani liste kutucuğundaki "Hüseyin" maddesine çift tıklandığında fonksiyonumuza indeks numarası olarak bir geliyor, bu indeks numarası ile birinci maddeyi "yaz" akışkanına kopyalıyor, sonra da bu akışkanın içeriğini ikinci metin kutusuna yazdırıyoruz.

2.8. Kademe Kutucuğu

Elektronik cihazların üzerinde bulunan, sesi düzeyi veya görüntü parlaklığı gibi ayarlar değiştirmek için sağa veya sola çekilerek kullanılan ayar düğmelerine benzer görünümlü kutucuğa kademe kutucuğu diyoruz. Bu kutucuk yatay olarak yaratıldığında sağa sola veya dikey olarak yaratıldığında aşağı yukarı kaydırılarak belli değerler arsında tam sayılar üretir. Bu tam sayılar, programdaki diğer fonksiyonlara parametre olarak kullanılır. Aşağıdaki resimde kademe kutucuğu kullanılarak bir resmin rengini değiştiren örnek bir uygulama gösterilmiştir. Bu uygulama ilk başlatıldığında en soldaki pencere görülmektedir. Çerçeve içindeki resim saf yeşil renktedir. Daha sonra ortadaki resimde gösterildiği üzere kademe orta noktaya çekilince yeşil saflığını yitirmektedir. En sağda ise kademe ayarı son noktaya kadar arttırıldığında resim saf turkuaz rengine dönüşür. Bu uygulamada kademe ayarının aldığı değere göre resim saf yeşil ile saf turkuaz arasında herhangi bir tona dönüşebilir. Bunu televizyondaki renk ayarı gibi düşünebilirsiniz.



Şimdi gelelim bu uygulamanın kaynak koduna:

```
#include"icb_gui.h"

int FRM1,SLE1;
ICBYTES resim;
```

```
void ICGUI_Create()
{
  ICG_MWTitle("Kademe Kutucuğu");
  ICG_MWSize(430, 300);
}

void KademeFonk(int kademe)
{
  ICG_SetWindowText(SLE1, "");
  ICG_printf(SLE1, "%d", kademe);
  resim = 0xff00+ kademe * 2.55;
  DisplayImage(FRM1, resim);
}

void ICGUI_main()
{
  CreateImage(resim, 200, 200, ICB_UINT);
  SLE1= ICG_SLEditBorder(5, 5, 125, 25, "");
  ICG_TrackBarH(5, 45, 125, 40, "", KademeFonk);
  FRM1 = ICG_FrameMedium(150, 20, 200, 200);
  resim = 0xff00;
  DisplayImage(FRM1, resim);
}
```

ICGUI_main() fonksiyonunda kademe kutucuğunu yaratan fonksiyon:

```
ICG_TrackBarH(5, 45, 125, 40, "", KademeFonk);
```

Dört adet pozisyon ve ebat büyüklüğü dışında bir de “KademeFonk” isimli bir fonksiyonu argüman olarak almaktadır. Burada dikkat edilmesi gereken husus, kademe kutucuğunun dikey yüksekliğidir. Bu kutucuğun tam olarak çizilebilmesi için en az 40 piksel genişlikte olması gerekmektedir.

ICGUI_main() fonksiyonunda daha önceki örnek programlardan farklı olarak resim akışkanı içerisinde 200x200 piksel boyutlarında “`unsigned int`” cinsinden bir matris yaratılmıştır. Daha evvelki örneklerde 24 bit renkli resim oluşturmak için şöyle bir komut kullanmıştık:

```
CreateMatrix(A, 250, 250, 3, ICB_UCHAR);
```

Üstteki komut 250x250 boyutunda 3 kanallı (KYM) ve her kanal için 8 bitlik bir matris oluştururken şimdi kullandığımız komut

```
CreateImage(resim, 200, 200, ICB_UINT);
```

200x200 tek kanallı, ancak kanalın kendisinin 32 bit olduğu bir matris yaratmıştır. Bu tür bir resim alfa için 8, KYM için de 24 bit ayırır.

I-See-Bytes kütüphanesinde 32 bitlik resim matrisi iki farklı şekilde yaratılabilir. İster bizim yaptığımız gibi 200x200 boyutunda tek kanalı 32 bit olan bir matris yaratabilirsiniz, isterseniz de 200x200x4 boyutunda her kanalı 8 bit olan bir matris yaratabilirsiniz:

```
CreateImage (A, 200, 200, 4, ICB_UCHAR);
```

Bunların ikisi de aynı tip resim üretir. Ancak bu matris elemanlarına ulaşmak isterseniz farklı yöntemlerle ulaşmanız gerekir.

Main fonksiyonun devamında, resim matrisi yaratıldıktan sonra, beşinci satırında şöyle bir ifade görmekteyiz:

```
resim = 0xff00;
```

Bu ifadede resim matrisinin her elemanını onaltılık düzendeki 0xff00 sayısına eşitlemektedir. Bu sayı resmin her pikselini saf yeşile dönüştürür. Peki bu sayının manası nedir? Bunu anlamak için öncelikle onaltılık düzende yazılan her basamağın 4 bite karşılık geldiğini hatırlamamız gerekir. Yani 8 bit için iki basamak, 32 bit için ise sekiz basamak kullanmamız gerekir. Biz her elemanı 32 bitlik bir sayı olan bir matrise yazmaya çalıştığımıza göre bu sayıyı aslında şöyle yazmamız gerekirdi:

```
resim = 0x0000ff00;
```

Ancak bir sayının önündeki sıfırlar manasız olduğu için biz daha kısa olan versiyonu seçtik. Uzun şekliyle yazıldığında her çift basamak bir renk kanalına denk gelir:

```
0x 00 00 ff 00
   A  K  Y  M    (Alfa, Kırmızı, Yeşil, Mavi)
```

ff sayısı, onluk düzende, bir kanalın alabileceği maksimum değer olan 255 ettiği ve diğer renk tonlarının değeri sıfır olduğu için uygulamamız çalıştırılır çalıştırılmaz ekrana saf yeşil bir resim basmaktadır.

Main fonksiyonundaki yaratılma tamamlandıktan sonra, artık aksiyon sadece kademe kutucuğunun fonksiyonu olan KademeFonk() içerisinde gerçekleşmektedir. Yukarıdaki kodda görüldüğü üzere bu fonksiyona girdi olarak “kademe” adlı bit tam sayı gelmektedir. Bu fonksiyon kademe çubuğuna her dokunulduğunda, yani yeri değişmediği durumlarda bile, çağırılmaktadır. Fonksiyon girdisi olan tam sayının aralığı demirbaş olarak 0-100 arasında ayarlanmıştır. Ancak bu aralık başka fonksiyonlarla değiştirilebilir.

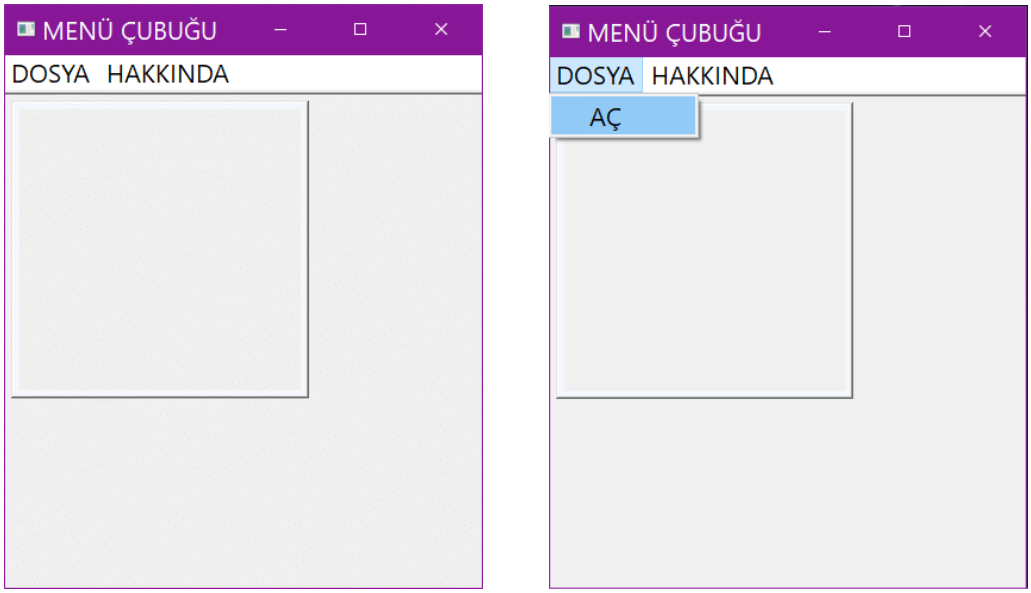
Adım adım bu fonksiyonu incelersek, ilk satırda **SLE1** kulpunun bağlı olduğu metin kutusunun içini sildiğini fark ederiz. Daha sonra “kademe” değişkeninin değerini bu metin kutusuna yazdığını, sonra da resim matrisinin her elemanını şu değere eşitlediğini görürüz:

```
resim = 0xff00+ kademe * 2.55;
```

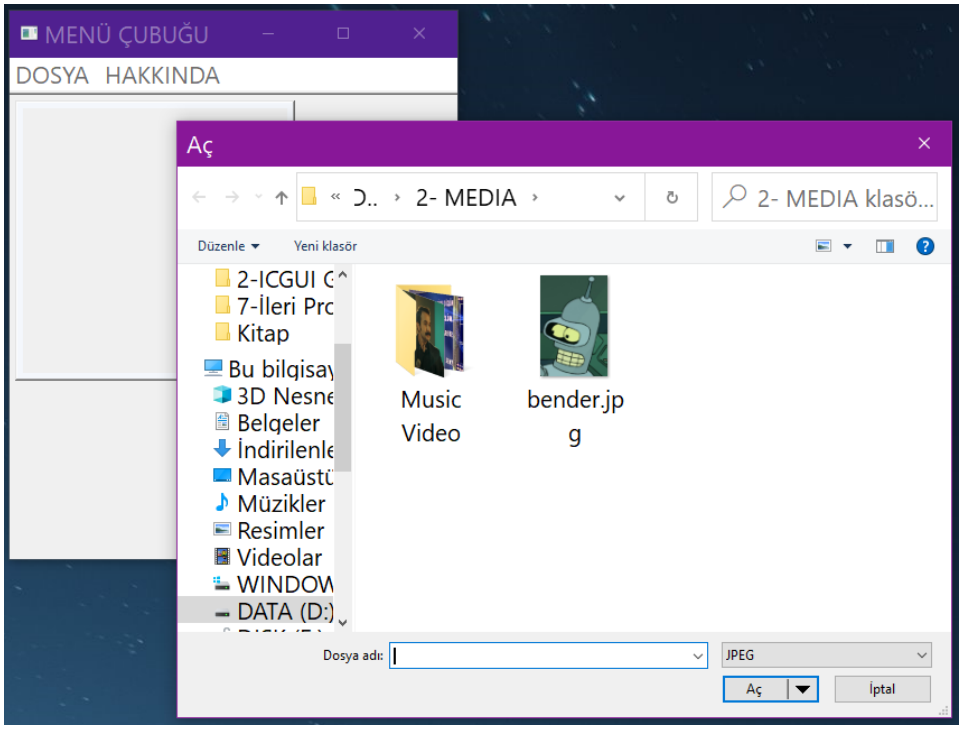
bu satırda daha evvel matrisi yeşile boyamak için kullandığımız 0xff00 değeri 0 ile 255 arasında bir sayı ile toplanmaktadır. Yani onaltılık düzende 0x0000 ile 0x00ff arasında bir sayı ile toplanmakta ve matrisin tamamı bu sayı haline gelmektedir. 0xff00 ile 0x00ff'i toplarsak taşma olmaz. Yani yeşil tonu maksimum değeri olan 255'de kalmaya devam ederken, biz kademe çubuğunu her arttırışımızda mavi tonunu güçlendiririz. Sonuçta da hem mavi hem de yeşil tonu maksimum değeri olan 255'e ulaştığında saf turkuaz rengini elde etmiş oluruz.

2.9. Menu Ekleme

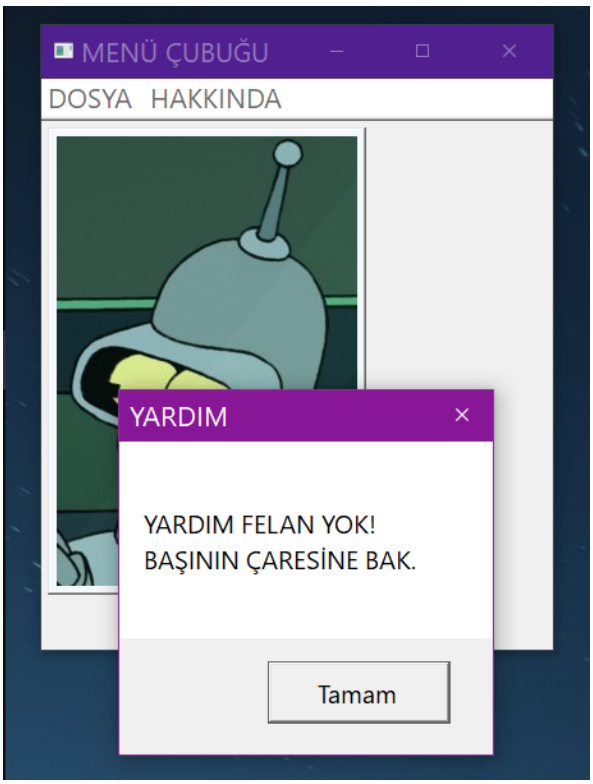
Profesyonel uygulamaların çoğunda ana pencerenin üst kısmında bir menü çubuğu bulunur. Bu menü çubuğundan birçok işlem gerçekleştirilebilir. Çok fazla ayarı veya işlemi olan uygulamalarda buton kullanmak çok yer kaplamaktadır. Menüler yüksek sayıda butonu dar bir alana sıkıştırma yolu olarak görülebilir. Aşağıda menü çubuğu olan bir uygulama gösterilmektedir. Bu uygulamanda menü çubuğunda sadece "DOSYA" ve "HAKKINDA" diye iki kategori bulunmaktadır. Menüdeki Dosya seçeneğine tıkladığımızda "AÇ" diye bir seçenek daha belirmektedir.



Aç seçeneğine bastığımızda ise aşağıda gösterildiği üzere bölüm 2.6'da öğrenmiş olduğumuz dosya açma menüsü karşımıza gelir.



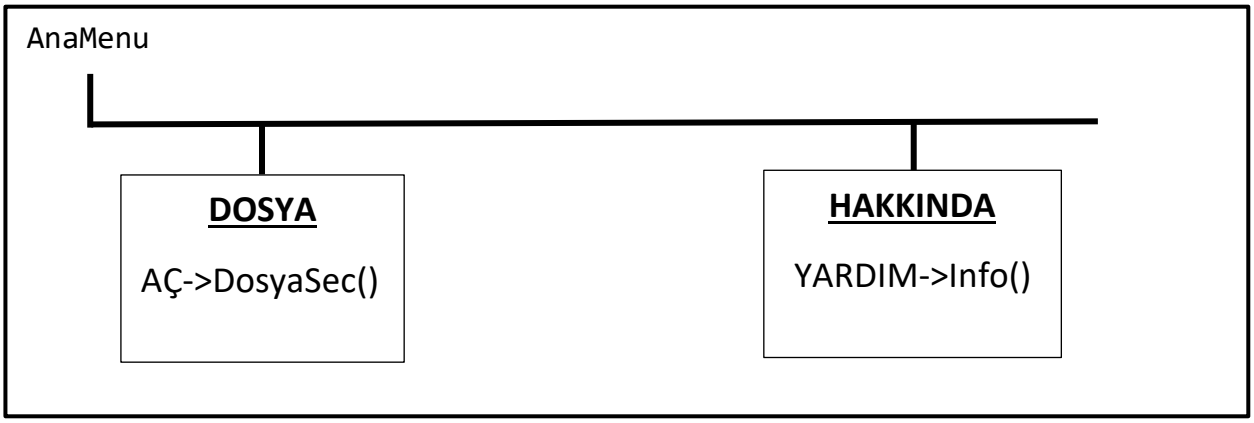
Bender.jpg resmini yükledikten sonra menü çubuğundaki "HAKKINDA" seçeneğine tıkladığımızda ise onun altında yer alan seçenek(ler) karşımıza gelecektir. Bu örnekte sadece "YARDIM" diye bir seçenek çıkmaktadır. Bu seçeneğe tıkladığımızda ise aşağıda sağda görülen mesaj kutucuğu karşımıza çıkar.



Şimdi bu uygulamanın kaynak kodunu inceleyelim. ICGUI_main() fonksiyonu içerisinde alışageldiğimiz biçimde bir resim çerçevesi yaratıldığını görüyoruz. Hemen arkasından yeni bir değişken türü ile karşılaşıyoruz: HMENU. Bu değişken tipinin baş harfinden bunun Windows işletim sistemine ait bir kulp (HANDLE) olduğunu tahmin edebiliriz. Bu cinsten üç farklı kulp yaratıldığını görüyoruz. Bunlar AnaMenu, DosyaMenu ve HakkindaMenu. Bu üçünün uygulamada sırayla menü çubuğunun kendisi, menü bardaki "DOSYA" ve "HAKKINDA" seçeneklerine karşılık geldiğini tahmin edebiliriz.

```
void ICGUI_main()
{
  FRM = ICG_FramePanel(5, 5, 250, 250);
  HMENU AnaMenu, DosyaMenu, HakkindaMenu;
  AnaMenu = CreateMenu();
  DosyaMenu = CreatePopupMenu();
  HakkindaMenu = CreatePopupMenu();
  ICG_AppendMenuItem(DosyaMenu, "AÇ", FileFunc);
  ICG_AppendMenuItem(HakkindaMenu, "YARDIM", Info);
  AppendMenu(AnaMenu, MF_POPUP, (UINT_PTR)DosyaMenu, "DOSYA");
  AppendMenu(AnaMenu, MF_POPUP, (UINT_PTR)HakkindaMenu, "HAKKINDA");
  ICG_SetMenu(AnaMenu);
}
```

Daha sonra menü çubuğunun kulbunu Win32 API'nin kendi fonksiyonlarından biri olan CreateMenu() fonksiyonu ile yaratılan menü çubuğuna bağlıyoruz. Diğer iki seçeneği ise önce içlerinde "AÇ" ve "YARDIM" diye alt seçenekler yaratıp ve bunlar seçildiğinde çalışmasını istediğimiz fonksiyonların ismini verip (ICG_AppendMenuItem()) sonra da CreatePopupMenu() adlı Win32 fonksiyonları ile yaratılan alt menülere bağlıyoruz. Yani aşağıdaki gibi bir ağaç yapısı hazırlıyoruz:



Son olarak ICG_SetMenu() fonksiyonu ile hazırlamış olduğumuz AnaMenu'yu uygulamamızın menüsü olarak seçiyoruz. Bundan sonra yapmamız gereken tek şey DosyaSec() ve Info() fonksiyonlarını yazmak:

```
#include"icb_gui.h"

int FRM;

void ICGUI_Create()
{
  ICG_MWSize(420, 500);
  ICG_MWTitle("MENÜ ÇUBUĞU");
}

void FileFunc()
{
  ICBYTES dosyol, resim;
  ReadImage(OpenFileMenu(dosyol, "JPEG\0*.JPG\0"), resim);
  DisplayImage(FRM, resim);
}

void Info()
{
  MessageBox(ICG_GetMainWindow(), "YARDIM FELAN YOK!\nBAŞININ\\
ÇARESİNE BAK.", "YARDIM", MB_OK);
}
```

FileFunc() bölüm 2.6'da gösterildiği şekilde dosya seçme penceresi açıp, seçilen resmi "resim" adlı bir matrise yüklüyor sonra da bu resmi çerçeve içerisinde gösteriyor. Info() ise Win32 API'nin fonksiyonlarından biri olan MessageBox'ı çağırarak bir yarım mesajı kutucuğu yaratıyor. Bu fonksiyonun detaylarını öğrenmek istiyorsanız Visual C++ dokümantasyonuna başvurabilirsiniz.

3. Matris Kullanımı

ICBYTES akışkanının dönüşebildiği en önemli şablonlardan biri matris yapısıdır. Matrisler C/C++ dilinde var olan dizin (array) yapıları ile hemen hemen aynı özelliklere sahip olmanın yanında içlerinde resim, ses ve 3-boyutlu modeller gibi birçok veriyi tutabilirler. ICBYTES kütüphanesi öğrenci sürümünde 2 değişik matrise izin verilmektedir. Bunlar "sade matris" ve "resim matrisidir." Sade matris ile resim matrisi arasında kullanıcı açısından hiçbir fark yoktur. Her ikisi de aynı işlemlere tabi tutulabilir. Ancak bu iki matrisin yaratılış şekilleri farklıdır. Örneğin Sade matris aşağıdaki yöntemlerle yaratılırken,

- ```
  ICBYTES A = {{1,2,3},{4,5,6},{7,8,9}};
  ```
  (Bu tür tanımlama için sadece double veya int cinsine izin vardır.)

- ```
  ICBYTES A;
  CreateMatrix(A, 250, 250, 3, ICB_UCHAR);
  ```

- ```
  ICBYTES A;
  A=5.4;
  ```

- ```
  ICBYTES A;
  A="buraya yazabilirsin";
  ```

Resim matrisi sadece tek yöntemle yaratılabilir:

- ```
  ICBYTES A;
  CreateImage(A, 250, 250, 3, ICB_UCHAR);
  ```

Kullanıcı açısından bu iki tür arasındaki tek fark ekranda gösterilirken resim matrisinin daha hızlı gösterilebilir olmasıdır. Ancak resim matrisi bu hızın bedeli olarak daha fazla hafıza harcar. Eğer matris sadece hesaplama yapmak için kullanılacaksa "sade matris" türü seçilmelidir. Ancak matris defalarca ekranda resim olarak basılacak ise, mesela bir video kaynağından gelen görüntüler veya bir animasyon gibi saniyede 20-30 kere ekrana sürekli bastırılacaksa "resim matrisi" seçilmelidir.
  ```

3.1. Matris Metotları

Matris metotları matrise bağlı gelen fonksiyonlardır. Bu fonksiyonlar matrisi taşıyan akışkan isminin arkasına nokta koyularak çağrılır. Örneğin:

```
ICBYTES A = {{1,2,3},{4,5,6}}; →
```
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Şeklinde bir matrisi tanımladıktan sonra,

`A.X()` bize A matrisinin X boyutunun uzunluğunu verir ki bu örnekteki uzunluk 3'tür.

`A.Y()` bize A matrisinin Y boyutunun uzunluğunu verir ki bu örnekteki uzunluk 2'dir.

`A.Z()` bize A matrisinin Z (3.) boyutunun uzunluğunu verir ki bu örnekteki uzunluk 1'dir.

`A.W()` bize A matrisinin W (4.) boyutunun uzunluğunu verir ki bu örnekteki uzunluk 1'dir.

3.2. Matris İşlemleri

3.2.1. Atama

En önemli matris işlemi atama işlemidir. Matrisin tanımlı olup olmamasına göre atama işleminin sonucu farklı olur. Örneğin:

```
ICBYTES A;
A=5;
```

İfadesinden sonra A akışkanının içerisinde tamsayı (integer) tek bir elemandan oluşan bir matris yaratılır. Ancak,

```
ICBYTES A;
CreateMatrix(A, 10, ICB_INT);
```

ifadesi, X boyutu 10, Y boyutu 1 olan yani 10 elemanlı bir vektör (veya dizin) yaratır. Bundan sonra

```
A=5;
```

Eşitlemesi bu vektörün içerisindeki 10 elemanın tamamını 5'e eşitler. Diğer yandan,

```
A=7.89;
```

ifadesini kullanırsak C/C++ dilinin davranışını taklit ederek vektörün tüm elemanlarını 7.89'un tamsayı kısmı olan 7'ye eşitler.

Eşitliğin her iki yanının da ICBYTES akışkanı olması durumunda kopyalanan akışkan tamamıyla aynı şekilde kopyalanana aktarılır. Yani,

```
B=A;
```

durumunda B'nin ne tür akışkan olduğuna bakılmaksızın A'nın aynısı haline gelir.

3.2.2. Matris Değişken Türleri

Matris değişken türleri ve elemanlarına ulaşmak için kullanılan fonksiyonlar aşağıdaki tabloda gösterilmiştir.

ICBYTES Kod Adı	Visual C++ Karşılığı	Bit Uzunluğu	Erişim
ICB_CHAR	char	8	A.C()
ICB_UCHAR	unsigned char (Byte)	8	A.B()
ICB_SHORT	short	16	A.S()
ICB_USHORT	unsigned short	16	A.u()
ICB_INT	int (long, int32_t)	32	A.I()
ICB_UINT	unsigned int (uns. long)	32	A.U()
ICB_LONG	long long (int64_t)	64	A.L()
ICB_ULONG	unsigned long long (uint64_t)	64	A.O()
ICB_FLOAT	float	32	A.F()
ICB_DOUBLE	double	64	A.D()

3.2.3. Toplama ve Çıkartma

Bir matrisi herhangi bir sayıyla veya başka bir matrisle toplamak veya çıkartmak için ICBYTES öğrenci versiyonunda sadece "+=" ve "-=" operatörleri tanımlıdır. Bu operatörlerin iki tarafındaki matris ya da sayı tipleri aynı olmadığında C/C++ dili kurallarına göre işlem yapılır. Yani iki matris toplanacaksa,

```
A+=B;
```

yazılır. B matrisi A ile toplanıp A'nın içine yazılır. A ve B matrisleri aynı boyutlara sahip olmalıdır. Yani A matrisi 5x8 ise B matrisi de 5x8 olmak zorundadır. Ancak türleri aynı olmak zorunda değildir. Mesela A matrisi ICB_INT ve B matrisi ICB_FLOAT ise, aynı C dilinde olduğu gibi B'deki elemanların tamsayı kısımları A'nın elemanlarına eklenir. A'nın tipi değişmez. Bir matris bir sayı ile toplandığında veya çıkartıldığında bu işlem matrisin tüm elemanlarına uygulanır. Yani,

```
A-=7;
```

İfadesi, A'nın tüm elemanlarından 7 sayısının çıkartılmasına sebep olur.

3.2.4. Çarpma ve Bölme

ICBYTES'ın öğrenci versiyonunda çarpma ve bölme *= ve /= operatörleri ile ve sadece temel değişken tipleri (char, int, float … vs) için tanımlıdır.

3.2.5. Karşılaştırma

İki matrisin eşit olup olmadığını "==" sembolü ile kontrol edebilirsiniz. Yani,

`if(A==B){…}` biçiminde ifadeler kurabilirsiniz.

3.2.6. Matris Tersi ve Determinant

Matris tersi için "inv()" fonksiyonu kullanılır:

```
bool inv(i, o)
```

tersi alınacak olan matris "i," sonuç ise "o" matrisinde yer alır. Matrisin tersi varsa bu fonksiyon `true` döndürür. Matrisin tersi yoksa `false` döndürür.

Determinant için ise "`determinant(i)`" fonksiyonu kullanılır. Bu fonksiyon "double" cinsi bir sayı döndürür.

3.2.7. Transpoze

A.t(B) → komutu B matrisinin transpozesini A matrisi içerisine yerleştirir. A ve B matrisleri aynı tiptir.

3.2.8. Dot product

C.dot(A,B) → komutu A.B çarpımını C matrisine yerleştirir.

3.2.9. Cross Product

C.cross(A,B) → komutu A×B çarpımını C matrisine yerleştirir.

3.3. Diğer Matris Fonksiyonları

Aşağıdaki tabloda matris üzerinde matematiksel olmayan işlemler yapan fonksiyonların listesi verilmiştir.

Fonksiyon ismi	Fonksiyon Açıklaması
`bool Convert2ImageMatrix(i);`	Sade matrisi resim matrisi haline getirir.
`bool ConvertType(m,type);`	Matris içeriğini koruyarak değişken tipini değiştirir. (mesela int → float)
`bool IsFloating(i)`	i matrisinin kayan nokta mı yoksa tam sayı mı olduğunu söyler.
`bool IsMatrix(i)`	i'nin matris taşıyıp taşımadığını söyler.
`bool AreDimsEqual(i, j)`	i ve j matrislerinin boyutlarının aynı olup olmadığını söyler.
`bool AreEqualImage(i, j)`	i ve j resimleri aynı boy ve bit derinliğinde ise true döndürür.
`bool FlipY(kaynak, hedef)`	Kaynak matrisini dikey yönde ters çevirip hedef matrisine yazar.

`void Free(M)`	M matrisini siler ve harcadığı hafıza alanını boşaltır.
`double Sum(M)`	M matrisinin tüm elemanlarının toplamını döndürür.
`void SumX(A,B)`	A matrisinin her satırındaki elemanların toplamını B vektörü olarak döndürür.
`void SumY(A,B)`	A matrisinin her sütunundaki elemanların toplamını B vektörü olarak döndürür.
`double Max(M)`	M matrisinin tüm elemanlarının en büyüğünü döndürür.
`void MaxX(A,B)`	A matrisinin her satırındaki elemanların en büyüğünü B vektörü olarak döndürür.
`void MinX(A,B)`	A matrisinin her satırındaki elemanların en küçüğünü B vektörü olarak döndürür.
`double Min(M)`	M matrisinin tüm elemanlarının en küçüğünü döndürür.
`void MaxY(A,B)`	A matrisinin her sütunundaki elemanların en büyüğünü B vektörü olarak döndürür.
`void MinY(A,B)`	A matrisinin her sütunundaki elemanların en küçüğünü B vektörü olarak döndürür.

Listesi verilmiş olan bu fonksiyonları kullanırken, kullanıcının sadece akışkanı yaratmalıdır; hedef matrisini yaratma zorunluluğu yoktur. Eğer hedef matrisi yeterli büyüklükte değilse fonksiyon “hedef” matrisini kendine yetecek büyüklüğe getirir.

3.4. Özel Matrisler

En çok ihtiyaç duyulan matrislerin başında birim (identity) matris gelmektedir. Sol üst köşeden sağ alt köşeye çaprazlama elemanları 1 olan ve diğer tüm elemanları 0 olan kare matris yaratmak için:

```
IdentityMatrix(matris, boy, tip)
```

Komutu kullanabilirsiniz. Sıfırdan başlayarak sağa ve aşağıya doğru birer birer artna matris yaratmak için ise:

```
IncreasingMatrix(matris, x,y,z, tip) kullanılabilir.
```

İstatistiksel uygulamalar için çok gerekli olan sıfır ile bir aralığında rastgele veri üretimi için ise:

Düz dağılım için → `RandomUniform(mat, x,y,z)`

Gauss dağılım için → `RandomNormal(mat, x,y,z)`

Son olarak, tüm satır ve sütunları toplamı eşit olan "sihirli" kare matrisler yaratmak için:

Magic(mat,3) → mat=$\begin{bmatrix} 2 & 9 & 4 \\ 7 & 5 & 3 \\ 6 & 1 & 8 \end{bmatrix}$

3.5. Arıza Teşhisi

ICBYTES kütüphanesinin en önemli özelliklerinden biri derlerken yakalayamayacağınız hataları, çalışma anında (run time) size iletip hatanın nerden kaynaklandığını bulmanızı sağlayacak son derece zeki mekanizmalara sahip olmasıdır.

C++ dili çok güçlü bir dildir. Hatta bilgisayar dilleri arasında makine dilinden sonra en güçlü dildir. Güçlü bir dil programcıya yapmak istediği her şeyi yapmaya izin veren bir dil demektir. C++ güçlü olmanın yanında, yine makine dilinden sonra en hızlı dildir. Hızdan kastımız ürettiği makine kodunun, yani .EXE dosyalarının çalışma hızıdır. Tüm bunlar nedeniyle C/C++ dili yaratıldığı 1972 yılından beri dünyada en fazla kullanılan ilk beş dil arasında sürekli yerini korumuştur. Bu popülerliği nedeniyle Java ve C# dilleri bile C++ dili üzerine kurulmuşlardır. Farklı yıllarda bu üç dil öne veya arkaya düşse bile bir tanesi mutlaka ilk üçte yerini koruduğu için, C++ dil yapısının en çok kullanılan dil yapısı olduğunu söylemek mübalağa olmaz.

Bu popülerliğine rağmen C/C++ dili öğrenmesi ve hatta öğrendikten sonra bile kullanması zor bir dildir. Bunun temel nedeni ürettiği kodu en hızlısı yapmak için tasarlanmış olması ve kullanıcıya verdiği özgürlüktür. C++ dilinde eğer işletim sistemi izin veriyorsa istediğiniz herhangi bir hafıza alanına ulaşabilirsiniz. Hatta sizin programınızın hafıza alanından çıkıp başka bir programın hafıza alanına bile ulaşabilirsiniz. DOS ve Windows 98 işletim sistemleri bunu mümkün kılıyordu. Windows 2000'den itibaren bu izinler kaldırıldı. C/C++ derleyicisi sizin kullandığınız işaretçilerin (pointer) ve dizi (array) indekslerinin doğru olup olmadığını kontrol etmez. Yani diyelim ki şöyle bir kod yazdınız:

```
int i[10];
i[-1]=5;
i[23]=7;
```

Bu kodda 10 elemanı olan bir dizin yaratıp sonra da bunun -1'inci ve 23'üncü elemanlarına sayı yazıyorsunuz. C/C++ dilinde, 10 elemanı olan bir dizinde geçerli indeksler 0 ile 9 arasındadır. Ama C/C++ derleyicisi bu kodu hiç itiraz etmeden derler. Sonra ne olur? Programı çalıştırdığınızda program çöker. Yani program orta yerinde sonlanır. Neden böyle olduğunu işletim sisteminiz bir uyarı mesajı ile

söyleyebilir de söylemeyebilir de. Söylese bile bu hata mesajı sizin bu problemi düzeltmenize yarayacak bir bilgiyi çoğu zaman içermez.

Diyelim ki siz bir firmasınız ve bir müşterinize stok programı yazdınız. Kullanıcınızın 100 tane rafı var ve her rafta kaç ürün bulunduğunu hafızada tutmak için 100 elemanı olan bir dizin kullandınız. Kullanıcı raftan bir ürün aldığında raf numarasını programa giriyor, program da o raftaki ürün sayısını bir azaltıyor. Kullanıcının girdiği raf numarası "raf_no" diye bir değişkende duruyorsa, basitçe bu program kodu aşağıdaki gibi olsun:

```
int raf[100];
raf[raf_no]--;
```

Çoğu zaman programınız gayet güzel çalışır. Ancak kullanıcı 11. Raf yazmaya çalışırken 1'e fazladan basıp 111 yazarsa programınız çöker. Ya da en iyi ihtimalle programınızdaki başka bir değişkenin değerini değiştirir. C/C++ dışındaki birçok dil mümkünse bunu derleme anında, ve hatta çalışırken kontrol eder ve uyarı mesajı verir. Ancak C/C++ bunu yapmaz. Nedeni mümkün olan en hızlı .EXE kodunu yaratmaktır. Bir dil bu tür kontrolleri yapabilmek sizin dizinlerle ilgi yazdığınız her satır için sınır kontrolü kodu çalıştırmak zorunda kalır. Yani başka bir dilde "`raf[raf_no]--;`" yazdığınızda derleyici oraya şöyle bir kod parçası ekler:

```
if(raf_no >=0 || raf_no <=99)
  raf[raf_no]--;
else Hata mesajı … karşı önlemler vs.
```

Üç mega piksellik bir resim işlerken her pikseli bir kere okuma bir kere yazma yaptığınızı düşünsek, bu if-else yapısının 6.000.000 kere çalıştırılması gerekir. Resim işlemek için harcamanız gereken zamanın 3 ila 5 katını harcarsınız. İşte bu nedenle gerçek zamanlı resim işleme uygulamaları gerektiren video işleme veya bilgisayarla görü gibi projelerde hiçbir firma C/C++ dışında bir dil kullanmaz. Yani burada aklı başında firmalardan bahsediyoruz. İki sene sonra topu atan firmalardan değil.

Burada mesele sadece zaman da değildir. Farz edin ki hareket eden bir robot yaptınız. Bu robot bir kamera ile etrafındaki cisimleri görüp onlara çarpmaktan kaçınması gerekiyor. Yukarıdaki işlemleri yapan bir dil kullandıysanız daha fazla işlem yaptığınız için daha fazla enerji de harcayacaksınız demektir. Bu da demek oluyor ki daha büyük piller kullanacaksınız robotunuz üzerinde. Daha büyük pilleri taşımak için daha büyük ve pahalı motorlar kullanacaksınız. Maliyetleriniz böyle artıp gidecek. Size rakip olan firma sizin gibi aptallık etmeyip kodunu C++ dilinde yazdıysa, maliyeti sizin yarısına düşürüp iki sene sonra sizi iflas ettirir. Bu yüzden (kaliteli) üniversitelerin mühendislik fakültelerinde yapılan mühendisliğin kalitesi kadar maliyetini düşürmenin yolları da öğretilir. En azından biz öğretmeye

çalışıyoruz. Bu arada eklemeden geçmemeliyiz, bu bir robot değil de uçan bir dron ise maliyetlerin kaç katına çıkacağını siz hesaplamaya çalışın.

3.4.1. Arıza Teşhisi Nasıl Başlatılır?

Farz edelim ki elimizde aşağıdaki gibi bir buton fonksiyonu olsun:

```
void butonfonk()
{
  ICBYTES a(1);
  CreateMatrix(a, 5, 5, ICB_INT);
  a.I(0, 0) = 5;
  a.C(2, 3) = 4;
}
```

Bu fonksiyon içerisinde "a" isimli bir akışkan yaratılıyor. Ancak yaratılma esnasında daha evvel görmediğimiz bir şey yapılıyor. Yanına parantez açılıp içerisine de "bir" (1) yazılmış. Bir akışkan bu şekilde yaratıldığında bunun manası "a" adlı akışkana bir numara verdiğimiz manasına geliyor. Bunun sebebini birazdan öğreneceğiz.

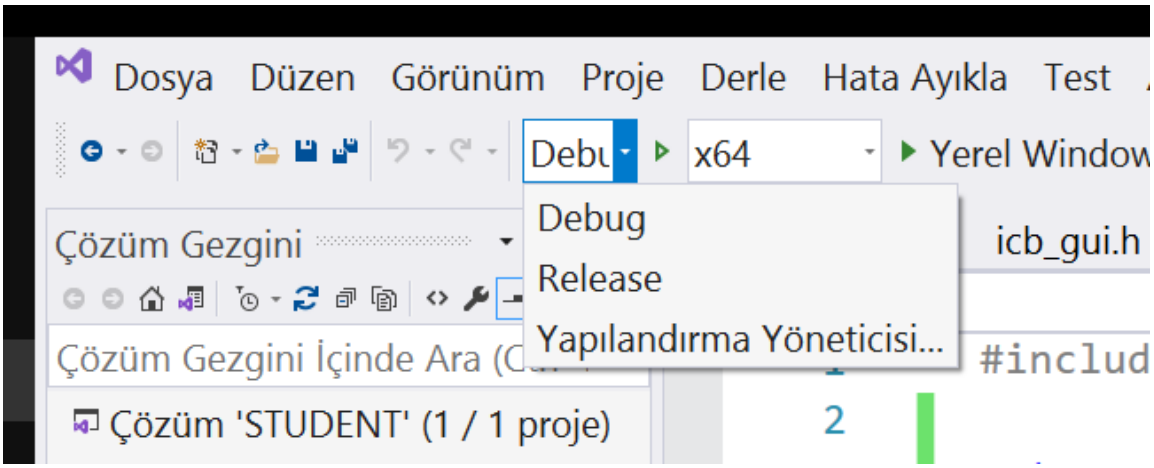
Daha sonra "a" akışkanı 5x5'lik tam sayı bir matris haline getiriliyor. Sonra da "a" matrisinin (0,0)'ıncı elemanı içerisine 5 yazılıyor. Ancak burada bir problem var. Matris indeksleri 1'den başlar. Dolayısı ile sıfırıncı elemana ulaşmaya kalkarsak bir dizinin -1. elemanına ulaşmaya çalışmak gibi olur. Sonraki satıra geldiğimizde ise tam sayı matris olan a'ya sanki char cinsindenmiş gibi ulaşılmaya çalışıldığını görüyoruz. Bu da en iyi ihtimalle yanlış elemana ulaşacaktır. Bu kodu çalıştırdığınızda program çökebilir. İşin kötü tarafı programın çökmesi iyi ihtimaldir. Çünkü program çökerse en azından programınızda hata olduğunu bilirsiniz. Program çökmezse yanlış hesaplamalar yapacak demektir. Düşünün ki bu bir muhasebe programı olsa muhasebe hesaplarınız yanlış çıkacak demektir. Bu bir tomografi cihazının görüntü işleme fonksiyonu olsa kansersiz hastada kanser olduğunu gösteren tomografi görüntüleri ortaya çıkaracaktır. Her iki durum da programcıyı sorumlu duruma düşürecektir. Neyse ki ICBYTES kütüphanesi bu tür hataları ortaya çıkaran son derece zeki hata teşhis ve ayıklama mekanizmalarına sahiptir. Bu mekanizmaları harekete geçirmek için programımızın baş tarafında ufak bazı değişiklikler yapmamız gerekmektedir. Öncelikle en başa hata ayıklama mesajlarını yazan nesneyi tanımlamalıyız:

```
#include"icb_gui.h"
extern ICBDIAG diag;
```

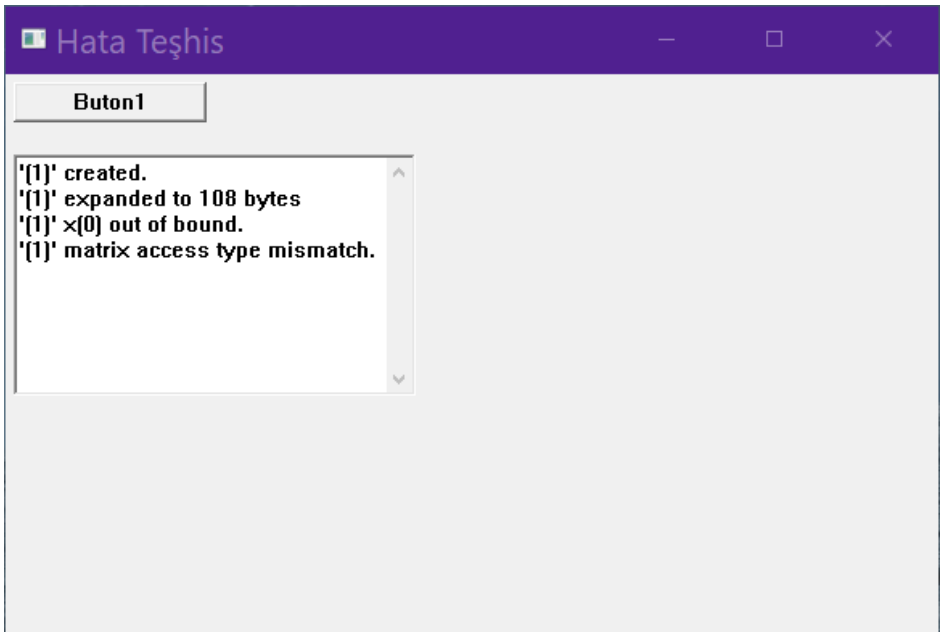
Programımızın daha ikinci satırında, kütüphane içerisinde yaratılmış olan "diag" isimli (diagnostic) bir nesneyi kullanmak istediğimizi derleyiciye söylüyoruz. Şimdi bu nesnenin mesajlarını yazabilmesi için çok satırlı bir metin kutucuğuna ihtiyacı var ve bunu da ICGUI_main() fonksiyonunda sağlıyoruz:

```
void ICGUI_main()
{
  ICG_Button(5, 5, 120, 25, "Buton1", butonfonk);
  MLE1 = ICG_MLEditSunken(5, 50, 250, 150, "", SCROLLBAR_V);
  diag.SetOutput(ICG_GetHWND(MLE1));
}
```

Son olarak programımızı “DEBUG” modunda derlememiz gerekiyor. Bunu yapmak için Visual Studio’nun menüsünden aşağıda görüldüğü gibi DEBUG kipini seçiyoruz.



Tüm bunları yaptıktan sonra programı derleyip çalıştırdığımızda, Buton’a basınca aşağıdaki ekran görüntüsü ortaya çıkacaktır.



Bu program, ICBYTES değişkeninin geçirdiği her önemli aşama ve hatalı kullanımları hakkında bizi bilgilendirir hale gelmiştir. İlk satırda:

'(1)' created.

Diyerek numarası "1" olan ICBYTES akışkanının yaratıldığını söylemektedir. Hatırlayınız, "a" akışkanı yaratılırken ona bir numara vermiştik:

```
ICBYTES a(1);
```

İşte bu numara, şu anda arıza teşhis sistemi tarafından bize hangi akışkan hakkında bili sağlandığını ifade etmek için kullanılmaktadır. Şimdi bu mesajların ikincisine bakalım:

(1)' expanded to 108 bytes

Akışkanımızın kullandığı hafıza miktarının 108 bayta genişlediği iletiliyor. Bunun sebebi "a" akışkanı içerisine 5x5'lik tam sayı bir matris yaratmamızdır. Tam sayılar 4 bayt kullanır. Dolayısı ile 25 tane tam sayı 100 bayt kullanacaktır. Fazladan 8 baytı da akışkanımız kendisi kullanıyor olmalı. Bundan sonraki satırda ise:

'(1)' x(0) out of bound.

Matrisimizin X boyutunun sıfırıncı elemanına ulaşmaya çalıştığımızı ve bunun hatalı olduğunu söylüyor. Hatırlarsanız buton fonksiyonunun üçüncü satırında "`a.I(0, 0) = 5;`" yazmıştık. Bunun hatalı erişim olduğunu bize bildiriyor. Buton fonksiyonunun son satırında ise "`a.C(2, 3) = 4;`" yazıp tam sayı bir matrise karakter cinsindenmiş gibi ulaşmaya çalışmıştık. Bu konuda da bizi uyarıyor:

'(1)' matrix access type mismatch.

İşte ICBYES kütüphanesini diğer kütüphanelerden farklı kılan, kullanım kolaylığı ve güvenilirlik sağlayan hata teşhis ve ayıklama özelliği bu şekilde çalışmaktadır.

3.6. Matris Sorgulama

ICBYTES akışkanının taşıdığı matrislerin birçok özelliği bulunmaktadır. Programcıların bu matrislerle çalışırken bu özelliklere ulaşmaları gerekecektir. Bunu kolaylaştırmak için kütüphane içerisine birçok fonksiyon eklenmiştir.

3.5.1. Boyut Sorgulama

Matris Boyutlarını sorgulamak için üç farklı metot bulunmaktadır. ICBYTES kütüphanesinde matrisler en fazla dört boyutlu olabildiği için bunlardan ilk ikisi matrisin boyutlarını (X,Y,Z,W) bir dizin (array) veya başka bir matris olarak döndürür:

```
long long* SizeinArray(ICBYTES& i);
ICBYTES & size(ICBYTES& i);
```

Bunlardan birincisi bu dört boyutun büyüklüğünü long long[4] bir dizin içerisinde döndürür. İkincisi ise ICBYTES akışkanı içinde yine long long cinsi 4 elemanlı bir vektör döndürür. Bu fonksiyonların kullanımını aşağıdaki gibi bir program yazarak daha iyi anlayabiliriz.

```
void MatrisYaz()
{
  ICBYTES A = { {1,2,3},{4,5,6},{7,8,9} };
  DisplayMatrix(MLE, A);
  ICBYTES B = size(A);
  DisplayMatrix(MLE, B);
  long long* t = SizeinArray(A);
  ICG_printf(MLE, "A matrisi boyutu:(%d , %d)", t[0], t[1]);
}
```

MLE burada çok satırlı bir metin kutucuğunun kulpudur. Bu fonksiyonun ekranda yazdıracağı çıktı aşağıdaki gibi olacaktır:

```
(INT 3 x 3)
1 2 3
4 5 6
7 8 9

(INT64 4 x 1)
3 3 1 1

A matrisi boyutu:(3 , 3)
```

Burada dikkat edilmesi gereken husus, 3x3 bir matris yarattığımızda diğer boyutların büyüklüğünün sıfır değil bir olduğudur.

Boyut sorgulamak için kullandığımız üçüncü fonksiyon ise GetMatrixDims() fonksiyonudur. Bu fonksiyon aşağıda tanımı yapılmış tamsayılardan birini döndürür:

```
ICB_HAS_X      1    (sadece X boyutu var)
ICB_HAS_Y      2
ICB_HAS_XY     3    (hem X hem Y boyutu var)
ICB_HAS_Z      4
ICB_HAS_XZ     5
ICB_HAS_YZ     6
ICB_HAS_XYZ    7    (hem X hem Y hem de Z boyutu var)
ICB_HAS_W      8
```

```
ICB_HAS_XW      9     (hem X hem W boyutu var)

...
ICB_HAS_XYZW    15    (hem X hem Y hem de Z boyutu var)
```

Bunlar dışında matris sorgulama yapan diğer metotlar EK-2'de verilmiştir.

3.7. Tipi Bilinmeyen Matrise Erişim

OpenCV'nin en kötü özelliklerinden birinin matris elemanlarına erişimindeki zorluklar olduğundan bahsetmiştik. Maalesef kötü dökümantasyonu sebebiyle bu problem katmerlenerek büyümektedir. Bir fonksiyonun size döndürdüğü matrisin tipini bilmiyorsanız o matrisi değiştirmek veya değerleri ekrana yazdırmak istediğinizde önce o matrisi double cinsinden bir matrise dönüştürmeniz önerilmektedir. Bu ise programcı için zorluğa, program için ise yavaşlamaya neden olmaktadır. Tipi bilinmeyen bir matrise yazmak ise imkânsızdır. Oysa ICBYTES kütüphanesinde tipini bilmediğiniz bir matrisin elemanlarını okumak çok kolaydır. Tek yapmanız gereken matrisin yanına parantez açıp ulaşmak istediğiniz elemanın indeksini gitmektir. Yani diyelim ki tipini bilmediğiniz A diye bir matrisiniz var ve (15,4) elemanını bir değişkene atmak ya da yazdırmak istiyorsunuz:

```
double t=A(15,4);
```

yazmanız yeterlidir. Burada dikkat etmeniz gereken iki husus bulunmaktadır:

- Bu şekilde sadece matrisin elemanını okuyabilirsiniz. **YAZAMAZSINIZ**.
- Matrisin tipi ne olursa olsun, döndürülen değer hep **double** cinsindendir.

Matrisin tipi int de olsa char da olsa size double cinsinden değer ulaşacağı için, bu tip bir değişken işinize yaramıyorsa tip kaydırma yapabilirsiniz:

```
int t=(int)A(15,4);
```

Eğer matris elemanına yazmak istiyorsanız o takdirde Set() fonksiyonunu kullanmalısınız. Mesela diyelim ki A'nın (15,4) elemanını -1.5 yapmak istiyoruz:

```
Set(15,4,A,-1.5);
```

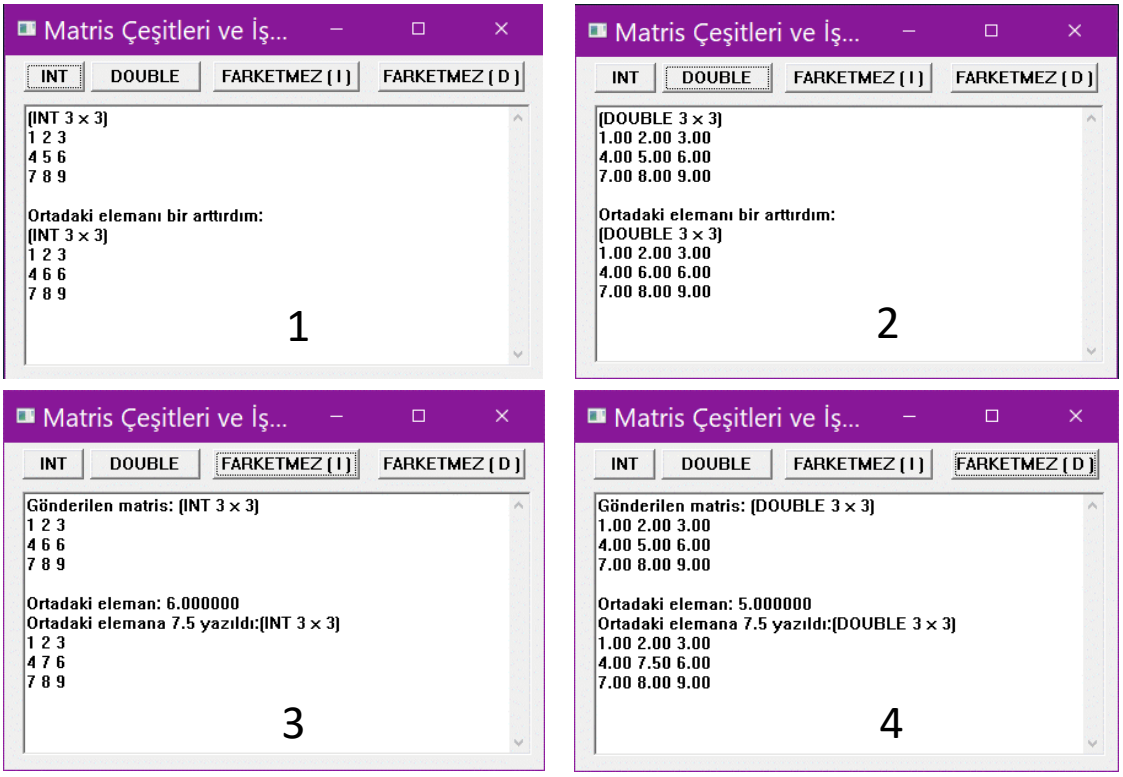
Komutu bu işlemi gerçekleştirir. Set() fonksiyonunun son girdisi yazmak istediğimiz değerdir. Bu değer herhangi bir tip olabilir. Yani -1.5 yerine tam sayı da (mesela 5) yazabiliriz. Son girdi olarak C++'da tanımlı tüm tipler (char, short, float…) kullanılabilir. Burada dikkat etmemiz gereken hususlar ise şunlardır:

- Eğer A matrisinin tipi tam sayı ise ve ona double/float yazarsanız yazdığınız sayının sadece tam sayı kısmı yazılır.
- Matrisin tipi unsigned ise veya yazdığınız değeri taşıyabilecek büyüklükte değilse C++ değişken atama kuralları doğrultusunda sayınız matrisin

elemanına yazılır. Yani A matrisi tipi unsigned ise, yukarıdaki Set() komutu A'nın (15,4) elemanına 4.294.967.295 yazar. Bu sebeple dikkatli olunmalıdır.

Aşağıda bu özellikleri kullanan örnek bir programın ekran görüntüleri verilmiştir. Bu programın düğmelerine sırayla bastığımızda şunlar olmaktadır:

1. Main içerisinde yaratılmış olan:
   `static ICBYTES I = { {1,2,3},{4,5,6},{7,8,9} };`
   (int) matris ekrana yazdırılmakta, sonra ortasındaki elemana I.I(2,2) metoduyla erişilip bir arttırılmaktadır.
2. Main içerisinde yaratılmış olan:
   `static ICBYTES D = { {1.0,2.0,3.0},{4.0,5.0,6.0},{7.0,8.0,9.0} };`
   (double) matris ekrana yazdırılmakta, sonra ortasındaki elemana D.D(2,2) metoduyla erişilip bir arttırılmaktadır.
3. Main içerisinde yaratılmış olan:
   `static ICBYTES I = { {1,2,3},{4,5,6},{7,8,9} };`
   farketmez() fonksiyonuna yollanmakta ve ortasındaki elemana U(2,2) metoduyla ulaşılıp değer aslında tam sayı 6 (1. Adımda bir arttırıldığı için) olmasına rağmen ekrana 6.000000 diye yazdırılmakta; sonra bu değer yerine Set(2,2,U,7.5) metoduyla 7,5 yazılmaya çalışılmakta ancak matris ekrana bastırıldığında ortadaki elemanın 7 olduğu görülmektedir.



4. Main içerisinde yaratılmış olan:
   `static ICBYTES D = { {1.0,2.0,3.0},{4.0,5.0,6.0},{7.0,8.0,9.0} };`

farketmez() fonksiyonuna yollanmakta ve ortasındaki elemana U(2,2) Set(2,2,U,7.5) metoduyla 7,5 başarıyla yazılmaktadır. Aşağıda bu programın kaynak kodu verilmiştir:metoduyla ulaşılıp değer aslında (double) 6.00 (2. Adımda bir arttırıldığı için) olmasına rağmen ekrana 6.000000 diye yazdırılmakta; sonra bu değer yerine

```
#include"icb_gui.h"

int MLE;

void ICGUI_Create()
{
  ICG_MWTitle("Matris Çeşitleri ve İşlemleri");
  ICG_MWSize(460, 320);
}

void Int_isle(void *icb)
{
  ICBYTES* pi = (ICBYTES*)icb;
  DisplayMatrix(MLE, *pi);
  ICG_printf(MLE, "Ortadaki elemanı bir arttırdım:\n");
  pi->I(2, 2)++;
  DisplayMatrix(MLE, *pi);
}

void Double_isle(void* icb)
{
  ICBYTES* pi = (ICBYTES*)icb;
  DisplayMatrix(MLE, *pi);
  ICG_printf(MLE, "Ortadaki elemanı bir arttırdım:\n");
  pi->D(2, 2)++;
  DisplayMatrix(MLE, *pi);
}

void Farketmez(void* icb)
{
  ICBYTES& U = *(ICBYTES*)icb;
  ICG_printf(MLE, "Gönderilen matris: ");
  DisplayMatrix(MLE, U);
  ICG_printf(MLE, "Ortadaki eleman: %f\n", U(2,2));
  Set(2, 2, U, 7.5);
  ICG_printf(MLE, "Ortadaki elemana 7.5 yazıldı:");
  DisplayMatrix(MLE, U);
}
```

```
void ICGUI_main()
{
  //hepsi integer olmalı
  static ICBYTES I = { {1,2,3},{4,5,6},{7,8,9} };
  ICG_Button(15, 5, 50, 25, "INT", Int_isle,(void*)&I);
  //hepsi double olmalı
  static ICBYTES D = { {1.0,2.0,3.0},{4.0,5.0,6.0},{7.0,8.0,9.0} };
  ICG_Button(70, 5, 90, 25, "DOUBLE", Double_isle,(void*)&D);
  ICG_Button(170,5,120, 25, "FARKETMEZ ( I )",Farketmez,(void*)&I);
  ICG_Button(305,5,120,25,"FARKETMEZ ( D )",Farketmez,(void*)&D);
  MLE = ICG_MLEditSunken(15, 40, 415, 215, "", SCROLLBAR_V);
}
```

Son olarak, tipi bilinmeyen bir matrise, istediğiniz bir tipe dönüştürerek de erişebilirsiniz. Mesela tam sayıya dönüştürmek için:

```
ConvertType(M, ICB_INT);
```

3.8. Karakter Erişimi

Tipi ne olursa olsun, bir matrisin hafıza alanına bayt olarak erişmek istiyorsanız tek yapmanız gereken köşeli parantez kullanmaktır:

A[5]='C';

Komutu, A matrisinin 6. Baytına 'C' harfini yani 67 değerini yazar. Eğer matrisiniz zaten karakter cinsinden bir vektör ise (yani char veya unsigned char), bu şekilde bir karakter dizini olarak kullanabilirsiniz. Burada dikkat etmeniz gereken hususlar şunlardır:

- Matris değil, C++ indeks biçimi kullanılır. Yani, indeks değerleri sıfırdan başlar ve matrisin bayt cinsinden büyüklüğünün bir eksiğine kadar gider.
- PC uyumlu makinalarda y koordinatı matris koordinatlarına ters olduğu için eğer vektör değil de matris kullanıyorsanız en aşağıdaki satırdan erişmeye başlar. Vektörde tek satır olduğu için bu bir problem yaratmaz. Aynı C++ dizinleri (array) gibi kullanabilirsiniz.

4. Grafik Arabirim Stilleri

Önceki bölümlerde I-See-Bytes (ICBYTES) kütüphanesinin temel grafik özelliklerini tanıttık. Bu özelliklerle birçok uygulama yazılabilir olsa da, karmaşık işler yapabilen, profesyonel görünümlü grafik arabirimler yaratabilmek için çok daha farklı ve çok daha çeşitli kutucuklara ve stillere ihtiyacımız var. Bu bölümde I-See-Bytes kütüphanesinin grafik arabirim fonksiyonlarını içeren alt kütüphanesi olan I-See-GUI (ICGUI) daha derinlemesine inceleyeceğiz.

4.1. Resimli Buton

Bu örnekte içerisinde yazı yerine resim taşıyan buton kutucuğu nasıl yaratılır onu göreceğiz. Aşağıdaki kod örneğinde,

```
#include"icb_gui.h"

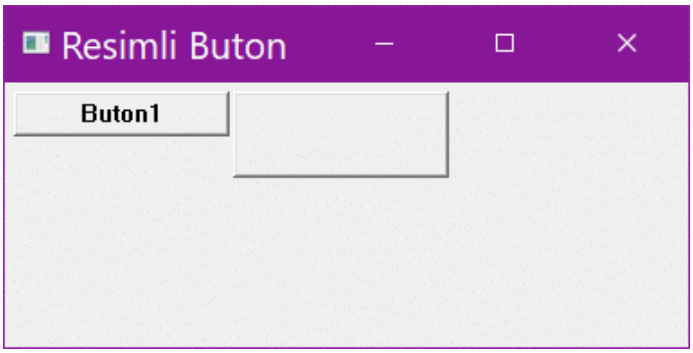
int BTN1;
ICBYTES resim;

void butonfonk1();
void butonfonk2(){}

void ICGUI_Create()
{
  ICG_MWSize(400, 200);
  ICG_MWTitle("Resimli Buton");
}

void ICGUI_main()
{
  ICG_Button(5, 5, 120, 25, "Buton1", butonfonk1);
  BTN1 = ICG_BitmapButton(127, 5, 120, 48, butonfonk2);
  ICB_ReadImage("Buton.bmp", resim);
}
```

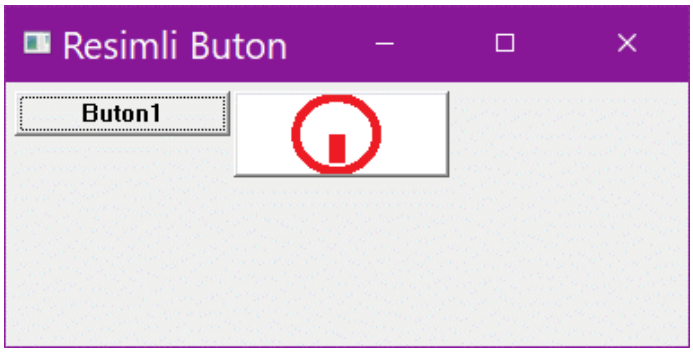
ICG_Main içerisisinde yeni tip bir Buton yaratma fonksiyonu görüyoruz. İsmi "ICG_BitmapButton()" olan bu fonksiyon 120x48 boyutlarında bir button yaratmaktadır. Diğer buton fonksiyonlarından farklı olarak bu fonksiyon karakter dizini almamaktadır. Sonuç olarak bu fonksiyonun yarattığı buton aşağıda gösterildiği üzere üzerinde yazı yazmayan bir butondur.



ICG_Main fonksiyonunda diğer dikkatimiz çeken bir komut olan "ReadImage()," "Button.bmp" diye, Windows'daki paint programı ile oluşturduğumuz bir dosyayı "resim" isimli akışkana matris olarak yüklemektedir. Burada bitmap dosyanın dosyolu verilmeden sadece ismi verilmiştir. Bu demek oluyor ki Button.bmp dosyası bu proje tarafından oluşturulan .EXE ile aynı klasörde yer almalıdır. Eğer proje Visual Studio üzerinden çalıştırılmakta ise Button.bmp dosyasının proje dosyası ile aynı klasörde olması gerekir.

Soldaki butona baktığımızda alışageldiğimiz bir buton olduğunu görüyor ve onun fonksiyonunu incelediğimizde aşağıda görüldüğü üzere Button.bmp dosyasını içerisine yüklediğimiz resim akışkanını kullanarak şu anda boş durmakta olan ikinci butonumuzun resmini sabitleyen "SetButtonBitmap()" fonksiyonunu çağırdığını görüyoruz. Buton1'e basıldığında uygulamamızın görüntüsü aşağıdaki hale dönüşüyor.

```
void butonfonk1()
{
  SetButtonBitmap(BTN1, resim);
}
```



Normal bir uygulamada, bir butona basmak yerine uygulama açıldığı anda resmin buton üzerine yerleştirilmesini istersiniz. Bunun olması için ICG_Main fonksiyonunu aşağıdaki gibi değiştirmeniz yeterli olacaktır.

```
void ICGUI_main()
{
  int BTN1;
  static ICBYTES resim;
  ICG_Button(5, 5, 120, 25, "Buton1", butonfonk1);
  BTN1 = ICG_BitmapButton(127, 5, 120, 48, butonfonk2);
  ICB_ReadImage("Buton.bmp", resim);
  SetButtonBitmap(BTN1, resim);
}
```

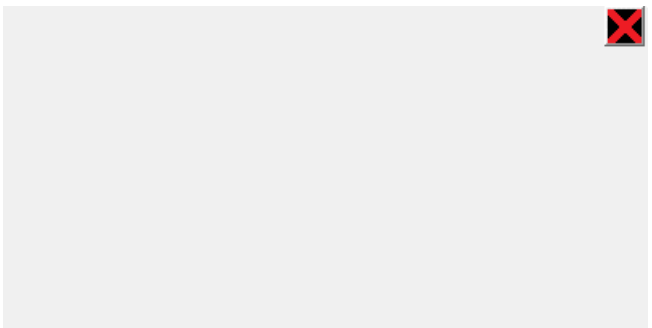
Bu tür resim butonlarını en çok kullanmayı isteyeceğimiz uygulamalar başlık çubuğu olmayan pencereleri olan uygulamalardır. Örneğin aşağıdaki kod, başlık çubuğunu ortadan kaldırıldığı için, programdan çıkmanızı sağlayan üzerinde kırmızı bir çarpı olan minik, siyah bir buton yaratır. Ekranın tamamını kullanmak istediğinizde bu tür pencereler tercih edebilirsiniz.

```
#include"icb_gui.h"

void butonfonk1(){exit(0);}

void ICGUI_Create()
{
  ICG_MW_RemoveTitleBar();
  ICG_MWSize(400, 200);
}

void ICGUI_main()
{
  int BTN1;
  static ICBYTES resim;
  BTN1 = ICG_BitmapButton(373, 0, 25, 25, butonfonk1);
  ICB_ReadImage("exit.bmp", resim);
  SetButtonBitmap(BTN1, resim);
}
```



4.2. Çok Satırlı Metin Kutucuğu Tipleri

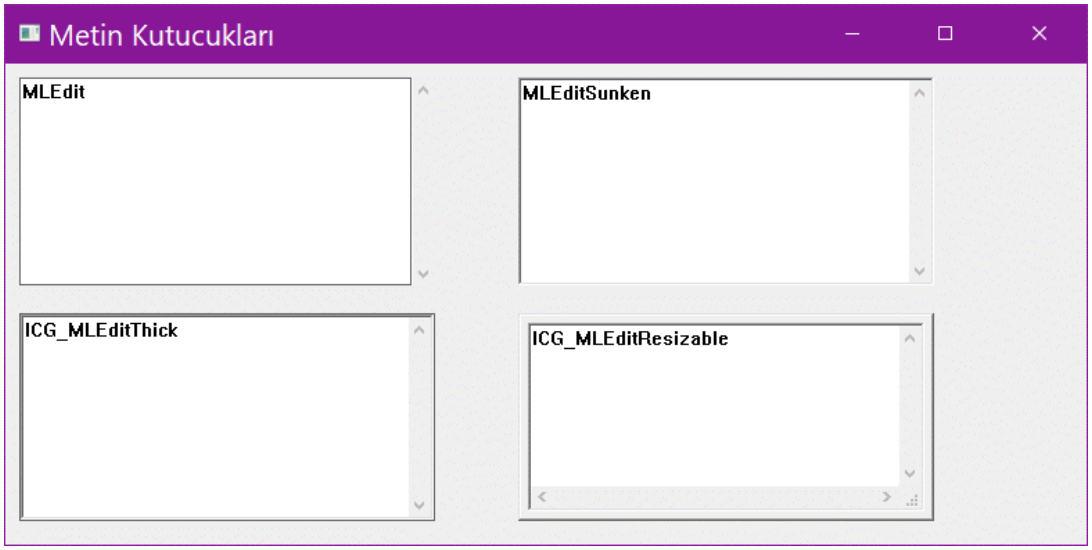
I-See-GUI öğrenci versiyonunda dört çeşit çok satırlı metin kutucuğu (Multi Line (ML) Edit Widget) bulunmaktadır. Aşağıdaki kod örneği bunların nasıl yaratılacağını göstermektedir.

```
#include"icb_gui.h"

void ICGUI_Create()
{
  ICG_MWSize(800, 400);
  ICG_MWTitle("Metin Kutucukları");
}

void ICGUI_main()
{
  ICG_MLEdit(10, 10, 300, 150,"MLEdit",1);
  ICG_MLEditSunken(370, 10, 300, 150, "MLEditSunken",
SCROLLBAR_V);
  ICG_MLEditThick(10, 180, 300, 150, "ICG_MLEditThick",
SCROLLBAR_V);
  ICG_MLEditResizable(370, 180, 300, 150, "ICG_MLEditResizable",
SCROLLBAR_HV);
}
```

Aşağıda bu kodun oluşturduğu uygulama penceresinin ekran görüntüsü verilmiştir. En sade versiyon olan MLEdit(), sol üst köşede gösterilmiştir. Genelde en fazla tercih edilen stil olduğu için MLEditSunken()' ı daha evvelki örneklerde de sıklıkla kullanmıştık. MLEditThick() biraz daha kalın bir çerçeveye sahiptir. MLEditResizable() ise fare imleci ile kenarlarından tutularak boyu değiştirilebilir bir stildir. Bu stili yaratırken "SCROLLBAR_HV" opsiyonu ile, yani hem yatay hem dikey kaydırma sütunu ile yaratmak faydalı olacaktır.



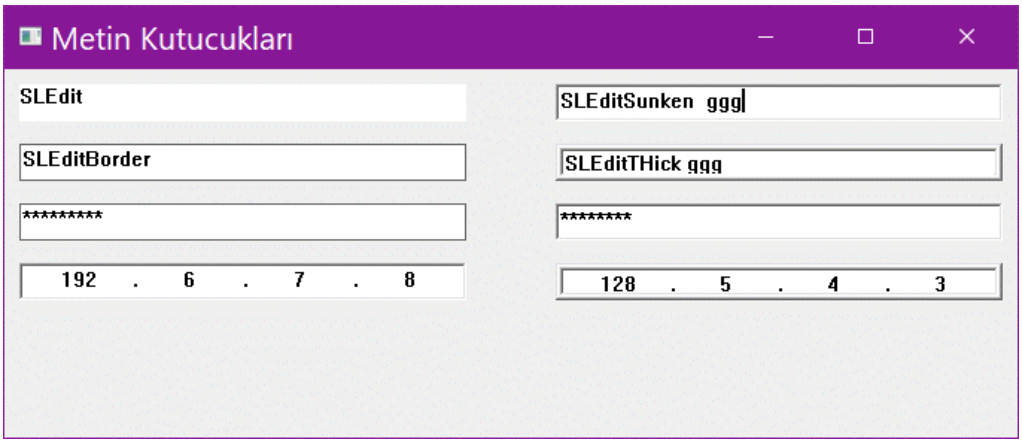
4.3. Tek Satırlı Metin Kutucuğu Tipleri

Aşağıdaki kod örneğinde ve uygulamanın ekran görüntüsünde görüldüğü üzere sekiz çeşit tek satırlı metin kutucuğu stili bulunmaktadır. Bunlardan ilk dördü düz metin kutucuğu iken sonraki iki tanesi parola kutucuğu, son ikisi ise internet adresi kutucuğudur.

```
#include"icb_gui.h"

void ICGUI_Create()
{
  ICG_MWSize(700, 300);
  ICG_MWTitle("Metin Kutucukları");
}

void ICGUI_main()
{
  ICG_SLEdit(10, 10, 300, 25,"SLEdit");
  ICG_SLEditSunken(370, 10, 300, 25, "SLEditSunken");
  ICG_SLEditBorder(10, 50, 300, 25, "SLEditBorder");
  ICG_SLEditThick(370, 50, 300, 25, "SLEditTHick");
  ICG_SLPasswordB(10, 90, 300, 25);
  ICG_SLPasswordSunken(370, 90, 300, 25);
  ICG_IPAddressSunken(10, 130, 300, 25, 0xc0060708);
  ICG_IPAddressThick(370, 130, 300, 25, 0x80050403);
}
```



Burada dikkat edilmesi gereken bir husus, kalın çerçeve tipi kullanıldığında çerçeve genişliğinin içeriye doğru büyümesinden dolayı “g” gibi bazı harflerin kuyruk kısımlarının gözükmüyor olmasıdır. Buradaki kutucukların hepsinin yüksekliği 25 piksel seçilmiştir. Sağ üstteki kutucukta g harfinin kuyruğu rahatlıkla

görülebilirken, onun altındaki kutucuğun kalın çerçevesinden dolayı gözükmemektedir.

Diğer dikkat edilecek husus, internet adres kutucuklarına adres yazmak için karakter dizgisi değil, "`unsigned int`" tipinde bir değişken kullanma gerekliliğidir:

```
ICG_IPAddressSunken(10, 130, 300, 25, 0xc0060708);
ICG_IPAddressThick(370, 130, 300, 25, 0x80050403);
```

Bu fonksiyonların son parametreleri okunması nispeten daha kolay olduğu için onaltılık düzen kullanılarak yazılmıştır. C++'da 0x ile başlayan sayılar onaltılık düzende yazılmış demektir. 0xc0'ın onluk düzendeki karşılığı 128dir.

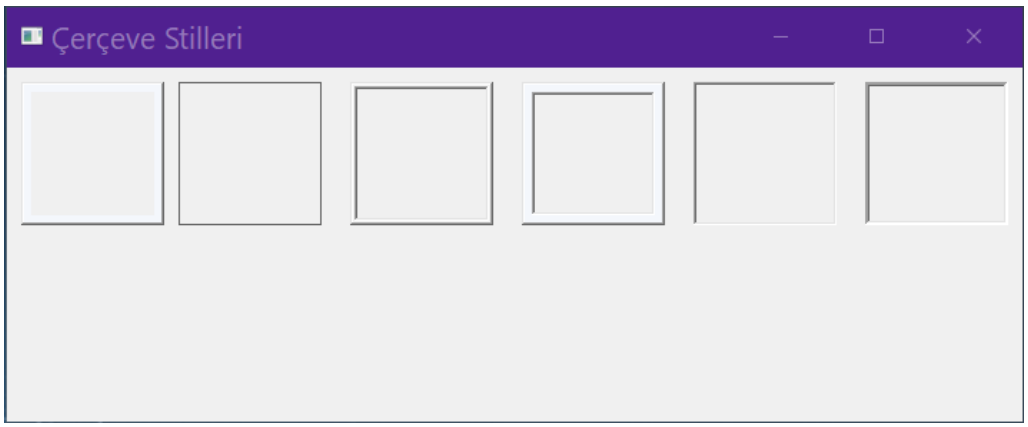
4.4. Resim Çerçevesi Stilleri

I-See-GUI kütüphanesinde altı çeşit resim çerçevesi stili bulunmaktadır. Bu stilleri yaratan bir kod örneği ve ortaya çıkan pencerenin görüntüsü aşağıda

```
#include"icb_gui.h"

void ICGUI_Create()
{
	ICG_MWSize(730, 300);
	ICG_MWTitle("Çerçeve Stilleri");
}

void ICGUI_main()
{
	ICG_FramePanel(10, 10, 100, 100);
	ICG_FrameThin(120, 10, 100, 100);
	ICG_FrameMedium(240, 10, 100, 100);
	ICG_FrameThick(360, 10, 100, 100);
	ICG_FrameSunken(480, 10, 100, 100);
	ICG_FrameDeep(600, 10, 100, 100);
}
```



verilmiştir. Hiçbir çerçevesi olmadığından burada gösterilemeyen bir de ICG_Frameless() metodu bulunmaktadır.

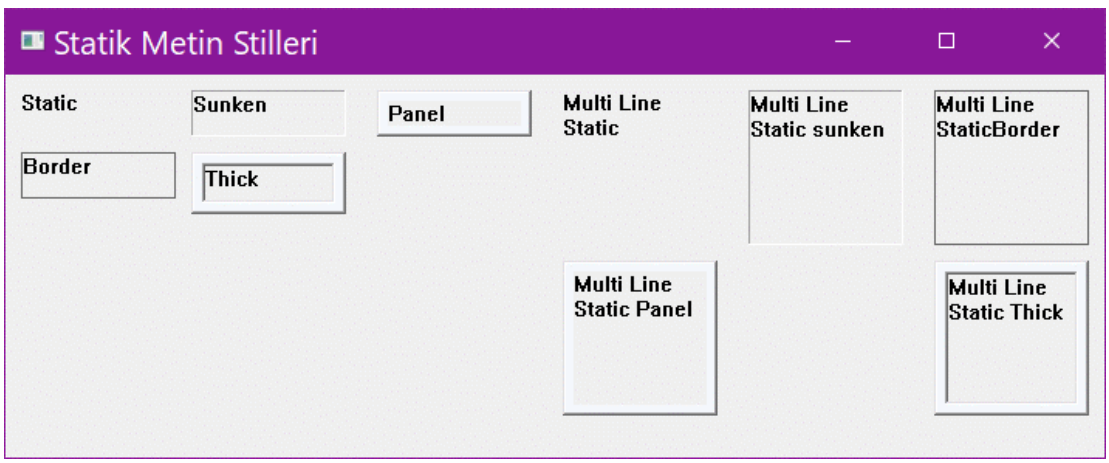
4.5. Statik Metin Stilleri

ICGUI kütüphanesindeki farklı statik metin stillerini oluşturan kod ve bu kodun çıktısı aşağıda gösterilmiştir.

```
#include"icb_gui.h"

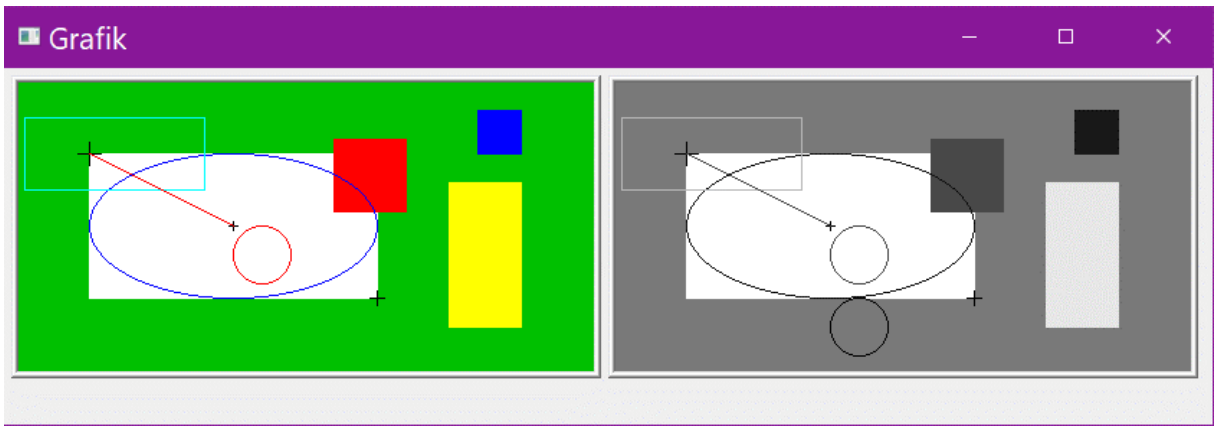
void ICGUI_Create()
{
  ICG_MWSize(730, 300);
  ICG_MWTitle("Statik Metin Stilleri");
}

void ICGUI_main()
{
  ICG_Static(10, 10, 100, 30,"Static");
  ICG_StaticBorder(10, 50, 100, 30, "Border");
  ICG_StaticSunken(120, 10, 100, 30,"Sunken");
  ICG_StaticThick(120, 50, 100, 40, "Thick");
  ICG_StaticPanel(240, 10, 100, 30,"Panel");
  ICG_MLStatic(360, 10, 100, 100,"Multi Line Static");
  ICG_MLStaticPanel(360, 120, 100, 100, "Multi Line Static
Panel");
  ICG_MLStaticBorder(600, 10, 100, 100, "Multi Line
StaticBorder");
  ICG_MLStaticSunken(480, 10, 100, 100, "Multi Line Static
sunken");
  ICG_MLStaticThick(600, 120, 100, 100, "Multi Line Static
Thick");
}
```



5. Şekil Çizme

I-See-Bytes kütüphanesi öğrenci versiyonunda grafik çizimleri için temel bazı yetenekler eklenmiştir. Bu yetenekler sayesinde öğrencilerin bilgisayar grafiği içeren alanlar olan kullanıcı arabirimi veya oyun programlama gibi konuları daha hızlı öğrenebilmeleri sağlanmaktadır. Aşağıdaki uygulamanın ekran görüntüsü bu özelliklerin bazılarını göstermektedir.



Bu örnekte temel grafik elemanları olan düz çizgi, dikdörtgen, çember ve elipsin nasıl çizileceği gösterilecektir. Uygulamanın sol tarafında32 bit renkli bir resim gösterilmişken, sağ tarafında ise bunun siyah beyaz versiyonu gösterilmektedir. Bu şekilleri çizmek için dört farklı fonksiyon kullanılmıştır. Bu fonksiyonların ilk argümanı ICBYTES akışkanı türündendir. Sonraki iki argümanı ise şeklin sol üst köşesi koordinatıdır. Elips için bile, bu iki argüman, içine çizilebildiği dikdörtgenin sol üst köşesi olmalıdır. Çizgi için ise başlangıç koordinatlarıdır. Bu koordinatlar sıfırdan değil, birden başlar çünkü bir matrisin içerisine yazılmaktadır. Beyaz dikdörtgen ve onun içine çizili mavi elipsi çizdiren fonksiyonlar aşağıda gösterilmiştir:

```
FillRect(renkli, 50, 50, 200, 100, 0xffffff);
Ellipse(renkli, 50, 50, 100, 50, 0xff);
```

Burada FillRect (Filled Rectangle: İçi dolu dikdörtgen) adlı fonksiyon ismi "renkli" olan ICBYTES cinsinden matris taşıyan akışkanın üzerinde çizim yapmaktadır. Sol üst köşe koordinatları x:50 ve y:50 olan ve yatay genişliği 200 piksel, düşey uzunluğu ise 100 piksel olan rengi beyaz (0xffffff: K:255 Y:255 M:255) bir dikdörtgen çizmektedir. Ellipse() fonksiyonu ise yine sol üst köşe koordinatları x:50 ve y:50 olan ve yatay yöndeki yarıçapı Rx:100, düşey yöndeki yarıçapı Ry:50 piksel olan, rengi 0xff (Mavi:255) bir elips çizmektedir. Önce beyaz dikdörtgen çizildiği için elips onun üzerine çizilmektedir. Aksi takdirde dikdörtgen elipsin üzerine çizilip onun görünmesini engelleyecektir. Resimde içi dolu olan tüm kare ve dikdörtgenler FillRect() fonksiyonu ile, tüm elips ve çemberler ise Ellipse() fonksiyonu ile çizilmiştir. Çember çizmek için tek gereken Rx ve Ry parametrelerinin aynı değere sahip olmasıdır.

Resimde dikdörtgenin sol üst köşesinden başlayıp orta noktasında sonlanan kırmızı renkli çizgi ve turkuaz renkli içi boş dikdörtgen ise, sırasıyla:

```
Line(renkli, 50, 50, 150, 100, 0xff0000);
Rect(renkli, 5, 25, 125, 50,0xffff);
```

Fonksiyonları ile çizdirilmiştir. Line() fonksiyonu çizginin başlangıç ve bitiş noktalarının koordinatlarına ihtiyaç duyar. Rect() ise FillRect() ile aynı argümanları alır.

Yukarıda ekran görüntüsü verilmiş olan uygulamanın C++ kodu aşağıda gösterilmiştir.

```
#include"icb_gui.h"

void ICGUI_Create()
{
  ICG_MWTitle("Grafik");
  ICG_MWSize(860, 300);
}

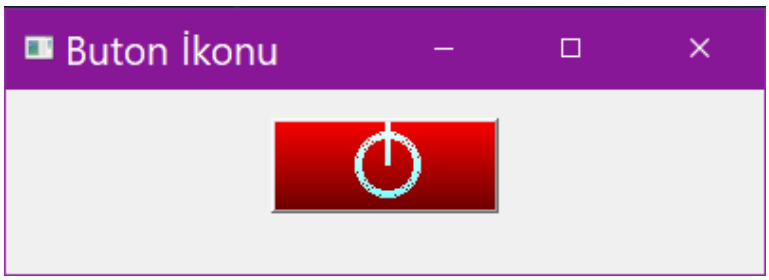
void ICGUI_main()
{
  int FRM1,FRM2;
  static ICBYTES renkli,siyahbeyaz;
  CreateImage(renkli, 400, 200, ICB_UINT);
  FRM1=ICG_FrameMedium(5, 5, 400, 200);
  FRM2 = ICG_FrameMedium(420, 5, 400, 200);
  renkli = 0xc000;
  FillRect(renkli, 50, 50, 200, 100, 0xffffff);
  FillRect(renkli, 220, 40, 50, 50, 0xff0000);
```

```
  FillRect(renkli, 300, 70, 50, 100, 0xffff00);
  FillRect(renkli, 320, 20, 30, 30, 0xff);
  Ellipse(renkli, 50, 50, 100, 50, 0xff);
  Ellipse(renkli, 150, 100, 20, 20, 0xff0000);
  Rect(renkli, 5, 25, 125, 50,0xffff);
  MarkPlus(renkli, 50, 50, 15, 0);
  MarkPlus(renkli, 150, 100, 5, 0);
  MarkPlus(renkli, 250, 150, 10, 0);
  Line(renkli, 50, 50, 150, 100, 0xff0000);
  Luma(renkli, siyahbeyaz);
  Ellipse(siyahbeyaz, 150, 150, 20, 20, 0);
  DisplayImage(FRM1, renkli);
  DisplayImage(FRM2, siyahbeyaz);
}
```

Ekran görüntüsünde geometrik şekiller dışında, çizginin iki ucunda ve beyaz dikdörtgenin sağ alt köşesinde farklı boylarda artı (+) işareti görmekteyiz. Bu işareti oluşturmak için MarkPlus() isimli fonksiyon kullanılmıştır. ICBYTES akışkanı dışında, sırasıyla işaretin merkezinin koordinatları, artı işaretinin boyutu ve renk tonu bu fonksiyona girdi olarak verilmiştir. Gri tonlamalı resmi oluşturmak için Luma() fonksiyonu kullanılmıştır. Dikkat edildiğinde renkli resimle gri tonlamalı resim arasındaki tek farkın beyaz dikdörtgenin hemen altına çizili siyah renkli ikinci bir çember olduğu görülecektir. Bu çemberin çizdirilmesindeki amaç, burada anlatılan çizim fonksiyonlarının 8 bit renk çözünürlüğe sahip resimler üzerinde de çalıştığını göstermektir.

5.1. Buton İkonu

Bölüm 4.1'de Windows işletim sistemiyle birlikte gelen Paint programı kullanarak bir buton ikonu çizip sonra bunu bir Bitmap butonuna yapıştırmıştık. Bu kısımda aynı işi şekil çizim fonksiyonları ile yerine getireceğiz. Aşağıdaki uygulama penceresi ekran görüntüsünde biraz daha profesyonel tasarıma sahip bir buton görmektesiniz.



Bu ikonu yaratan komut silsilesi aşağıda gösterilmiştir:

```
#include"icb_gui.h"

void ICGUI_Create()
{
  ICG_MWTitle("Buton İkonu");
  ICG_MWSize(420, 150);
}

void butonfonk(){}

void ICGUI_main()
{
  int BTN1;
  static ICBYTES butonresmi;
  CreateImage(butonresmi, 120, 50, ICB_UINT);
  butonresmi = 0xff0000;
  BTN1 = ICG_BitmapButton(140, 15, 120, 50, butonfonk);
  Ellipse(butonresmi, 45, 8, 17, 17, 0xffffff);
  Ellipse(butonresmi, 46, 9, 16, 16, 0xffffff);
  Ellipse(butonresmi, 47, 10, 15, 15, 0xffffff);
  Ellipse(butonresmi, 48, 11, 14, 14, 0xffffff);
  Line(butonresmi, 61, 1, 61, 25, 0xffffff);
  Line(butonresmi, 62, 1, 62, 25, 0xffffff);
  Line(butonresmi, 63, 1, 63, 25, 0xffffff);
  for(int y=1;y<= butonresmi.Y();y++)
    for (int x = 1; x <= butonresmi.X(); x++)
    {
      butonresmi.U(x, y)-= 0x030000 * y;
    }
  SetButtonBitmap(BTN1, butonresmi);
}
```

Bu programda 32 bitlik bir resim oluşturulup piksellerin hepsi kırmızı yapılıyor:

```
static ICBYTES butonresmi;
CreateImage(butonresmi, 120, 50, ICB_UINT);
butonresmi = 0xff0000;
```

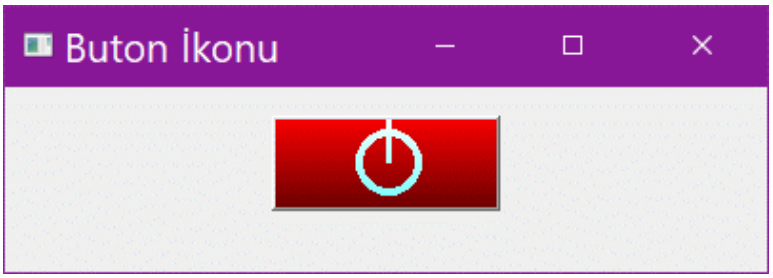
Daha sonra dört piksel kalınlığında beyaz bir çember yaratmak için iç içe dört tane beyaz çember çiziliyor. Sonra bu çemberin tepesinden ortasına doğru üç adet yan yana beyaz çizgi çiziliyor. Böylelikle sıkça kullanılan aç/kapa ikonu oluşturulmuş oluyor. Daha sonra ise butona derinlik hissi kazandırmak için matrisin ilk satırındaki piksellerin kırmızı tonundan 3, ikinci satırındakilerden 6, üçüncü satırındakilerden 9 ve bu şekilde arttırılarak en son satırından 150

çıkarılıyor. Bu da kırmızı rengin yavaş yavaş koyulaşmasına sebep olup gölge efekti yaratıyor.

Bu örnekte dört piksel kalınlığında beyaz bir çember yaratmak için iç içe dört tane beyaz çember çizdik. Maalesef bunun sonucu olarak çemberler arasında boşluklar meydana geldi. Daha iyi bir yöntem ise FillEllipse() kullanarak önce beyaz bir daire yaratıp, sonra da içerisinde yarıçapı dört piksel daha küçük kırmızı bir daire yaratmak olurdu:

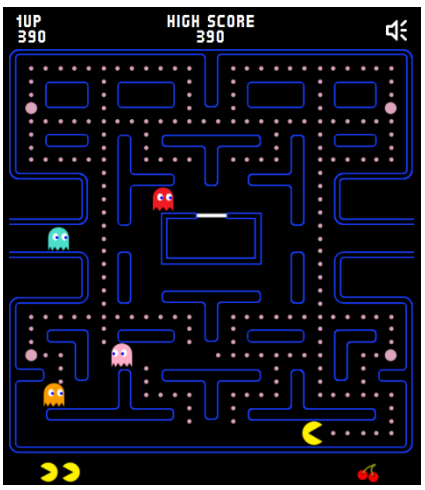
```
FillEllipse(butonresmi, 45, 8, 17, 17, 0xffffff);
FillEllipse(butonresmi, 49, 12, 13, 13, 0xff0000);
```

Aşağıda görüldüğü üzere, bu yöntemle yaratılan çemberde boşluklar oluşmamaktadır. Bu da bize şekil çizerken birden fazla gidiş yöntemi olduğunu ve bu yöntemlerden bazılarının diğerlerine göre daha iyi sonuç verdiğini göstermektedir.



5.2. Karmaşık Çizimler

Farz edelim ki bir oyun programı yazmak istiyorsunuz ve bunun için çok sayıda geometrik şekli ekrana çizdirmeniz gerekiyor. Örneğin aşağıda gösterilmiş olan 80'lerin meşhur oyunu Pacman'i yazmak istediğiniz varsayalım.



Bu oyundaki gibi labirentleri oluşturmak için çok sayıda geometrik şekil çizdirmeniz gerekiyor ve bu şekillerin her biri için bir fonksiyon çağırmak yazacağınız kod miktarını yüzlerce satır haline getirecektir. Oysa aşağıda verilen kod örneğinde tek yapmanız gereken bu geometrik şekillerin koordinatlarına sahip olmaktır:

```
#include"icb_gui.h"

ICBYTES labirent = { {270, 50, 20, 80},{210, 110, 140, 20},{270, 170, 20,80},
{210, 230, 140, 20},{270, 410, 20, 80}, {210, 470, 140, 20},{150, 170, 80, 20},
{330, 170, 80, 20},{50, 470, 60, 20},{50, 530, 60, 40},{150, 530, 80, 40},
{150, 230, 20, 80},{150, 350, 20, 140},{150, 410, 80, 20},{450, 470, 60, 20},
{450, 530, 60, 40},{330, 530, 80, 40 },{390, 230, 20, 80},{390, 350, 20, 140},
{330, 410, 80, 20},{50, 50, 180, 20},{150, 50, 20, 80},{330, 50, 180, 20},
{390, 50, 20, 80},{90, 110, 20, 80},{50, 170, 60, 20},{450, 110, 20, 80},
{450, 170, 60, 20} };

ICBYTES zindan = { {210, 290, 350, 290},{350, 290, 350, 370},{350,370,300,370},
{300, 370,300,360},{300,360,340,360},{340,360,340,300},{340,300,220,300},
{220,300,220,360},{220,360,260,360},{260,360,260,370},{260,370,210,370},
{210,370,210,290} };

ICBYTES cepheic = { {10,10,550,10},{550,10,550,110},{550,110,510,110},
{510,110,510,130},{510,130,550,130},{550,130,550,230},{550,230,450,230},
{450,230,450,310},{450,310,550,310},{550,310,550,350},{550,350,450,350},
{450,350,450,430},{450,430,550,430},{550,430,550,610},{550,610,290,610},
{290,610,290,530},{290,530,270,530},{270,530,270,610},{270,610,10,610},
{10,610,10,430},{10,430,110,430},{110,430,110,350},{110,350,10,350},
{10,350,10,310},{10,310,110,310},{110,310,110,230},{110,230,10,230},
{10,230,10,130},{10,130,50,130},{50,130,50,110},{50,110,10,110},
{10,110,10,10} };

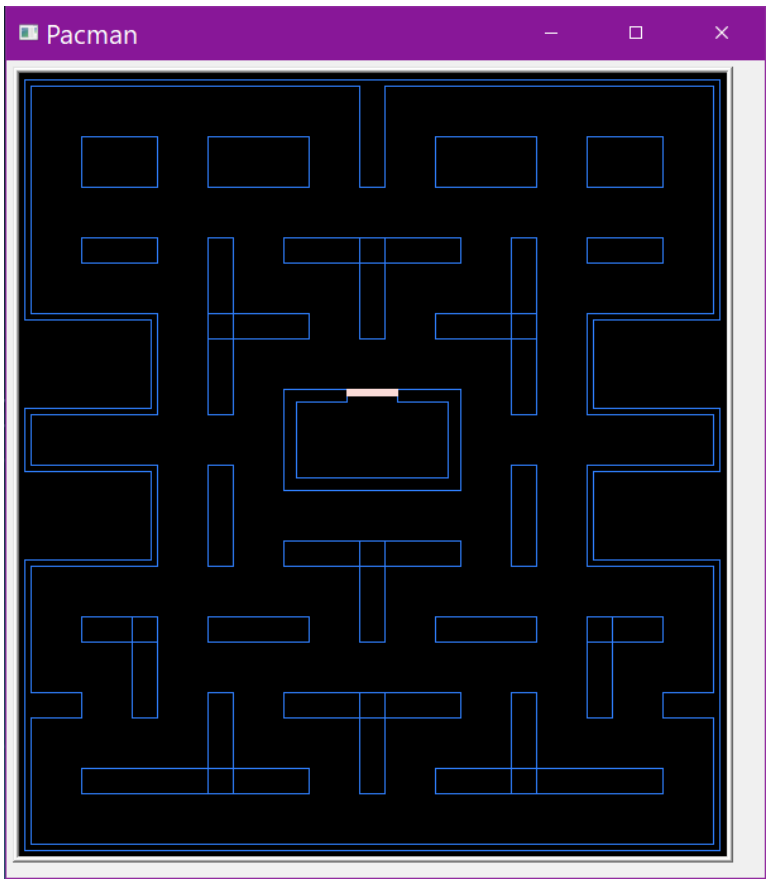
ICBYTES cephedis = { {5,5,555,5},{555,5,555,235},{555,235,455,235},
{455,235,455,305},{455,305,555,305},{555,305,555,355},{555,355,455,355},
{455,355,455,425},{455,425,555,425},{555,425,555,615},{555,615,5,615},
{5,615,5,425},{5,425,105,425},{105,425,105,355},{105,355,5,355},
{5,355,5,305},{5,305,105,305},{105,305,105,235},{105,235,5,235},
{5,235,5,5} };

void ICGUI_Create()
{
  ICG_MWTitle("Pacman");
  ICG_MWSize(620, 700);
}

void ICGUI_main()
{
  int FRM1;
  static ICBYTES saha,ekran;
  int color = 0x2e27ff;
  CreateImage(saha, 560, 620, ICB_UINT);
  FRM1 = ICG_FrameMedium(5, 5, 400, 200);
  saha = 0;
  Line(saha, zindan, 0x3080ff);
  Line(saha, cepheic, 0x3080ff);
  Line(saha, cephedis, 0x3080ff);
```

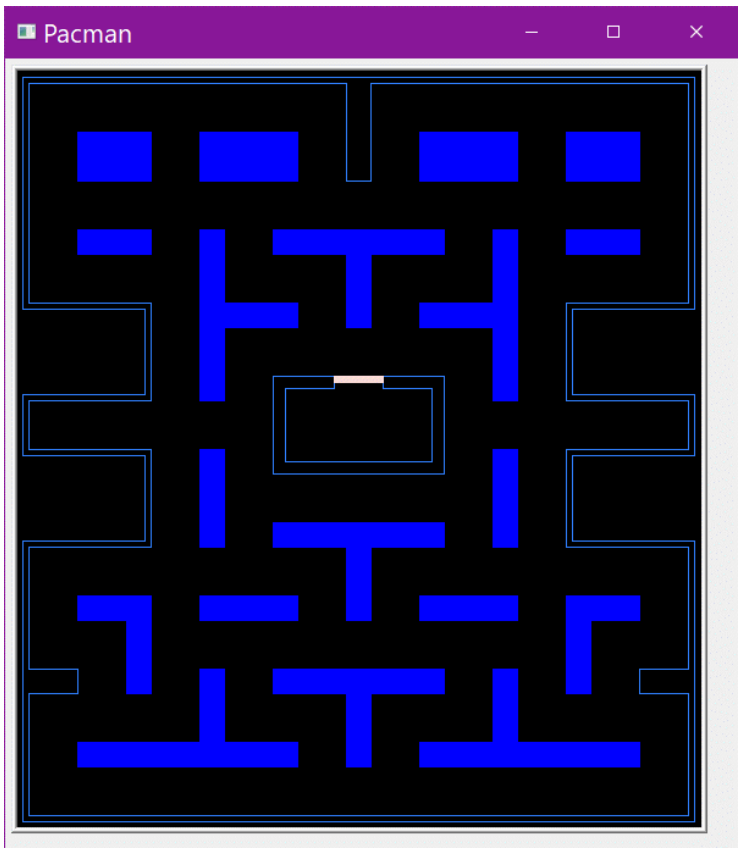
```
  FillRect(saha,260, 365, 40, 5, 0xFADBD8);
  Rect(saha, labirent, 0x3080ff);
  FlipV(saha, ekran);
  DisplayImage(FRM1, ekran);
}
```

Toplam 50 satırdan oluşan bu kod aşağıda görüldüğü gibi Pacman oyununun labirentini çizebilmektedir:



Bu örnekte fonksiyon çağırma sayımızın düşme sebebi Line() ve Rect() gibi fonksiyonların piksel koordinatları yerine bu koordinatları içeren matrisleri girdi olarak kabul etmeleridir. Bu fonksiyonlar, tam sayı elamanları olan ve X boyutunda en az dört elemanı olan (m×n; m>3) herhangi boyda bir matrisi kullanarak sayısız geometrik şekil çizebilmektedirler. Bu koordinatları hesaplayabilen bir grafik tasarım programı kullanıldığında oyun tasarlamak çok kolay olmaktadır.

Üstelik te kısa bir dokunuşla labirentin görünümü kolaylıkla değiştirilebilmektedir. Örneğin labirenti oluşturmak için Rect() fonksiyonu yerine FillRect() kullanıp rengini de tam mavi (0xff) yaptığımızda aşağıdaki görüntüyü elde ederiz.



5.3. Resim İşaretleme

Birçok uygulama grafik çizimleri gerektirmektedir. Bunlara örnek olarak finansal programları verebiliriz. Trendleri gözleyebilmek için finansal veriler çeşitli türde grafikler olarak ekranda çizilebilirler. Verilerin çizilmesi gerekebilecek diğer alanlar ise fizik ve matematik gibi temel alanlardır. Burada da ölçülen veya hesaplanan veriler eğriler olarak ekrana yansıtılabilirler. Bu grafikler üzerinde önemli noktaları işaretleyip değerleri resmin üzerine yazabilmek için "Sadece Baytlar Görüyorum" kütüphanesi içerisinde metotlar mevcuttur.

Bunlardan ilk akla gelen çember çizme fonksiyonudur. Bu fonksiyon kullanılarak grafiğin değer okunan noktalarını işaretlemek için küçük çemberler çizilebilir:

```
bool Circle(ICBYTES& i, int x, int y, int r, int color);
```

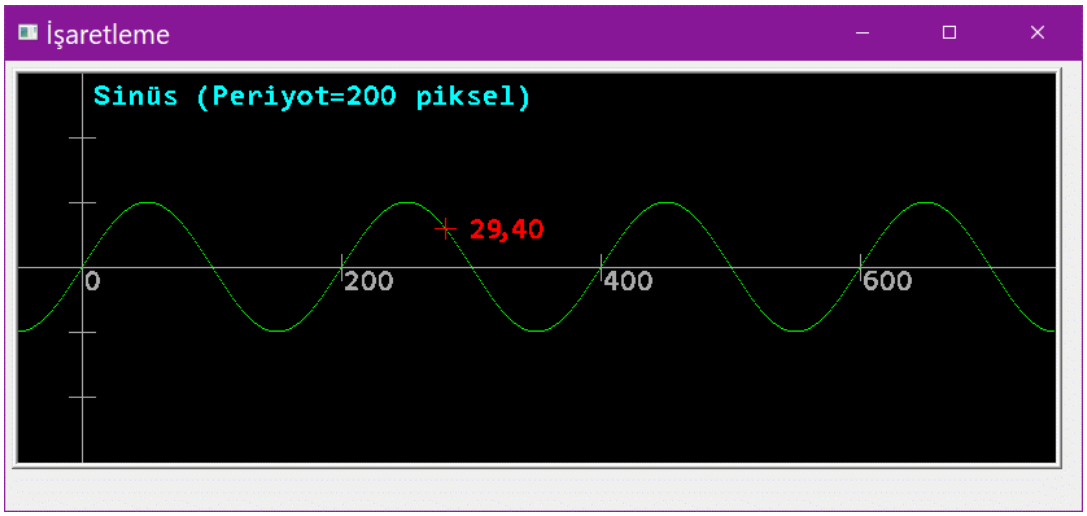
Bu fonksiyonda x ve y çemberin merkezini, r ise yarıçapını belirtmektedir. Bunun dışında sadece resim işaretlemede kullanılan fonksiyonlardan bazıları şunlardır:

```
void MarkPlus(ICBYTES& i, int x, int y, int size, int color);
void MarkVert(ICBYTES& i, int x, int y, int size, int color);
void MarkHorz(ICBYTES& i, int x, int y, int size, int color);
```

Bu fonksiyonlardan en üstteki merkez koordinatı (x,y) olan ve ebatları (size) değişkeni ile betimlenen artı şeklinde bir şekil çizerken, MarkVert() dikey, MarkHorz() ise yatay bir çizgi çizer. İşaretleme fonksiyonları arasında en önemlisi kuşkusuz resim üzerine yazı yazmanıza olanak veren aşağıdaki fonksiyondur.

```
void Impress12x20(ICBYTES& i, int x, int y, const char* txt,
                  unsigned color);
```

Bu fonksiyon, 12x20 piksel boyunda, sabit genişlikte bir font kullanarak, "txt" karakter dizgesi içine yazdığınız her şeyi sağ üst köşesi (x,y) koordinatları ile belirlenen noktadan başlayarak tek satır boyunca resim üzerine yazar. Aşağıda bu fonksiyonlar kullanılarak yazılmış bir uygulamanın ekran görüntüsü verilmiştir.



Bu uygulamada, tepe noktası 50, bir tam devinimi ise 200 piksel olan bir sinüs sinyalinin grafiği çizdirilmektedir. Yatay koordinatının x=330 olduğu noktadaki değeri (+) sembolü kullanılarak işaretlenip, o noktada fonksiyonun almış olduğu değer (29.40) resmin üzerine yazdırılmıştır. Ayrıca resmin tepesine, sinyalin özellikleri turkuaz renkle yazdırılmıştır. Bir sonraki sayfada bu uygulamanın kodu verilmiştir.

```
#include"icb_gui.h"
#include<math.h>

void ICGUI_Create()
{
  ICG_MWTitle("İşaretleme");
  ICG_MWSize(850, 400);
}

void ICGUI_main()
{
  int FRM1;
  static ICBYTES resim;
  FRM1 = ICG_FrameMedium(5, 5, 400, 200);
  CreateImage(resim, 800, 300, ICB_INT);
  Impress12x20(resim, 60, 10, "Sinüs (Periyot=200 piksel)", 0x00ffff);
  Line(resim, 50, 1, 50, 300, 0xaaaaaa);
  Line(resim, 1, 150, 800, 150, 0xaaaaaa);
  MarkVert(resim, 250, 150, 20, 0xaaaaaa);
  MarkVert(resim, 450, 150, 20, 0xaaaaaa);
  MarkVert(resim, 650, 150, 20, 0xaaaaaa);
  MarkHorz(resim, 50, 100, 20, 0xaaaaaa);
  MarkHorz(resim, 50, 50, 20, 0xaaaaaa);
  MarkHorz(resim, 50, 200, 20, 0xaaaaaa);
  MarkHorz(resim, 50, 250, 20, 0xaaaaaa);
  Impress12x20(resim, 53, 153, "0", 0xaaaaaa);
  Impress12x20(resim, 253, 153, "200", 0xaaaaaa);
  Impress12x20(resim, 453, 153, "400", 0xaaaaaa);
  Impress12x20(resim, 653, 153, "600", 0xaaaaaa);
  double f;
  int y,ondalikbasamak,isaret;
  char tam[8],ondalik[8];
  for (int x = 1; x < 800; x++)
  {
    f = 50.0 * sin(3.1415 * (double)(x-50) * 0.01);
    y =150.0- f;
    resim.I(x, y) = 0x00ff00;
    if (x == 330)
    {
      MarkPlus(resim, x, y, 15, 0xff0000);
      _ecvt_s(tam, 32, f, 4, &ondalikbasamak, &isaret);
      _ecvt_s(ondalik, 32, f, 4, &ondalikbasamak, &isaret);
      tam[ondalikbasamak] = 0;
      Impress12x20(resim, x+20, y-7, tam, 0xff0000);
      Impress12x20(resim, x + (ondalikbasamak+1)*13, y - 7,
",",0xff0000);
      Impress12x20(resim, x + (ondalikbasamak + 2) * 13, y - 7,
&ondalik[ondalikbasamak], 0xff0000);
    }
  }
  DisplayImage(FRM1, resim);
}
```

Bu kodu incelediğimizde gözümüze ilk çarpan projemize `"icb_gui.h"` dışında eklenen `<math.h>` kütüphanesidir. Bu kütüphane sinüs fonksiyonu sin()'i çağırabilmek için eklenmiştir. Daha sonra ICGUI_main() fonksiyonu içerisinde "resim" adında bir akışkan içerisinde 800x300 boyutlarında bir resim matrisi oluşturulmaktadır. Hemen arkasından bu resmin sol üstüne Impress12x20() fonksiyonu kullanılarak turkuaz renkte `"Sinüs (Periyot=200 piksel)"` yazısı yazdırıldığını görmekteyiz.

Line() fonksiyonu arkası arkasına iki kez çağrılarak x ekseni ve y ekseni çizdirilmekte, daha sonra MarkVert() metotu ile x ekseninin 200, 400, ve 600 noktalarına dikey çentikler atılmaktadır. Aynısı y ekseni üzerinde MarkHorz() fonksiyonu kullanılarak yapılmakta ve böylelikle x ve y eksenleri çizilmektedir. Daha sonra Impress12x20() fonksiyonu kullanılarak orijinden başlayarak x eksenindeki çentiklerin sağ alt tarafına 0, 200, 400 ve 600 yazılmaktadır.

Tüm bunların arkasından sinüs sinyalini çizdirmek için x=1 den 800'e kadar artan for döngüsü içerisinde y değeri olarak:

```
f = 50.0 * sin(3.1415 * (double)(x-50) * 0.01);
```

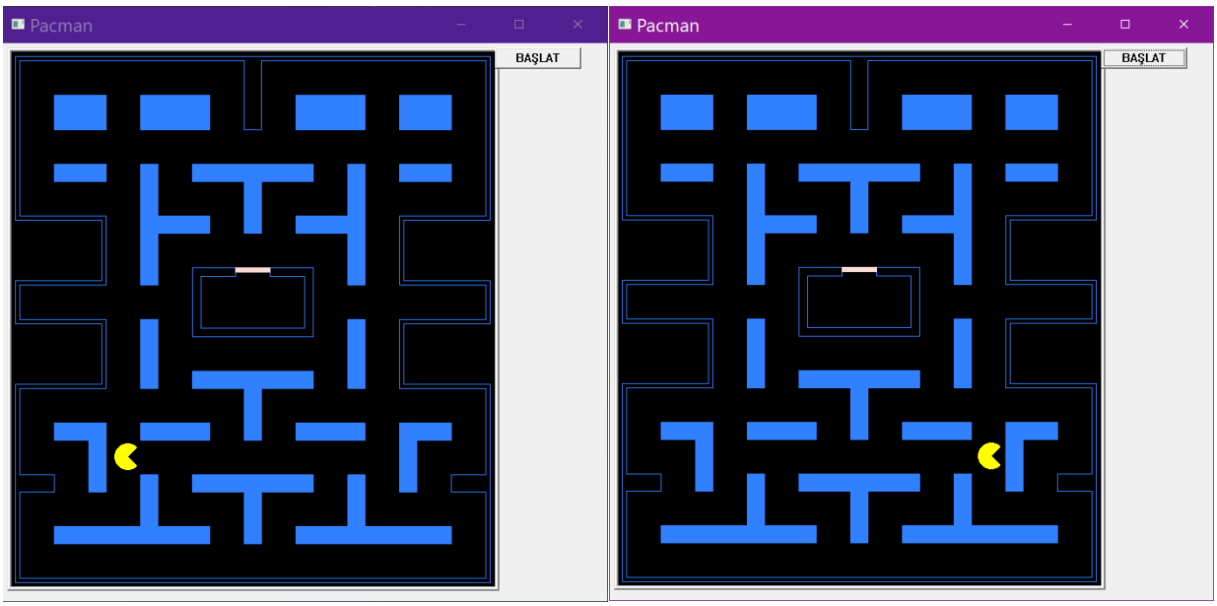
genliği 50, periyotu 200 piksel olan ve 50 piksel sola kaydırılmış bir sinüs fonksiyonu hesaplanıp çizdirilmekyedir. For döngüsü indeksi olan x 330 değerine ulaştığında bir "if" şartı sinüs fonksiyonunun o noktada aldığı değeri kırmızı renkli bir artı işareti ile işaretleyip yanına da değerini yazmaktadır. Duble kayan nokta sayıları bir karakter dizgisine dönüştürmek için _ecvt_s() fonksiyonu kullanılmıştır.

5.4. Basit Animasyon

Oyun programcılığı aslında interaktif animasyon programlama olarak özetlenebilir. Sabit bir arka plan çizildikten sonra oyuncunun ve bilgisayarın yönettiği farklı hareketli nesnelerin çizilmesi gerekir. Oyunculuk jargonunda bu nesnelere yaratık (sprite) veya Hareketli Nesne Bloğu (HNB) (MOB) denmektedir.

Ancak günümüzde kısa süreli animasyonlar kullanıcı arabiriminin de vazgeçilmez parçası olmaya başlamışlardır. Mesela üzerine tıklamanızı isteyen bir butonun yanıp sönmesi veya bir ayar kutucuğun şekil değiştirmesi basit ve kısa süreli animasyonlar kategorisinde düşünülebilir. Bunun nasıl yapıldığına örnek olarak Pacman oyununa devam edelim. Aşağıda, bölüm 5.2'de labirentini

çizdiğimiz Pacman oyununun başkahramanı olan HNB'yi yatay yönlü hareket ettiren bir animasyonun başlangıç ve bitiş anlarının ekran görüntüsü verilmiştir.



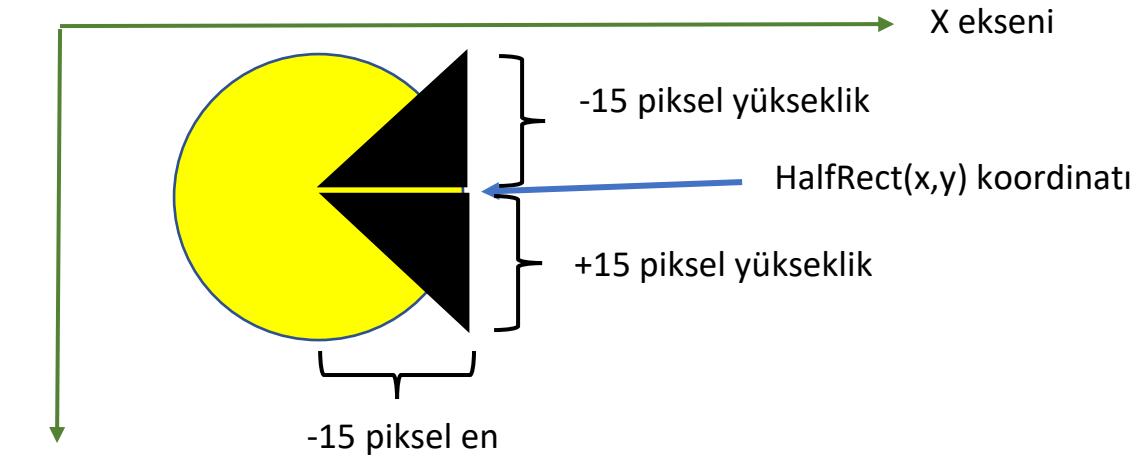
Burada gösterilmiş olan dairesel Pacman nesnesi, resim üzerinde merkezi (165,470) olan noktadan başlayıp aşağı yukarı 2,5 saniyede yatay olarak 300 piksel kaymaktadır. Bu animasyonu gerçekleştiren buton fonksiyonu aşağıda verilmiştir.

```
int FRM1;

void Baslat(void *i)
{
  ICBYTES& ekran = *(ICBYTES*)i;
  for (int x = 150; x < 450; x+=2)
  {
    FillCircle(ekran, x - 15, 470, 15, 0xffff00);
    HalfRect(ekran, x, 470, -15, -15, 0);
    HalfRect(ekran, x, 470, -15, 15, 0);
    DisplayImage(FRM1, ekran);
    Sleep(10);
    FillRect(ekran, x - 30, 455, 30, 30, 0);
  }
}
```

Bu fonksiyon içerisinde üç adet yeni metotla karşılaşıyoruz. Bunlardan birincisi olan FillCircle(), adından da anlaşılacağı üzere merkezi (x,y) olan ve yarıçapı 15 olan içi dolu sarı bir daire çizmektedir. Arkasından Pacman'in ağzını oluşturmak için HalfRect() isimli metot ile siyah dik üçgenler çizilerek sarı

dairenin bir kısmı silinmektedir. Bunun nasıl olduğu aşağıda detaylı bir şekilde gösterilmiştir.



HalfRect() isimli metodun ilk iki girdisi oluşturulan üçgenin dik köşesinin koordinatlarıdır. Sonraki iki girdisi ise oluşturulan üçgenin genişliği ve uzunluğudur. Ancak bunlar eksi olurlarsa üçgen ters yönde çizilir.

Pacman çizildikten sonra Sleep(10) komutu ile program 10 milisaniye boyunca dondurulmaktadır. 10 milisaniye saniyenin 1/100'üdür. Daha sonra üzerine siyah, dolu bir dikdörtgen çizilierek Pacman ekrandan tamamen silinir. Ancak hemen arkasından Pacman iki piksel sağda yeniden çizilir ve gözümüz bunu Pacmanin hareketi olarak algılar. Bu işlemler 150 kere tekrar edildiğinde gözümüz bunları Pacman'i sağa doğru kaydıran bir animasyon olarak görür.

Bu koddaki diğer farklı husus bir ICBYTES akışkanının buton fonksiyonuna parametre olarak gönderilmiş olmasıdır. Buton fonksiyonu paramteresi sadece void cinsi bir işaretçi olabilir. Bu nedenle tip kaydırma yapılarak "i" adlı işaretçi ICBYES cinsinden bir referans'a dönüştürülmüştür:

```
ICBYTES& ekran = *(ICBYTES*)i;
```

ICGUI_main() fonksiyonunun içeriği de aşağıda verilmiştir:

```
void ICGUI_main()
{
  static ICBYTES saha, ekran;
  int color = 0x2e27ff;
  CreateImage(saha, 560, 620, ICB_UINT);
  ICG_Button(570, 5, 100, 25, "BAŞLAT", Baslat, &ekran);
```

```
  FRM1 = ICG_FrameMedium(5, 5, 400, 200);
  saha = 0;
  Line(saha, zindan, 0x3080ff);
  Line(saha, cepheic, 0x3080ff);
  Line(saha, cephedis, 0x3080ff);
  FillRect(saha, 260, 365, 40, 5, 0xFADBD8);
  FillRect(saha, labirent, 0x3080ff);
  FlipV(saha, ekran);
  DisplayImage(FRM1, ekran);
}
```

5.5. Daha Karmaşık Figürler

Pacman'i çizdirmek için daire ve yarım dikdörtgenler yeterli oldu. Ancak birçok oyunda çok daha detaylı nesneler çizmek gerekmektedir. I-See-Bytes kütüphanesinde bu tür daha karmaşık grafik nesnelerini çizmenize yardımcı olacak fonksiyonlar da bulunmaktadır. Bu fonksiyonlardan biri Paste() fonksiyonudur. Örneğin aşağıdaki gibi bir resmi ReadImage() fonksiyonu kullanarak bir matrise yükledikten sonra Paste() komutu kullanarak daha büyük bir resmin içerisine yapıştırabilirsiniz.

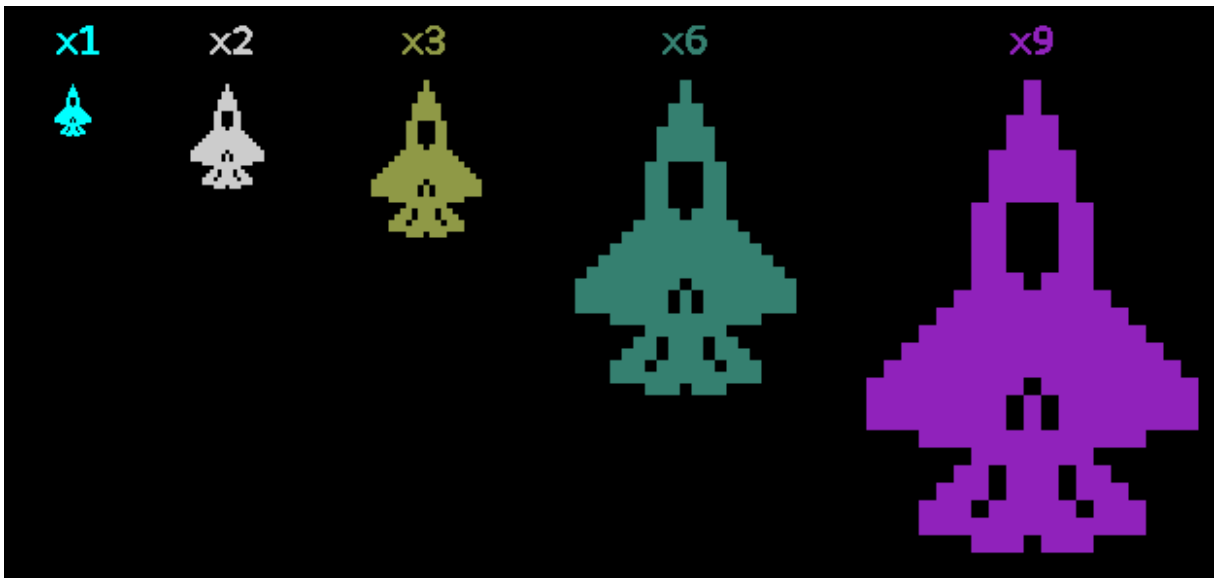
Diğer taraftan SetPixels() fonksiyonu kullanarak da grafik nesneleri yaratabilirsiniz. Aşağıdaki kod örneğinde Türk savaş uçağı Kaan'ın basit bir çiziminin nasıl yapılacağı gösterilmektedir.

```
ICBYTES m;
int color = 0x2e27ff;
CreateImage(m, 210, 100, ICB_UINT);
int FRM = ICG_FrameMedium(5, 5, 160, 110);

ICBYTES kaan = { {10,1},{10,2},{9,3},{10,3},{11,3},{9,4},{10,4},{11,4},{8,5},
{9,5},{10,5},{11,5},{12,5},{8,6},{9,6},{10,6},{11,6},{12,6},{8,7},{9,7},{10,7},{11
,7},{12,7},{7,8},{8,8},{12,8},{13,8},{7,9},{8,9},{12,9},{13,9},{7,10},{8,10},{12,1
0},{13,10},{7,11},{8,11},{12,11},{13,11},{7,12},{8,12},{9,12},{11,12},{12,12},{13,
12},{6,13},{7,13},{8,13},{9,13},{10,13},{11,13},{12,13},{13,13},{14,13},{5,14},{6,
14},{7,14},{8,14},{9,14},{10,14},{11,14},{12,14},{13,14},{14,14},{15,14},{4,15},{5
,15},{6,15},{7,15},{8,15},{9,15},{10,15},{11,15},{12,15},{13,15},{14,15},{15,15},
{16,15},{3,16},{4,16},{5,16},{6,16},{7,16},{8,16},{9,16},{10,16},{11,16},{12,16},{
13,16},{14,16},{15,16},{16,16},{17,16},{2,17},{3,17},{4,17},{5,17},{6,17},{7,17},{
8,17},{9,17},{10,17},{11,17},{12,17},{13,17},{14,17},{15,17},{16,17},{17,17},{18,1
7},{1,18},{2,18},{3,18},{4,18},{5,18},{6,18},{7,18},{8,18},{9,18},{11,18},{12,18},
{13,18},{14,18},{15,18},{16,18},{17,18},{18,18},{19,18},{1,19},{2,19},{3,19},{4,19
},{5,19},{6,19},{7,19},{8,19},{10,19},{12,19},{13,19},{14,19},{15,19},{16,19},{17,
19},{18,19},{19,19},{1,20},{2,20},{3,20},{4,20},{5,20},{6,20},{7,20},{8,20},{10,20
},{12,20},{13,20},{14,20},{15,20},{16,20},{17,20},{18,20},{19,20},{4,21},{5,21},{6
,21},{7,21},{8,21},{9,21},{10,21},{11,21},{12,21},{13,21},{14,21},{15,21},{16,21},
{7,22},{8,22},{9,22},{10,22},{11,22},{12,22},{13,22},{6,23},{7,23},{9,23},{10,23},
```

```
{11,23},{13,23},{14,23},{5,24},{6,24},{7,24},{9,24},{10,24},{11,24},{13,24},{14,24
},{15,24},{4,25},{5,25},{6,25},{8,25},{9,25},{10,25},{11,25},{12,25},{14,25},{15,2
5},{16,25},{4,26},{5,26},{6,26},{7,26},{8,26},{9,26},{10,26},{11,26},{12,26},{13,2
6},{14,26},{15,26},{16,26},{7,27},{8,27},{9,27},{11,27},{12,27},{13,27}};
m = 0;
SetPixels(kaan, 65, 15, 0x8f9946, m);//x3
ICBYTES  p, q;
CreateImage(p, 19, 27, ICB_UINT);
SetPixels(kaan, 1, 1, 0x9022bb, p);//x9
MagnifyX3(p, q);
Paste(q, 150, 15, m);//x9
SetPixelsX2(kaan, 100, 15, 0x358070, m);//x6
MagnifyX3(m, panel);
SetPixels(kaan, 30, 45, 0xffff, panel);
Impress12x20(panel, 30, 15, "x1", 0xffff);
SetPixelsX2(kaan, 100, 45, 0xcccccc, panel);
Impress12x20(panel, 109, 15, "x2", 0xcccccc);
Impress12x20(panel, 208, 15, "x3", 0x8f9946);
Impress12x20(panel, 342, 15, "x6", 0x358070);
Impress12x20(panel, 521, 15, "x9", 0x9022bb);
DisplayImage(FRM, panel);
```

Bu kod örneğinde “kaan” isimli akışkan içerisinde birçok sayıdan oluşan 2×N büyüklüğünde bir matris yaratılmaktadır. Bu matris aslında çizilecek nesnenin x ve y koordinatlarını içermektedir. Daha sonra SetPixels() fonksiyonu ile aynı boyda veya SetPixelsX2() fonksiyonu ile iki kat büyütülmüş olarak bu nesne çizdirilebilmektedir. Diğer taraftan, MagnifyX3() fonksiyonu ile herhangi bir resim matrisi üç katına büyütülebilmektedir. Dolayısı ile tüm bu fonksiyonlar birbiri ardına kullanılarak aşşağıda görüldüğü üzere Kaan uçağı çeşitli boylarda çizdirilebilmektedir.



5.6. GDI Çizim Fonksiyonları

Windows işletim sistemi Grafik Cihaz Arabirimi (Graphics Device Interface) yada kısaca GDI denilen bir kısım içerir. Bu programlama arayüzü grafik çizimleri için fonksiyon ve ardışık tanımları içerir. I-See-Bytes kütüphanesi her ne kadar Windows işletim sistemi üzerinde yazılmış olsa da gelecekte başka işletim sistemleri veya donanım platformlarına aktarılması kaçınılmazdır çünkü nihai amacı robot görü için kullanılmaktır. Robotlar ise çok farklı tipte gömülü sistem, işlemci ve işletim sistemi taşımaktadırlar. Bu nedenle I-See-Bytes kütüphanesinin GUI kısmı (kutucukları oluşturan) dışında kalan tüm kısımları Windows işletim sisteminden bağımsız çalışacak şekilde yazılmıştır. Bunlara şu ana kadar bahsettiğimiz grafik çizim fonlsiyonları da dahildir.

Ancak temel hedef Windows olduğu için I-See-Bytes kütüphanesi tam bir entegrasyon sağlamak isteyen yazılımcılar için Windows işletim sistemine birleştirilmesini mümkün kılacak tüm katmanları sunmaktadır. Bunlardan en önmelisi Windows'daki HBITMAP kulpudur. Windows, resimleri hafızasında bitmap formatında tutar ve bunlara ulaşmak için kullanıcıya HBITMAP cinsinden bir kulp sunar. I-See-Bytes kütüphanesinde, eğer CreateImage(.) kullanarak bir resim yaratırsanız bu matrisin HBITMAP kulbuna şu şekilde erişebilirsiniz:

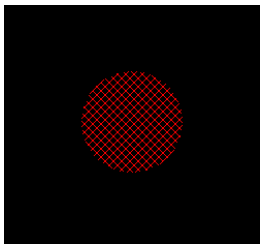
```
ICBYTES m;
int color = 0x2e27ff;
CreateImage(m, 210, 100, ICB_UINT);
HBITMAP hbmp = m.GetHBitmap();
```

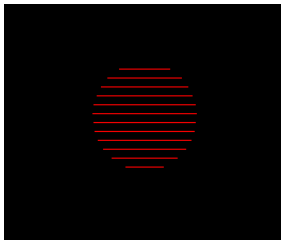
Bunun dışında, Windows GDI kullanan bazı grafik çizim fonksiyonlarına da I-See-Bytes içerisinde yer verilmiştir. Bu fonksiyonları isimlerinin sonundaki "WIN" harflerinden tanıyabilirsiniz. Bu fonksiyonlar bu bölümde anlattığımız diğer grafik çizim fonksiyonları ile aynı şekilde kullanılırlar ancak fazladan bazı özellikler içerirler. Örneğin yukarıda bahsettiğimiz FillCircle(.) fonksiyonunun Windows versiyonunu aynışekilde çağırırsanız:

```
FillCircleWIN(img, 150, 150, 50, 0xff);
```

Aşağı soldaki şekli, sonuna parametre eklediğinizde ise,

```
FillCircleWIN(img, 150, 150, 50, 0xff, HS_HORIZONTAL)
```
sağdakini elde edersiniz:





Bu fonksiyonun sonuna eklediğimiz parametre (HS_HORIZONTAL), Microsoft'un kendi tanımladığı bir parametredir. Farklı desenleri oluşturmak için Visual Studio içerisindek "wingdi.h" dosyasında tanımlanmış olan aşağıdaki seçenekleri de kullanabilirsiniz:

```
HS_HORIZONTAL
HS_VERTICAL
HS_FDIAGONAL
HS_BDIAGONAL
HS_CROSS
HS_DIAGCROSS
```

Yukarıdaki seçenekleri aynı şekilde FillRectWIN(.), FillEllipseWIN(.) fonksiyonlarında da kullanabilirsiniz.

ANCAK,
UNUTMAYIN Kİ, BU FONKSİYONLARI SADECE `CreateImage(.)` FONKSİYONU KULLANARAK YARATTIĞINIZ MATRİSLERLE KULANABİLİRSİNİZ. AYRICA GELECEKTE I-SEE-BYTES KÜTÜPHANESİ WINDOWS DIŞINDAKİ PLATFORMLARA TAŞINDIĞINDA BU FONKSİYONLARI KULLANAMAZSINIZ.

Çizgi çizmek için kullandığımız Line(.) fonksiyonunun aynısı da bulunmaktadır. Aşağıdaki şekilde çağırdığınızda Line(.) fonksiyonu ile aynı sonucu elde edersiniz:

```
LineWIN(img, 30, 30, 270, 290, 0xff0000);
```

Ancak, sonuna bir parametre daha ekleyerek çizgi kalınlığını değiştirme seçeneğiniz bulunmaktadır:

```
LineWIN(img, 30, 30, 270, 290, 0xff0000, 3);
```

Sona eklenmiş olan 3, çizilen kırmızı çizgiyi 3 pixel kalınlığa ulaştırır.

5.7. GDI ile Resim Üstüne Yazı Yazdırma

5.3 Resim İşaretleme konusu anlatılırken resim üzerine Impress12x20(.) fonksiyonu kullanarak nasıl yazı yazıldığını göstermiştik. Windows GDI kullanarak resim üzerinde çok daha fazla yazı yazdırma seçeneğine sahip olmaktayız. Bu seçeneklerden birincisi farklı fontlar kullanabilmek. Ancak web tasarımı ile uğraşmakta iseniz farklı fontlar kullanmanın tehlikelerinden de haberdarsınız demektir.

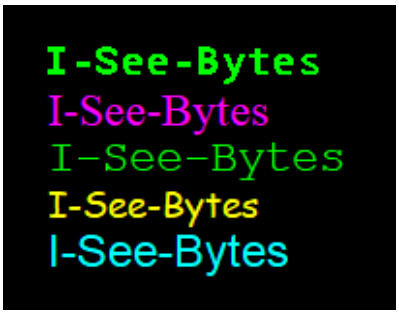
Sistemler üzerine yüklü fontlar standard değildir. Birçok font sanatsal tasarım kabul edildiğinden kullanımı parayla satın alınmaktadır. Bu nedenle Windows'ta yüklü olan bir font Android veya Linux'ta yüklü olmayabilir. Bu

durmu bilen web tasarımcılar mümkün olduğunca standart font kullanmaya çalışırlar ancak o bile garanti değildir. Aşağıdaki kodda en standart hale gelmiş fontlar kullanarak resim üzerine yazı yazmamıza rağmen, Linux üzerinde bu fontların çalışmadığına şahit olacağız:

```
ICBYTES img;
CreateImage(img, 300, 300, ICB_UINT);
Impress12x20(img, 30, 30, "I-See-Bytes", 0xff00);
DrawTextWIN(img,30, 50, "I-See-Bytes", 0xff00ff,27,0,"times");
DrawTextWIN(img, 30, 75, "I-See-Bytes", 0xff00, 27, 0, "courier new");
DrawTextWIN(img, 30, 100, "I-See-Bytes", 0xffff, 27, 0, "Comic Sans MS");
DrawTextWIN(img, 30, 125, "I-See-Bytes", 0xffff00, 27, 0, "ariel");
DisplayImage(FRM1, img);
```

Bu kodu Windows-11 işletim sistemi çalıştıran bir bilgisayarda çalıştırırsak aşağıda soldaki çıktıyı elde ederiz. İlk satırda, I-See-Bytes içerisinde yüklü olan "Impress" isimli font kullanılmaktadır. Sonraki satırlarda ise Ariel, Times ve courier new gibi en sık rastlanan fontlar kullanılmaktadır.

Daha evvel bahsettiğimiz üzere, I-See-Bytes ile derlenmiş .exe programlar intel uyumlu (AMD) işlemci üzerinde Linux işletim sistemi ile birlikte de sorunsuz çalışabilmektedir. Bunun için tek yapmanız gereken WINE uyumluluk katmanı programını Linux makinanız üzerine yüklemektir. İşte aşağıda soldaki resim ise aynı .exe dosyasının Debian Linux üzerinde WINE kullanarak çalıştırdığınızda elde edeceğiniz sonuçtur.



Açıkça görüldüğü üzere, ilk satır, yani "Impress" fontumuz hiçbir değişikliğe uğramaz. Diğer taraftan Windows'ta çalışan fontlar arasında bir tek "Ariel" fontu Linuxta doğru çizilmiştir. Diğer fontlar tanımsız olduğu için WINE programı **onların yerine de Ariel fontunu kullanmak zorunda kalmıştır**.

İŞTE BU ÖRNEK I-SEE-BYTES KÜTÜPHANESİNDE WİNDOWS PLATFORMUNA BAĞIMLI KALMAMAK İÇİN NEDEN FAZLADAN ÇABA SARFEDİLDİĞİNİ GÖSTEREN BİR KANITTIR.

WINE programı Linux üzerinde Windows işletim sistemini birebir taklit etmek için yazılmış bir programdır. Buna rağmen basit bir fontu neden taklit edememiştir? Çünkü bu fontlar parayla satılmaktadır. Ancak WINE programı kar amacı gütmeden, açık kaynak kodlu olduğu ve bedava dağıtıldığı için Microsoft'un satın alabildiği fontları satın alamamıştır. İşte bu nedenle, farklı fontlar kullanırken dikkatli olmamız gerekir. Eğer yazdığımız programı Windows dışındaki platformlar üzerinde de çalıştırmayı hedefliyorsak, o platformlarda da aynı sonucu alacağımızdan emin olmalıyız.

Hele ki Windows'tan tamamen farklı olan, Android veya gömülü sistemler içeren platformlar üzerinde de çalışmasını istiyorsak GDI fonksiyonlarından uzak durmalıyız. Ancak müşteri kitleniz sadece Windows kullanıyorsa, o takdirde bunları gönül rahatlığı ile kullanabilirsiniz.

Ayrıca GDI fonksiyonlarının I-See-Bytes'ın henüz yapamadığı bazı özellikleri de bulunmaktadır. Deminki kodu aşağıdaki gibi değiştirirseniz:

```
ICBYTES img;
CreateImage(img, 300, 300, ICB_UINT);
DrawTextWIN(img,30, 70, "I-See-Bytes", 0xff00ff,30,0,"times",350);
DrawTextWIN(img, 30, 95, "I-See-Bytes", 0xff00, 27, 0, "courier new",250);
DrawTextWIN(img, 230, 120, "I-See-Bytes", 0xffff, 27, 0, "Comic Sans MS",1900);
DrawTextWIN(img, 30, 270, "I-See-Bytes", 0xffff00, 27, 0, "ariel",900);
DisplayImage(FRM1, img);
```

Aşağıdaki görünümleri elde edersiniz.

Dikkat ettiyseniz, temelde farklı yaptığımız tek şey DrawTextWIN(.) fonksiyonunun en sonuna bir parametre eklemek oldu. Bu parametre, yazıyı döndürmek istediğimiz açının 10 katına eşit bir tam sayıdan ibarettir. Yani 35 derece yukarı kaldırmak için 350 sayısını son parametre olarak ekledik. Eksi sayıları da kullanabilirsiniz.

6. Cihaz Erişimi

I-See-Bytes kütüphanesinin en büyük kolaylıklarından biri de cihaz erişimidir. OpenCV gibi resim işleme kütüphaneleri mikrofon veya ses kartı çıkışına erişim izni vermezken ses işleme için yazılmış özel kütüphaneler ise kamera ve LIDAR gibi bilgisayara bağlanarak çalışmak için tasarlanmış cihazlara erişim vermezler. I-See-Bytes kütüphanesinin profesyonel versiyonu Ethernet üzerinden IP kameralara, LIDAR'lara ve hatta robot kolu kontrol devrelerine erişme yeteneğine sahiptir. Ancak öğrenci versiyonunda bu yetenekler ses kartı giriş/çıkışı ve USB kameralara (Direct-X sürücüleri olan IP kameralar dahil) erişim ile sınırlıdır. Bu bölümde, cihaz erişimi içeren uygulamaların nasıl yazıldığını öğreneceğiz.

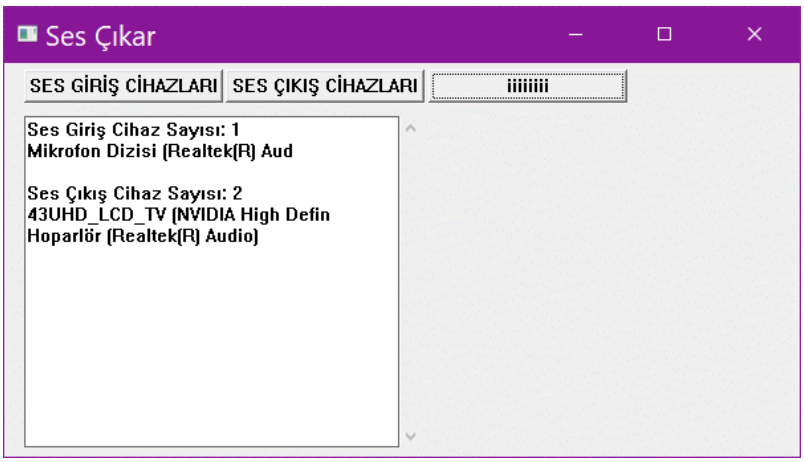
6. 1. Ses Çıkışı

Birinci bölümde I-See_Bytes kütüphanesinde iki temel veri tipi olduğunu ve bunlardan birincisinin verileri tutmak için kullandığımız ve kütüphaneye ismini veren ICBYTES, diğerinin ise cihazlara erişim için kullanacağımız ICDEVICE olduğundan bahsetmiştik. Bu bölümde ICDEVICE akışkanının nasıl kullanılacağını göreceğiz.

Cihaz erişiminin ilk adımı bilgisayara bağlı olan cihazların sayısının ve özelliklerinin sorgulanmasıdır. Ancak maalesef birçok kütüphane bu özelliği sağlayamamaktadır. Örneğin OpenCV kütüphanesi bilgisayarınıza takılı kameraların sayısını bile size söyleyemez. Oysa I-See-Bytes kütüphanesi, diğer birçok kütüphaneye göre çok gelişmiş sistem yoklama fonksiyonlarına sahiptir. Örneğin aşağıdaki komutlar size sadece kaç tane ses çıkış cihazınız olduğunu söylemekle kalmaz, onların marka ve özelliklerini de belirtir:

```
ICBYTES cikis;
int sayi=WaveOutputDevices(cikis);
```

WaveOutputDevices() adlı fonksiyonun Türkçe manası “Dalga Çıkış Cihazları” manasına gelmektedir. Burada dalgadan kasıt ses dalgası sinyalidir. Ancak insan kulağının duyma eşiğinin ötesinde dalgalar da yaratılabileceği için sadece dalga denmiştir. Bu fonksiyon girdi olarak ICBYTES cinsinden bir akışkan bekler. Diyelim ki bilgisayarınızın donanımına bağlı ses çıkışı özelliklerine sahip iki cihaz varsa bu durumda 32x2 boyutlarında karakter cinsinden bir matris yaratıp her satırına bunların markalarını ve modellerini yazar. Aşağıda ekran görüntüsü verilen uygulamada, o anda çalıştığı bilgisayara bağlı olan ses giriş ve çıkış cihazlarının sayısı ve isimleri gösterilmektedir.



Soldan birinci butona basıldığında dalga giriş cihazlarının sayısı ve isimleri ekrana yazılmakta, ortadaki butona basıldığında ise dalga çıkış cihazlarının sayısı ve isimleri yazılmaktadır. Bu çıktıdan anlaşılmaktadır ki bu bilgisayarın kendi hoparlörü Realtek cinsinden bir ses çipine bağlıdır. Yine bilgisayar büyük ihtimalle HDMI kablo ile 43 inçlik UHD bir ekrana yada televizyona bağlıdır ve onun ses sistemine de erişebilmektedir. Daha da önemlisi 43 inçlik ekran şu anda demirbaş ses arabirimi olarak ayarlanmıştır. Ortadaki “SES ÇIKIŞ CİHAZLARI” butonuna basıldığında çağrılan fonksiyon aşağıda gösterilmiştir:

```
void CikisCihazlar()
{
  ICBYTES cikis;
  ICG_printf(MLE, "Ses Çıkış Cihaz Sayısı: %d\n",WaveOutputDevices(cikis));
  Print(MLE, cikis);
  ICG_printf(MLE, "\n");
}
```

Burada “MLE” yukarıdaki resimde içine cihaz bilgilerinin yazdırıldığı çok satırlı metin kutucuğunun kulpudur. Bu uygulamanın tüm kodu aşağıda gösterilmiştir.

```
#include"icb_gui.h"

int MLE;
unsigned char ii[67] ={139,131,127,108,108,96,67,84,54,31,40,18,9,11,5,11,12,
30,37,54,74,91,103,127,139,147,164,176,194,202,218,230,230,244,250,248,237,
226,225,201,182,171,139,117,98,74,56,37,26,24,9,9,11,16,24,37,43,58,75,84,84,
103,108, 122,124,138,141 };

void ICGUI_Create()
{
  ICG_MWTitle("Ses Çıkar");
  ICG_MWSize(620, 350);
}

void GirisCihazlar()
{
  ICBYTES girdi;
  ICG_printf(MLE, "Ses Giriş Cihaz Sayısı: %d\n", WaveInputDevices(girdi));
  Print(MLE, girdi);
  ICG_printf(MLE, "\n");
}

void CikisCihazlar()
{
  ICBYTES cikis;
  ICG_printf(MLE, "Ses Çıkış Cihaz Sayısı: %d\n",WaveOutputDevices(cikis));
  Print(MLE, cikis);
  ICG_printf(MLE, "\n");
}

void iiiii()
{
  ICDEVICE d;
  ICBYTES i;
  if (CreateSound(i, 1, 3000, ICB_UCHAR, 8000) == 0)
  {
    for (int y = 0; y < 3000; y++) i.B(1, y) = ii[y % 61];
  }
  if (!CreateCompatibleDevice(d, ICB_WAVEOUT, 0, i))
    ICG_printf(MLE, "Ses Kartı Erişilemedi!\n");
  if(!WaveOut(d, i))
    ICG_printf(MLE, "Cihaz bu sesi Çıkartmıyor!\n");
  Sleep(400);
  WaveOut(d, i);
  CloseDevice(d);
}

void ICGUI_main()
{
  ICG_Button(15, 5, 150, 25, "SES GİRİŞ CİHAZLARI", GirisCihazlar);
  ICG_Button(167, 5, 150, 25, "SES ÇIKIŞ CİHAZLARI", CikisCihazlar);
  ICG_Button(320, 5, 150, 25, "iiiiiiii", iiiii);
  MLE = ICG_MLEdit(15, 40, 300, 250, "", SCROLLBAR_V);
}
```

Kaynak kodundan anlaşıldığı üzere bu uygulama, bilgisayara bağlı ses giriş/çıkış cihazlarını sorgulamak dışında asıl amacı "iiii" sesini çıkartmaktır. Bunu yapabilmek için kodun baş tarafında bu sesin sayısal verisini içeren dalga bilgisi "`unsigned char ii[67]`" dizini içerisine yazılmaktadır. Daha sonra sesi çıkartan butona basıldığında çağrılan iiiii() adlı fonksiyon ilk kez karşılaştığımız bir fonksiyon çağırmaktadır:

```
CreateSound(i, 1, 3000, ICB_UCHAR, 8000)
```

Bu fonksiyon aslında 1x3000 büyüklüğünde bayt cinsinden bir matris yaratmaktadır. Ancak bu matris (vektör) ses verisi taşıdığı için aynı resim taşıyan matris yaratırken CreateImage() fonksiyonu kullandığımız gibi matrisle birlikte özel bazı veri yapıları daha oluşturmaktadır. Böylelikle kullanıcı bu sesi dinlemek istediğinde bu özel veri yapıları hazır olduğu için sesi çalma işlemi daha hızlı gerçekleşmektedir. Ayrıca CreateSound() fonksiyonu CreateMatrix() veya CreateImage() fonksiyonlarından farklı olarak fazladan bir girdi beklemektedir. Bu da en sonda yer alan "8000" sayısıdır. Bu sayı ses verisinin hangi hızda çalınacağını ayarlar. Yani ses kartının her saniye bu matristeki verilerin 8000 adetini kullanacağını belirtir. Bizim matrisimiz 3000 uzunluğunda olduğuna göre bu demek oluyor ki bu matris ancak 3/8 saniye = 375 milisaniye boyunca ses çıkartabilecektir.

Dikkat edilirse en tepedeki "ii[67]" dizini sadece 67 eleman uzunluğundadır. Ancak neyse ki çıkarttığı ses periyodik, yani kendini tekrar eden bir sestir. Dolayısı ile 67 elemandan oluşan bu dizini 3000 elemandan oluşan matrisimizin (aslında vektörümüzün demek daha doğru olur) içerisine defalarca yazarsak "iiiii" sesinin duyulma süresini uzatabiliriz. İşte arkadan gelen for döngüsü bunu yapmaktadır.

Daha sonra yine yeni bir fonksiyon ile karşılaşmaktayız:

```
ICDEVICE d;
ICBYTES i;
CreateCompatibleDevice(d, ICB_WAVEOUT, 0, i)
```

Bu fonksiyon, sıfır numaralı dalga çıkış cihazına ulaşmak istediğimizi ve "i" matrisi içerisinde sayısal verileri bulunan ses dalgasını bu cihazda çalmak istediğimiz söylemektedir. Burada hatırlamamız gereken i matrisi içinde mono, yani tek kanallı (çift kanallı olsa stereo olurdu), bayt cinsinden verilerin bulunduğu ve bu verilerin saniyede 8000 tanesinin çalınması gerektiğidir. Dolayısı ile bize bunları yapabilecek bir donanım gerekmektedir. Eğer donanım bu veriyi çalabilecek durumda ise "d" akışkanı içerisinde bu donanıma erişim sağlayan bir veri yapısı oluşturulur ve donanım kullanılmaya hazır hale gelir. Bu

durumda CreateCompatibleDevice() (Uyumlu Cihaz Yarat) fonksiyonu `true` döndürür. Eğer olumlu dönüş olduysa bundan sonra tek yapmamız gereken WaveOut() fonksiyonunu istediğimiz kez çağırıp "i" matrisini buna yollayarak matris içindeki sesi çalmak olacaktır. İşimiz bittiğinde ise CloseDevice(d) komutu ile "d" akışkanı içerisinde yaratılmış olan cihaza erişim verilerini silebiliriz.

Eğer stereo ses çıkarmak isteseydik, matrisimizi 1x3000 boyutları yerine 2x3000 yaratmamız gerekirdi:

```
CreateSound(i, 2, 3000, ICB_UCHAR, 8000);
```

Sonra da for döngüsü içerisinde i.B(1,y) ile birlikte i.B(2,y)'nin de içine yazmamız gerekirdi.

6. 2. Ses Dosyası Okuma

Windows işletim sisteminde IBM ve Microsoft'un birlikte geliştirdiği "Waveform Audio Format" kelimelerinin kısaltılmış hali olan WAVE yada WAV adlı ses kayıt formatı kullanılır. Bu format özellikle sıkıştırılmamış ses dosyaları için AIFF formatı ile birlikte en çok kullanılan iki formattan biridir. Windows işletim sisteminin açılışta veya hata yaptığınızda veya başka herhangi bir nedenle çıkarttığı seslerin tümü ".wav" formatındaki dosyalarda tutulur. Bu dosyalara C:\Windows\Media dosyolundan ulaşabilirsiniz. Bu formattaki ses dosyalarını matrise yüklemek için tek yapmanız gereken ReadWave() fonksiyonunu çağırmaktır:

```
ICBYTES A;
ReadWave(A, "C:/Windows/Media/tada.wav");
```

Eğer bu fonksiyon dosyayı başarılı bir şekilde okuduysa A akışkanı içerisindeki ses matrisini çalmak için kolaylık olsun diye WaveOut() fonksiyonunun diğer versiyonunu kullanabilirsiniz:

```
WaveOut(A, 0);
```

Bu fonksiyon, tek satırda A matrisinin taşıdığı ses verisinin sıfır numaralı yani demirbaş ses çıkış cihazı tarafından çalınmasını sağlar. Bu arada "tada.wav" dosyasını çalmadan evvel şu komutu uygularsanız:

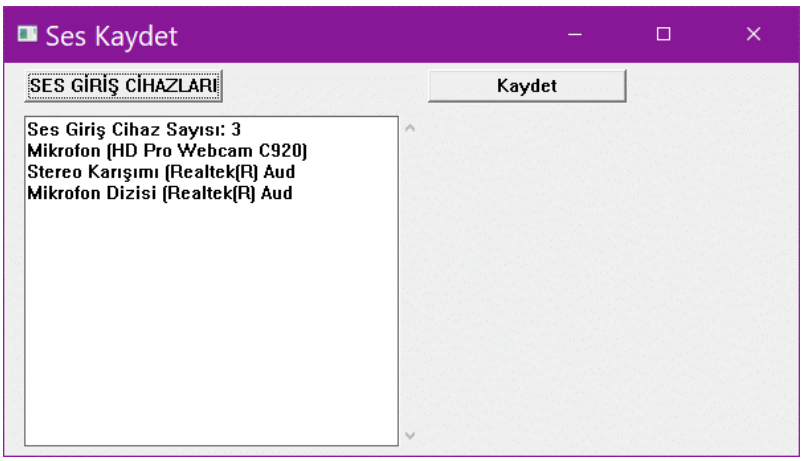
```
A*=6;
```

Ses düzeyinin bir hayli yükseldiğini duyacaksınız. Burada ses verisini 6 ile çarparak sesin genliğini arttırmış olduk. Eğer 15 ile çarpsaydık seste bozulmalar olduğunu

(distorsiyon) duyardınız çünkü matrisin kullandığı değişken tipinin taşıyamayacağı büyüklükte sayılara ulaşmış olurduk.

6. 3. Ses Kaydı

Bir matris ya da vektör içerisine mikrofondan veya başka bir sinyal kayıt hattından ses kaydı yapmak istiyorsak bir önceki bölümde yaptıklarımızın çok benzeri adımları tekrar etmemiz gerekmektedir. Öncelikle ses ya da sinyal kaydedeceğimiz cihazı belirlememiz gerekir. Bunun için bilgisayarımıza bağlı olan ses/sinyal giriş cihazlarını listelemeliyiz:



Geçen seferkinden farklı olarak bu kez bilgisayarımıza bağlı üç adet giriş cihazı görüyoruz. Bu kez sıfır numaralı, yani demirbaş sinyal giriş cihazımızın Logitech firmasına ait HD PRO Webcam C920 model bir web kamerasının mikrofonu olduğunu görüyoruz. Bir numaralı giriş cihazı ise bir Stereo Mixer, ve son sırada yani iki numarada bilgisayarımızın ses kartının mikrofon girişi olduğunu görmekteyiz. Aşağıda gösterilen ve "Kaydet" butonuna tıkladığımızda çağrılan fonksiyon, web kameranın mikrofonundan iki saniye boyunca saniyede 8000 örnek alma hızında bir kayıt alıp sonra da bunu bize geri dinletmektedir:

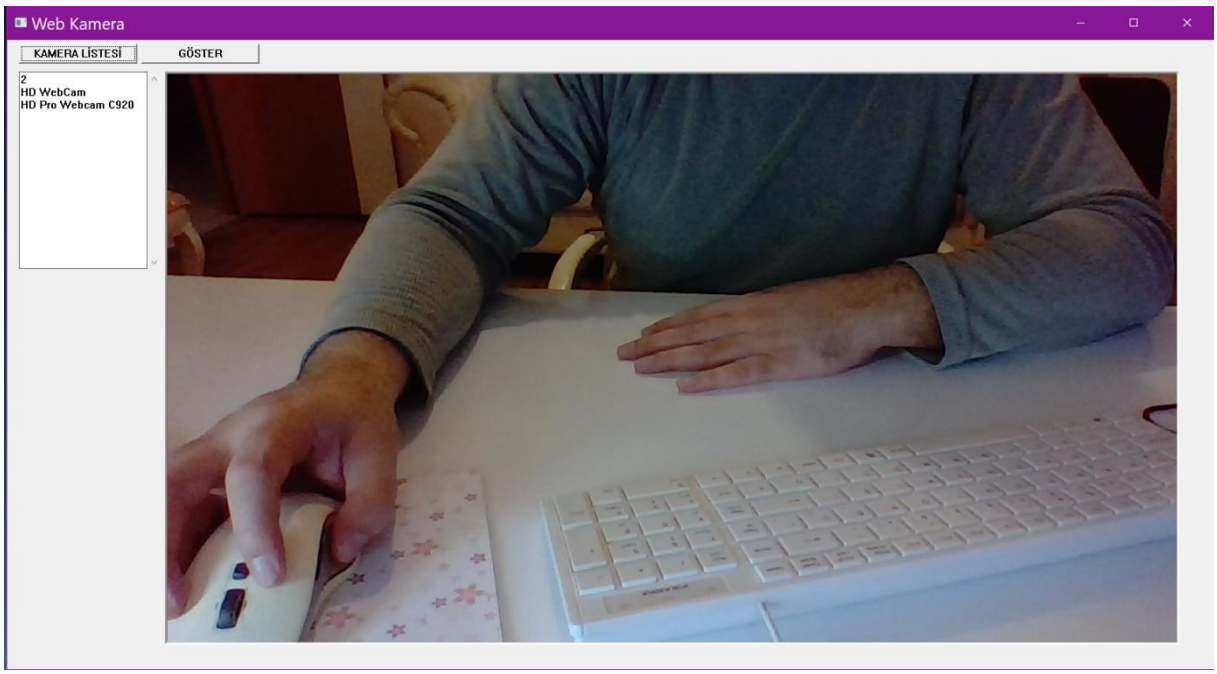
```
void Kaydet()
{
  ICDEVICE d;
  ICBYTES i;
  CreateSound(i, 1, 16000, ICB_UCHAR, 8000);
  if(CreateCompatibleDevice(d, ICB_WAVEIN,0,i))
    ICG_printf(MLE,"Kayıt Başladı\n");
  WaveIn(d, i);
  ICG_printf(MLE, "Kayıt Bitti\n");
  WaveOut(i, 0);
}
```

Bir önceki kısımda anlatıldığı gibi öncelikle CreateSound() fonksiyonu kullanarak bir vektör yarattık. Geçen seferden farklı olarak bu kez vektörümüzü 3000 uzunlukta değil, 16000 uzunlukta oluşturduk. Bunun nedeni iki saniye sürecek bir kayıt almak istememizdir. Eğer mikrofonun saniyede 8000 sayısal örnek almasını istiyorsak iki saniye için 16000 sayıyı tutabilecek uzunlukta bir vektöre ihtiyaç duyarız. Daha sonra CreateCompatibleDevice() fonksiyonunu tekrar kullandık ancak bu kez ikinci parametreyi "`ICB_WAVEIN`" olarak verdik. Bu parametre fonksiyona bir giriş cihazı yaratmak istediğimizi iletti. Bir önceki bölümde dikkat ederseniz bu parametre "`ICB_WAVEOUT`" idi.

Sonra WaveIn() fonksiyonunu çağırarak ses kaydını "i" vektörünün içerisine yerleştirdik. Artık WaveOut() fonksiyonu ile kaydettiğimiz sesi dinlemekten başka yapacak bir şey kalmadı. Bu biki bölümde I-See-Bytes kütüphanesi ile ses giriş çıkışlarına ulaşmanın ne kadar kolay olduğunu öğrenmiş olduk.

6. 4. Kamera Erişimi

Aşağıda gösterilen uygulamada bilgisayarımıza bağlı olan web kamera sayısı, model isimleri ve kameradan alınan görüntü gösterilmektedir.



Metin kutusundan anlaşıldığına göre bu bilgisayara iki kamera bağlıdır. Bunlardan sıfır numaralı demirbaş olanı genel bir isme sahip "HD Webcam" olarak isimlendirilmiştir. Bu tür isimler genellikle bir notebook veya tablet

bilgisayar üzerinde gelen web kameralarını tanımlamak için kullanılmaktadır. Bu tanımdan, kullanılmakta olan bilgisayarın bir laptop olduğunu tahmin edebiliriz. İkinci kamera ise Logitech firmasına ait HD PRO Webcam C920 modeli olduğunu anlamaktayız. Bu uygulamanın kaynak kodu aşağıda gösterilmiştir.

```
#include"icb_gui.h"

int MLE,FRM;

void ICGUI_Create()
{
  ICG_MWTitle("Web Kamera");
  ICG_MWSize(1550, 850);
}

void Kameralar()
{
  ICBYTES kamera_listesi;
  ICG_printf(MLE,"%d\n",GetVideoSourceList(kamera_listesi));
  Print(MLE, kamera_listesi);
}

void Goster()
{
  ICDEVICE d;
  ICBYTES i;
  CreateDXCam(d, 0);
  for (int x = 0; x < 100; x++)
  {
    CaptureImage(d, i);
    DisplayImage(FRM, i);
    Sleep(30);
  }
  CloseDevice(d);
}

void ICGUI_main()
{
  ICG_Button(15, 5, 150, 25, "KAMERA LİSTESİ", Kameralar);
  ICG_Button(170, 5, 150, 25, "GÖSTER", Goster);
  MLE = ICG_MLEdit(15, 40, 180, 250, "", SCROLLBAR_V);
  FRM = ICG_FrameDeep(200, 40, 1024, 1024);
}
```

Bu uygulamada "KAMERA LİSTESİ" butonuna basıldığında GetVideoSourceList() adlı fonksiyonun çağrıldığını görmekteyiz. Bu fonksiyon girdi olarak bir ICBYTES akışkanı alıyor. Fonksiyon bu akışkan içerisine 32xkamera sayısı büyüklüğünde karakter bir matris yerleştiriyor ve bu matrisin her satırına sisteme bağlı olan kameralardan birinin ismini yazıyor. Ayrıca fonksiyonun kendisi de bir tam sayı döndürüyor ki bu tam sayı da bize kamera sayısını söylüyor. Daha sonra biz fonksiyona girdi olarak verdiğimiz ICBYTES akışkanını (kamera_listesi) ekrana yazdırdığımızda bilgisayara o anda bağlı olan iki kameranın modelini görüyoruz.

Diğer buton olan GÖSTER'e bastığımızda 3 saniye boyunca kameradan gelen görüntünün ekrana yansıtıldığını görüyoruz. Bunu gerçekleştiren fonksiyonu incelediğimizde cihaz yaratmak için kullanılan yeni bir fonksiyon ile karşılaşıyoruz:

```
ICDEVICE d;
ICBYTES i;
CreateDXCam(d, 0);
```

Bu fonksiyon sıfır numaralı kamerayı bağlanmamızı sağlayan ICDEVICE türünden bir cihaz yapısı yaratıyor. Bu fonksiyonun ismi içerisindeki DXCam terimi "DirectX Kamera'nın" kısaltılmasından geliyor. Bunun manası şu demek:

Bilgisayarınıza herhangi bir donanım hattından bir kamera bağlı ise, ki bu yollar USB, Ethernet, FireWire vs gibi farklı standartlar olabilir; eğer bu kamera DirectX sürücüleri üzerinden işletim sistemine tanıtıldıysa, CreateDXCam() fonksiyonu ile buna erişebilirsiniz.

Yani USB kameralar yanında eğer DirectX sürücüleri varsa, network kameralarına da bağlanabilirsiniz. I-See-Bytes kütüphanesinin öğrenci sürümü sadece DirectX üzerinden kameralara erişim imkanı sağlamaktadır. Eğer IP, yani ağ kameralarına RTSP protokolü üzerinden erişmek isterseniz bu kütüphanenin daha yüksek versiyonlarını (mesela **Profesyonel** versiyonu) kullanmalısınız.

Göster fonksiyonunun devamında bir başka komut daha ilk kez karşımıza çıkıyor:

```
CaptureImage(d, i);
```

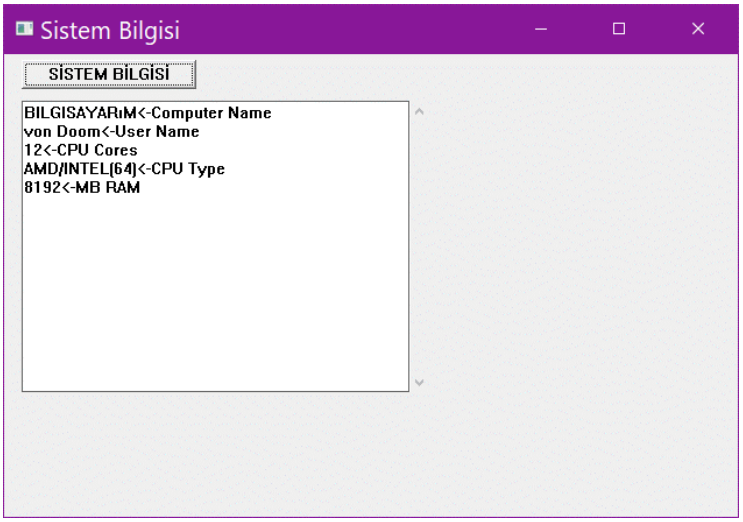
Adından da anlaşılacağı üzere bu fonksiyon "d" cihazından, ki bu durumda kamera oluyor, "i" akışkanı içerisine bir resim yakalıyor. Bu resim kameranın o anda baktığı görüntüdür. Sonra da bu görüntü ekrana basılıyor. Bu işlem her 30

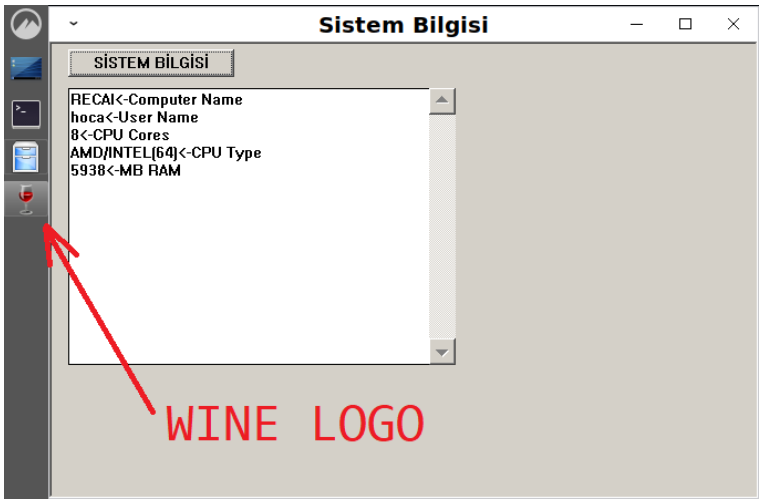
milisaniyede bir 100 kez tekrarlanınca 3 saniyelik canlı bir görüntü ekrana yansıtılmış oluyor.

6. 5. Sistem Bilgisi Sorgulama

Bazı uygulamaların üzerinde çalıştıkları bilgisayarın donanımı hakkında bilgi sahibi olmaları gerekir. Örneğin bilgisayardaki işlemcinin türü veya kaç çekirdeğe sahip olduğu gibi. Aşağıda gösterilen uygulama, sistem hakkındaki şu bilgileri sıra ile listeler:

- Bilgisayar İsmi
- Kullanıcı İsmi
- İşlemci çekirdek sayısı
- İşlemci Türü
- RAM yani Hafıza miktarı





En üstte gösterilen resim bu uygulamanın Windows işletim sisteminde çalıştırılmasıyla elde edilmiştir. Alttaki ise aynı executable'ın, yani derlenmiş kodun 64 bitlik Debian Linux üzerinde çalışması ile elde edilmiştir. Sol tarafta görülen Linux araç çubuğunda programımızın ikonu WINE logosu olan şarap kadehi ile gösterilmektedir. Daha evvel bahsettiğimiz gibi, I-See-Bytes kütüphanesi kullanarak derlediğiniz programların ikili kodu, yani .EXE dosyası hem windows üzerinde hem de WINE yüklenmiş Linux bilgisayarlar üzerinde çoğu zaman sorunsuz bir şekilde çalışmaktadır.

WINE bir emulatör, simülatör veya "virtual machine" değildir (www.winehq.org). Bir uyumluluk katmanıdır. Yani windows sistem çağrılarını Linux sistem çağrılarına tercüme eden bir katmandır. Dolayısı ile Java'ya veya Python'a göre muazzam hızlı çalışır. Programlarınızın Windows işletim sisteminde çalıştığı hızla hemen hemen aynı hızlarda çalıştığını gözlemlersiniz. Aradaki tek fark, ilk çalıştırma anında programın yüklenmesi, yani ekranda belirmesi 4-5 saniye geç olabilir. WINE uyumluluk katmanı her Linux dağıtımı (Ubuntu, Red Hat, Pardus vs) için yazılmaktadır. Yüklemesi çok kolaydır. Hatta birçok Linux çeşidi WINE'ı demirbaş olarak yüklenmiş halde sunar. Bunun sunulmadığı Linux distroları için ise manuel olarak yüklemek çok kolaylaştırılmıştır. Bu demek oluyor ki Java'nın çok övünerek sunduğu platformlardan bağımsız olma özelliği I-See-Bytes tarafından da sunulmaktadır. Üstelik de Java'ya göre en az 10 kat hızlı çalışan programlar üreterek.

WINE programı artık Android işletim sistemi için de üretilmektedir. Ancak bunun için I-See-Bytes kütüphanesinin ARM işlemcisi için derlenmiş versiyonu gerekecektir. Böyle bir versiyon henüz desteklenmemektedir ancak gelecekte olması mümkündür.

Sistem hakkında bilgi sağlayan bu uygulamanın kodu aşağıda verilmiştir.

```
#include"icb_gui.h"

int MLE;

void ICGUI_Create()
{
  ICG_MWTitle("Sistem Bilgisi");
  ICG_MWSize(650, 450);
}
```

```
void Bilgi()
{
  ICBYTES info;
  SystemInfo(info);
  Print(MLE,info);
}

void ICGUI_main()
{
  ICG_Button(15, 5, 150, 25, "SİSTEM BİLGİSİ", Bilgi);
  MLE = ICG_MLEdit(15, 40, 350, 250, "", SCROLLBAR_V);
}
```

7. İleri Düzey Arabirim Teknikleri

Şu ana dek anlatılan özellikler kullanılarak çok farklı çeşitte, hatta profesyonel programlar yazmak mümkündür. Örneğin, nasıl çalıştığı biliniyorsa bir muhasebe programı veya bir veri tabanı ile birleştirilerek bir stok kayıt programı yazılabilir. Ancak bazı tür yazılımlar vardır ki alışageldiğimiz programlama teknikleri yeterli olmaz. Bunun sebebi Windows işletim sisteminde meydana gelen her türlü olayın programınıza bir mesajla iletilmesidir. Yani kullanıcı klavyedeki bir tuşa her bastığında, fare her oynatıldığında yazdığınız programa bir mesaj gelir. Programınız bu mesajları sürekli işleyerek ve yorumlayarak çalışır. Bu mesaj işleme maratonuna bir an bile ara verirseniz programınızın donduğunu görürsünüz. Peki bu mesajlar nerede işlenmektedir?

Maalesef bu mesajları işlemek çok karmaşık ve uzun programlar gerektirir. Her biri farklı amaçlar ve parametreler içeren bu mesajları işleyebilen programlar yazmak Windows işletim sisteminin en alt programlama arabirimi olan WIN32 API'ı çok iyi bilmeyi gerektirir. İşte I-See-Bytes kütüphanesinin en büyük faydalarından biri bu karmaşık işlemleri kendisinin halledip programcıya sade, basit ve mantıklı bir dizi fonksiyon sağlayarak mesajları istediği gibi değerlendirmesine olanak sağlamasıdır.

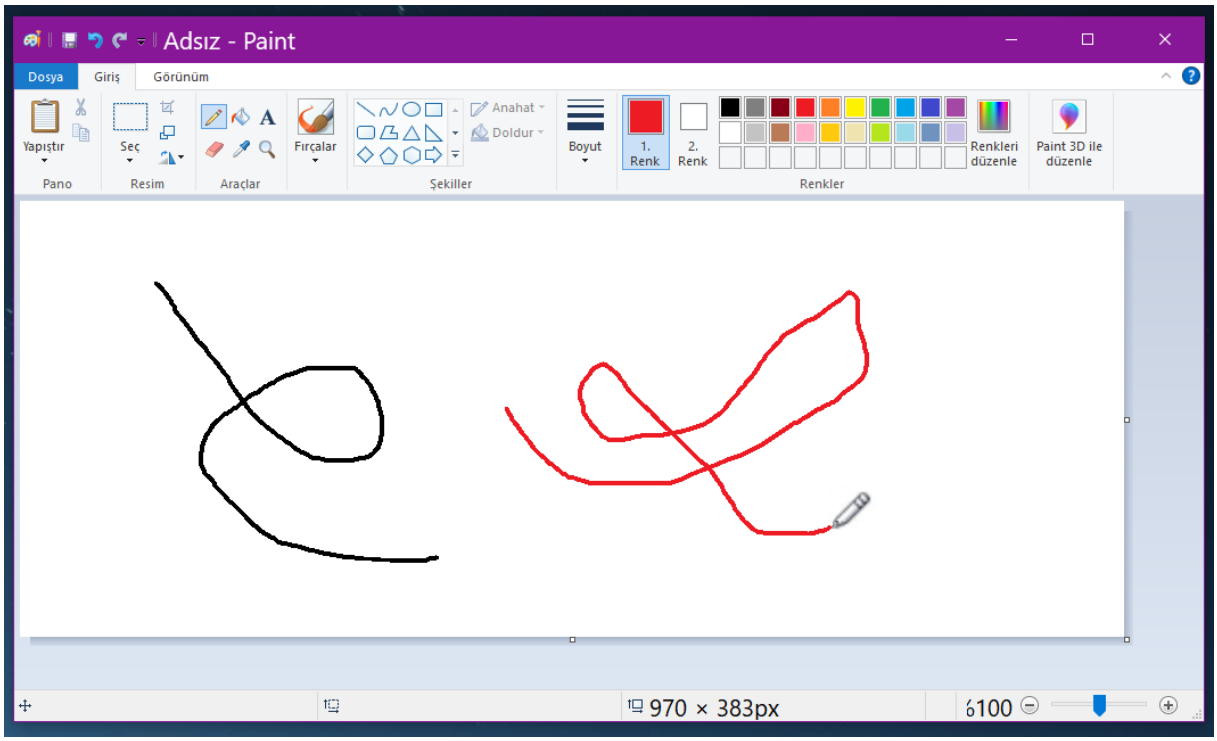
Windows işletim sistemindeki bu mesaj yapısının en büyük zorluğu uzun süren bir program yazdığımızda ortaya çıkar. Örneğin bir oyun programı bir başladığında sürekli olarak ekrana animasyonlar çizdirir. Eğer şu ana kadar öğrendiğimiz tekniklerle 5-10 dakika süren bir animasyon programı yazarsanız, bu program çalıştığında, bitinceye kadar dışarıdan hiçbir tepki alamadığını görürsünüz. Mesela pencereyi yerinden oynatamaz, minimize edemez ve hatta kapatamazsınız. Çünkü kodunuz sürekli olarak ekrana bir şeyler çizdirirken

işletim sisteminin yolladığı mesajları işleyemez. Animasyonlar veya sürekli resim çizen programların başında oyun programları ve video programları gelir. Ancak bunlarla sınırlı değildir. Animasyonlar artık bilimsel yazılımlarda, matematiksel analizler yapan finansal programlarda ve benzeri birçok yerde kullanılmaktadır. Kullanıcılara daha zengin deneyimler yaşatmak için günümüz uygulamaları görsel ve işitsel birçok öğeyi ekrana sürekli olarak yansıtmayı tercih etmektedir.

İşte bu bölümde bu tür uygulamaları yazabilmek için gerekli teknik erişkinliğe ulaşmaya çalışacağız.

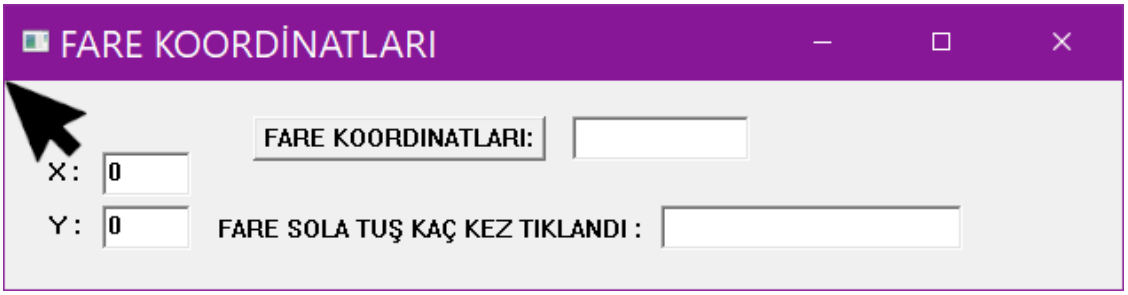
7.1. Fare Parametrelerine Ulaşma

Alışageldiğimiz uygulamaların çoğunda fare (mouse) işletim sistemi tarafından kontrol edilir ve gelen verileri programcının işlemesine gerek yoktur. Program sadece fare bir kutucuğun üzerine gelip de o kutucuk ile etkileşime girdiğinde bundan haberdar olur. Ancak bazı özel uygulamalarda programın farenin konumunu bilip ona göre özel fonksiyonları aktive etmesi gerekir. Buna en bilindik örnek Windows işletim sistemindeki Paint programıdır. Böyle bir uygulama yazmak istiyor olsak, fare imlecinin koordinatlarını sürekli olarak bilmemiz gerekir.



I-See-Bytes kütüphanesi içerisinde bu verilere erişmemizi sağlayan mekanizmalar bulunmaktadır.

Aşağıda ekran görüntüleri verilen uygulamada hem fare hareket edince otomatik çağrılan fonksiyonlar, hem de biz istediğimizde farenin koordinatlarını bize veren fonksiyonlar kullanılmıştır. Birinci tip fonksiyonlar fare imleci bizim uygulamamızın pencere sınırları içerisine girdiği anda çağrılmaya başlanır ve fare imlecinin her hareket edişinde yeniden çağrılır. Bu şekilde programımız fare imleci hareket ettiği anda tepki verebilir. Pencerenin en solunda yer alan X ve Y koordinatları değerleri farenin her hareketinde değişir. Bu örnekte fare imlecinin ucu penceremizin sol üst köşesine getirilmiştir. Bu nedenle X ve Y değerleri sıfırdır.



Aşağıdaki örnekte ise fare imlecinin ucu "FARE KOORDINATLARI" yazan butonun sol üst köşesine getirilip sol fare tuşa tıklanarak bu butona basılmıştır. Birazdan göreceğimiz üzere bu butonun sol üst köşesinin koordinatları kaynak kodumuzda (140,20) olarak seçilmiştir. Bu da bize Windows'un sağladığı koordinat bilgisinin %100 tutarlı olmadığını göstermektedir. Butona tıklandığı anda aşağıdaki buton fonksiyonu çağrılmaktadır:


```
void FareKoor()
{
  char ch[16];
  sprintf_s(ch, 16, "%d , %d", ICG_GetMouseX(), ICG_GetMouseY());
  ICG_SetWindowText(SLEF, ch);
}
```

Burada görüldüğü üzere ICG_GetMouseX() ve ICG_GetMouseY() fonksiyonları bize istediğimiz anda farenin X ve Y koordinatlarını döndürebilmektedir.

Fare koordinatlarını anlık olarak sorgulamak için hangi fonksiyonları çağırmak gerektiğini öğrendik. Peki yukarıdaki ekran görüntülerinin sol tarafında gördüğümüz, fare hareket ettirildiğinde kendi kendine otomatik değişen X: ve Y: koordinatları nasıl yaratılmaktadır? Bunu öğrenmek için kodun tamamını gözden geçirmeliyiz.

```
#include"icb_gui.h"
#include<stdio.h>

int SLEX, SLEY,SLEF,SLESAY;

void ICGUI_Create()
{
  ICG_MWTitle("FARE KOORDİNATLARI");
  ICG_MWSize(650, 170);
}

void FareHareketEdince(int x, int y)
{
  char ch[8];
  _itoa_s(x,ch, 8, 10);
  ICG_SetWindowText(SLEX, ch);
  _itoa_s(y, ch, 8, 10);
  ICG_SetWindowText(SLEY, ch);
}

void FareKoor()
{
  char ch[16];
  sprintf_s(ch, 16, "%d , %d", ICG_GetMouseX(), ICG_GetMouseY());
  ICG_SetWindowText(SLEF, ch);
}

void FareSolTiklaninca()
{
  static int say = 0;
  say++;
  char ch[32];
  sprintf_s(ch, 32, "%d kez tıklandı", say);
  ICG_SetWindowText(SLESAY, ch);
}

void ICGUI_main()
{
  ICG_Static(25, 43, 20, 25, "X :");
  ICG_Static(25, 73, 20, 25, "Y :");
  SLEX = ICG_SLEditSunken(55, 40, 50, 25, "");
  SLEY = ICG_SLEditSunken(55, 70, 50, 25, "");
  ICG_Button(140, 20, 165, 25, "FARE KOORDINATLARI:", FareKoor);
  SLEF = ICG_SLEditSunken(320, 20, 100, 25, "");
  ICG_SetOnMouseMove(FareHareketEdince);
```

```
  ICG_Static(120, 75, 240, 25, "FARE SOLA TUŞ KAÇ KEZ TIKLANDI :");
  SLESAY = ICG_SLEditSunken(370, 70, 170, 25, "");
  ICG_SetOnMouseLClick(FareSolTiklaninca);
}
```

Bu kodu incelediğimizde, ICG_Main fonksiyonu içerisinde fare hareket ettiğinde çalıştırılması istenen FareHareketEdince() adlı fonksiyonun kütüphaneye şu şekilde bildirildiğini görüyoruz:

```
ICG_SetOnMouseMove(FareHareketEdince);
```

Bu fonksiyon çağrıldıktan sonra, fare pozisyonu her değiştiğinde FareHareketEdince(int x, int y) adlı fonksiyon otomatik olarak çağrılmaya başlanacaktır. Bu fonksiyonun girdileri fare imlecinin pozisyon koordinatlarıdır. Fonksiyonun içerisinde ise tek yapılan bu koordinatlar metin kutucuklarına yazdırmaktan ibaret olduğunu görmekteyiz.

Aynı şekilde farenin sol tuşuna her tıklandığında çağrılmasını istediğimiz bir fonksiyon varsa bunu da kütüphaneye ICG_SetOnMouseLClick() fonksiyonu kullanarak belirtiyoruz:

```
ICG_SetOnMouseLClick(FareSolTiklaninca);
```

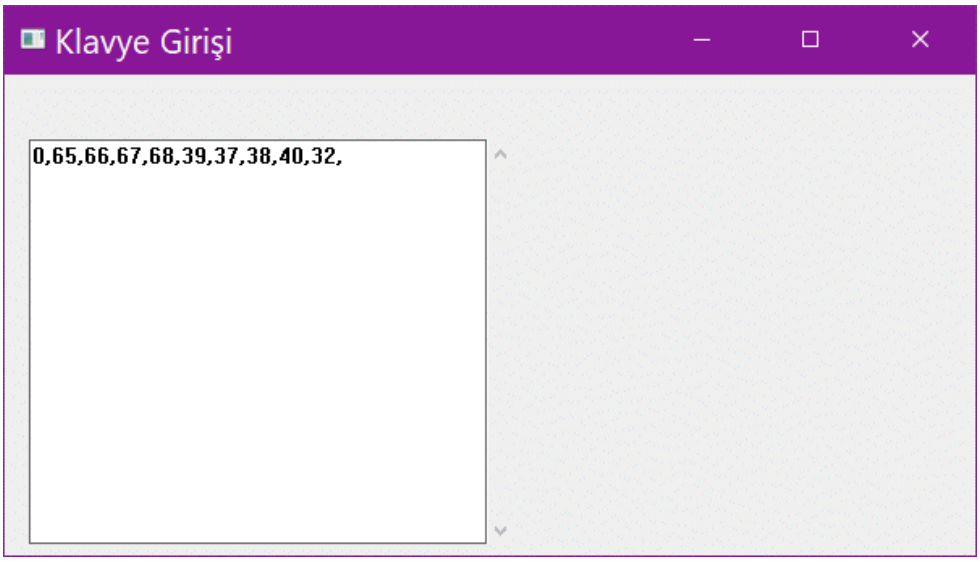
FareSolTiklaninca() fonksiyonun kodunu irdelediğimizde bu kodun tek yaptığının farenin kaç kez tıklandığını sayıp bunu **SLESAY** kulpuna bağlı metin kutucuğu içerisine yazdırmaktan ibaret olduğunu görmekteyiz. Farenin sol tuşuna beş kez tıklandıktan sonra, imlecin **SLEF** kulpuna bağlı metin kutucuğunun sağ üst köşesine geldiği ekran görüntüsü aşağıda verilmiştir. Bu metin kutucuğunun sol üst köşesinin (320,20) koordinatında yaratıldığını ve genişliğinin 120 piksel olduğunu düşünürsek, sağ üst köşesinin de (420,20) koordinatlarında olması gerektiğini hesaplayabiliriz. Bu da X: ve Y: kutucuklarında okuduğumuz değerle hemen hemen aynıdır.

7.2. Klavye Okuma

Programcıların endişelenecekleri en son şeylerden biri klavyeden basılan tuşların ne olduğunu anlamaya çalışmaktır. Bunu zaten işletim sistemi bizim için yapmaktadır. Bir metin kutucuğu yarattığımızda bunun içine harfler yazılması işletim sistemi tarafından gerçekleştirilen temel görevlerden biridir. Programcının tek yapması gereken metin kutucuğuna yazılı olan kelimeleri bir hafıza alanına kopyalamaktır. Peki ya metin kutucuğu yoksa? Veya metin kutucuğu seçili değilken klavyeden giriş yapılmakta ise?

Bu tür bir programlama gereksinimi en çok oyun programlarken karşımıza çıkmaktadır. Bir oyun programında ekranda herhangi bir metin kutucuğu olmamasına rağmen oyun kontrollerini sağlamak, sağa sola ateş etmek için kullanıcı klavye tuşlarına basmaya devam eder. İşte bu bölümde, klavyeden gelen bu bilgilerin nasıl işleneceğini göreceğiz.

Aşağıdaki uygulamayı ilk çalıştırdığınızda metin kutucuğu içerisinde "0," yazılı olduğunu göreceksiniz. Daha sonra klavyede sırasıyla A, B, C, D sağ ok, sol ok, yukarı ok, aşağı ok ve boşluk tuşlarına basarsanız, aşağıdaki görüntünün aynısını elde edersiniz. Burada dikkat çekmemiz gereken husus, klavyedeki her tuşun her zaman aynı sayıyı yollayacak olmasıdır. Yani büyük harf a ile küçük harf A aynı sayıyı döndürecektir (65). Diğer bir deyişle, "Caps Lock" tuşunun basılı olması bir şey değiştirmez. Ancak aynısını numerik pad (varsa) söyleyemeyiz. Numerik pad, yani çoğu klavyede en sağda bulunan hesap makinası tuşlarına benzer tuşlar, "Num Lock" tuşunun açık ya da kapalı olmasına göre farklı değerler döndürür. Dolayısı ile oyun programınızda bu tuşları kullanacaksanız bu tuşun açık ya da kapalı olma durumları için önlem almalısınız.



Aşağıda bu uygulamanın kaynak kodu verilmiştir. Bu kodda dikkat çeken tek komut ICG_SetOnKeyPressed() komutudur. Bu fonksiyon kullanıcı klavyedeki herhangi bir tuşa bastığında hangi fonksiyonun çağrılacağını tanımlar. Bu fonksiyonun ismi ise bu örnekte KlavyeyeBasinca(int k) olarak seçilmiştir. ICBYTES kütüphanesinin bu fonksiyona tolladığı parametre (k) kalvyede basılan tuşun sayısal kodudur. Bu fonksiyonun tek yaptığı MLE kulbu kullanarak metin kutucuğunun içine bu sayıalrı yazdırmaktan ibarettir.

```
#include"icb_gui.h"

int MLE;

void ICGUI_Create()
{
  ICG_MWTitle("Klavye Girişi");
  ICG_MWSize(620, 350);
}

void KlavyeyeBasinca(int k)
{
 ICG_printf(MLE, "%d,", k);
}

void ICGUI_main()
{
  MLE = ICG_MLEdit(15, 40, 300, 250, "", SCROLLBAR_V);
  ICG_SetOnKeyPressed(KlavyeyeBasinca);
}
```

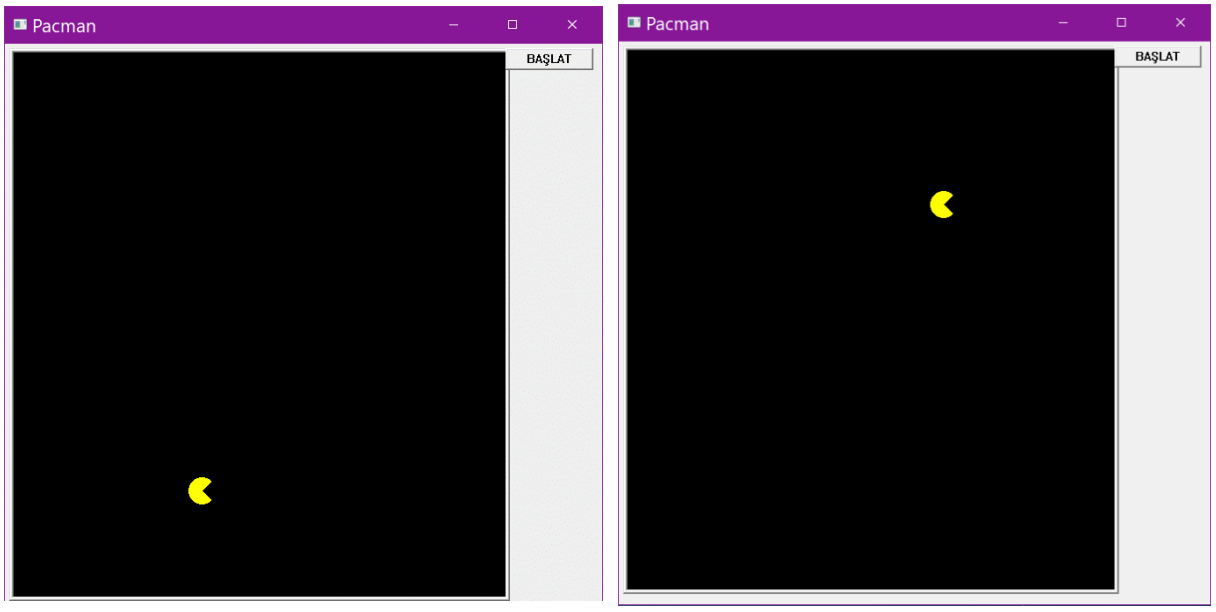
7.3. Pacman'e Devam

Bu kitabın beşinci bölümünde Pacman oyununun grafiklerinin nasıl çizdirildiğini öğrenmiştik. Ayrıca Pacman'i arka arkaya çizip silerek ve kaydırarak kısa bir animasyon yaptırmıştık. Bu başlıkta animasyonu sürekli hale getirmenin yanı sıra, Pacman'in hareketlerini klavyeden kontrol etmesini öğreneceğiz.

Bu bölümde daha evvel de bahsettiğimiz üzere Windows işletim sisteminde uygulamanın donmadan çalışabilmesi için işletim sisteminin yolladığı mesajların sürekli işlenmesi gerekmektedir. Bu nedenle çok uzun süren döngüler kuramayız çünkü bu döngüler çalışırken programımız mesajları işleyemez. Peki böyle bir durumda dakikalar, hatta saatler süren animasyonları nasıl çalıştıracağız? Cevap paralel programlama teknikleri ile. Bu kitabın amacı okuyucuya paralel

programlama tekniklerini öğretmek değildir. Ancak ICBYTES kütüphanesi ile Win32 API içerisinde yer alan paralel programlama teknolojilerinin nasıl birleştirileceğine dair basit bir örnek vermekle yetineceğiz. Her ne kadar basit olsa da verilen bu örneği geliştirerek arzu eden okuyucular Pacman oyununun tamamını programlamayı başarabilirler.

Eğer Pacman'i kaydırarak çizdiren animasyonu kendi asıl (main) programımıza paralel çalışan bir fonksiyon haline getirebilirsek, asıl programımız Windows'dan gelen mesajları işlerken biz de rahat rahat animasyonumuzu yaratabiliriz. İşte içerisinde paralel fonksiyonların aynı anda çalıştığı programlara çok sicimli (multithreaded) programlar diyoruz. Birbirine paralel çalışan her fonksiyona ise parçacık veya sicim (thread) adı verilmektedir. Windows işletim sistemi paralel fonksiyon yaratmak için CreateThread() adlı fonksiyonu kullanır. Paralel olarak çalıştırmak istediğiniz fonksiyonun ismini CreateThread() adlı bu fonksiyona parametre olarak vererek bu fonksiyonu paralel olarak çalıştırmaya başlayabilirsiniz. Biz de programımızda "BAŞLAT" butonuna basıldığında, CreateThread() fonksiyonunu kullanarak animasyonumuzu başlatacağız. Aşağıda uygulamamızın ekran görüntüleri verilmiştir.



Bu uygulamada "BAŞLAT" butonuna basıldıktan sonra, sarışın Pacman ok tuşları kullanılarak sağa ve yukarı doğru hareket ettirilmektedir. Eğer arka plana labirent çizilmiş olsaydı, Pacman'ın labirentte dolaştığı bir animasyon elde edilebilecekti.

Aşağıda bu uygulamanın kaynak kodu gösterilmiştir.

```
#include"icb_gui.h"

void ICGUI_Create()
{
  ICG_MWTitle("Pacman");
  ICG_MWSize(700, 700);
}

int sicim_acik = 0, klavyegirdi = 0;
int FRM1,BTN;

void* Pacman(PVOID lpParam)
{
  ICBYTES saha;
  CreateImage(saha, 560, 620, ICB_UINT);
  int x = 230,y=500;
  while (sicim_acik)
  {
    FillCircle(saha, x - 15, y, 15, 0xffff00);
    HalfRect(saha, x, y, -15, -15, 0);
    HalfRect(saha, x, y, -15, 15, 0);
    DisplayImage(FRM1, saha);
    Sleep(10);
    FillRect(saha, x - 30, y-15, 30, 30, 0);
    if (klavyegirdi == 39)//sağ ok tuşuna basıldıysa
      x++;
    else if (klavyegirdi == 37)//sol ok tuşuna basıldıysa
      x--;
    else if (klavyegirdi == 38)//aşağı ok tuşuna basıldıysa
      y--;
    else if (klavyegirdi == 40)//yukarı ok tuşuna basıldıysa
      y++;
  }
  return(NULL);
}

void KlavyeyeBasinca(int k)
{
  klavyegirdi = k;
}
```

```
void Baslat()
{
  DWORD dw;
  if (sicim_acik == 0)
  {
    sicim_acik = 1;
    ICG_SetWindowText(BTN, "BİTİR");
   CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)Pacman, NULL, 0, &dw);
  }
  else
  {
    sicim_acik = 0;
    ICG_SetWindowText(BTN, "BAŞLAT");
  }
  SetFocus(ICG_GetMainWindow());
}

void ICGUI_main()
{
  BTN=ICG_Button(570, 5, 100, 25, "BAŞLAT", Baslat);
  FRM1 = ICG_FrameMedium(5, 5, 560, 620);
  ICG_SetOnKeyPressed(KlavyeyeBasinca);
}
```

Bu kodun çoğunluğu daha evvel görmüş olduğumuz örnekleri içerdiği için sadece yeni kısımlarına odaklanacağız. Baslat() fonksiyonu içerisinde ilk olarak global olan **sicim_acik** değişkeninin sıfır mı bir mi olduğu kontrol edilmekte, değeri tersine dönüştürülüp "BAŞLAT" butonunun yazısı "BİTİR'e" dönüştürülmektedir. Eğer sıfırdan bire dönüşüyorsa CreateThred() komutu kullanılarak paralel çalışmasını istediğimiz Pacman() adlı fonksiyon başlatılmaktadır.

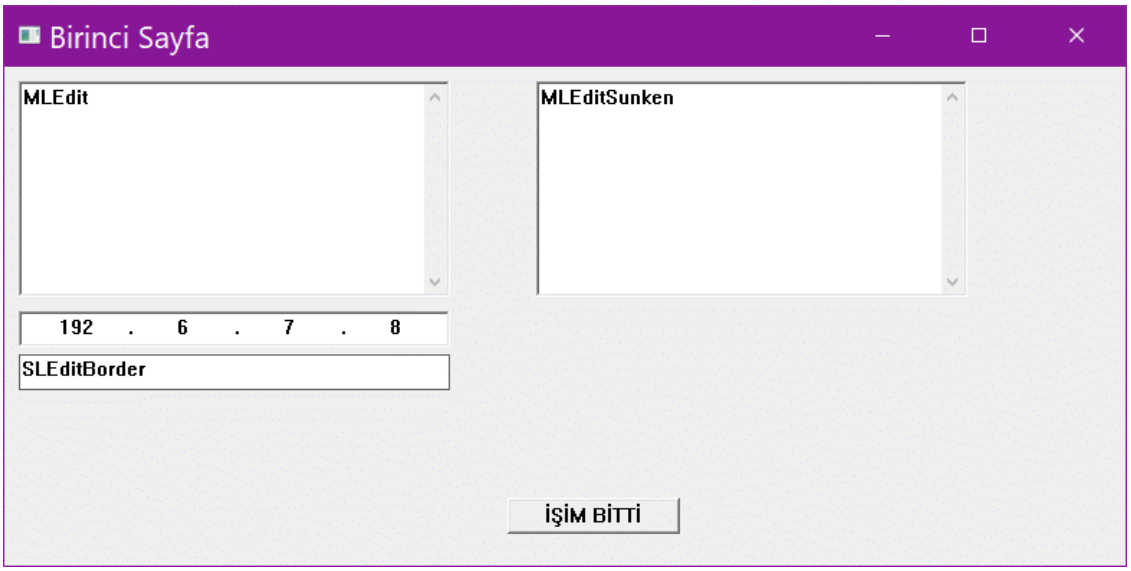
Animasyonun tamamının gerçekleştiği Pacman() fonksiyonuna odaklandığımızda bu fonksiyonun öncelikle üzerinde çizim yapabileceği "saha" adlı bir resim yarattığını görüyoruz. Daha sonra Pacman'in ilk belireceği noktanın koordinatlarını x=230 ve y=500 olarak saptandığı görülüyor.

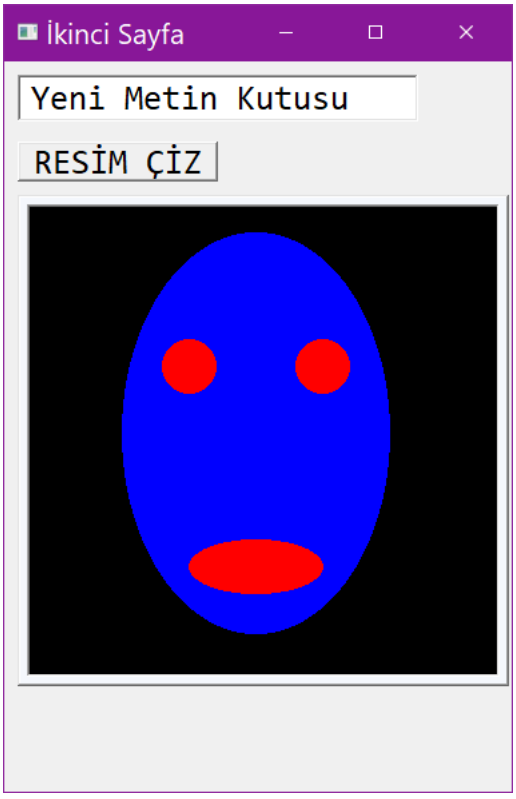
İşin animasyon kısmı sonsuz bir while() döngüsü içerisinde yapılıyor. Bu while döngüsü global **sicim_acik** değişkeni bire eşit olduğu sürece devam edecektir. Döngü içerisinde (x,y) koordinatları kullanılıp Pacman'in önce çizdirildiği, 10 milisaniye sonra da silindiği görülmektedir. Bu kısım beşinci bölümde anlatılanlardan farklı değildir. Tek fark x koordinatının sürekli arttırılması yerine klavyeden girilen son ok tuşuna göre x ve y koordinatlarının arttırılıp azaltılmasıdır. Klavyeden okunan değer sürekli olarak **klavyegirdi** adlı

global değişkene yazdırıldığı için Pacman() fonksiyonu içerisinden de rahatlıkla okunabilmektedir.

7.4. Pencere Dönüştürme

Karmaşık uygulamalarla çalışırken sadece bir pencere yeterli olmayabilir. Bir pencerede işlemler tamamlandıktan sonra ikinci bir pencereye geçiş sağlamak gerekebilir. Böyle durumlarda uygulamanın ana penceresini dönüşüme tabi tutmanız gerekir. Aşağıda böyle uygulamanın iki penceresi gösterilmiştir.





Windows'da farklı pencerelere geçiş yapmanın birden fazla yöntemi vardır. Burada o yöntemlerden en basiti anlatılacaktır. Bu yöntemde birinci pencerede var olan kutucuklarla işimiz bittiğinde bir butona basıp yeni bir pencereye geçmek için birinci penceredeki kutucukların hepsini yok edip sonra ikinci pencerede yeniden yaratılmaktadır. Aşağıda bu uygulamanın kaynak kodu gösterilmiştir.

```
#include"icb_gui.h"
void ICGUI_Create()
{
  ICG_MWSize(800, 400);
  ICG_MWTitle("Birinci Sayfa");
}

int MLE1, MLE2, SLE1, SIP1, BTN1; //ilk sayfa kulpları
int SLE2, FRM1, BTN2; //ikinci sayfa kulpları

void ResimCiz()
{
  ICBYTES resim;
  CreateImage(resim, 350, 350, ICB_UINT);
  FillEllipse(resim, 70, 20, 100, 150, 0xff);
  FillEllipse(resim, 200, 100, 20, 20, 0xff0000);
  FillEllipse(resim, 100, 100, 20, 20, 0xff0000);
  FillEllipse(resim, 120, 250, 50, 20, 0xff0000);
  DisplayImage(FRM1, resim);
}

void YeniEkran()
{
  ICG_DestroyWidget(MLE1);
  ICG_DestroyWidget(MLE2);
  ICG_DestroyWidget(SIP1);
  ICG_DestroyWidget(SLE1);
  ICG_DestroyWidget(BTN1);
  ICG_MWSize(400, 600);
  ICG_MWTitle("İkinci Sayfa");
  ICG_SetFont(30, 0, "Consolas");
  SLE2 = ICG_SLEditSunken(10, 10, 300, 35, "Yeni Metin Kutusu");
  BTN1 = ICG_Button(10, 60, 150, 30, "RESİM ÇİZ", ResimCiz);
  FRM1= ICG_FrameThick(10, 100, 350, 350);
}

void ICGUI_main()
{
  MLE1 = ICG_MLEditSunken(10, 10, 300, 150, "MLEdit", 1);
  MLE2 = ICG_MLEditSunken(370, 10, 300, 150, "MLEditSunken", SCROLLBAR_V);
  SIP1 = ICG_IPAddressSunken(10, 170, 300, 25, 0xc0060708);
  SLE1 = ICG_SLEditBorder(10, 200, 300, 25, "SLEditBorder");
  BTN1 = ICG_Button(350, 300, 120, 25, "İŞİM BİTTİ", YeniEkran);
}
```

Bu kodda görüldüğü üzere, uygulamanın ilk sayfasının tasarımı ICG_main() fonksiyonu içerisinde alışageldiğimiz şekilde yapılmıştır. Farklı olan, "İŞİM BİTTİ" butonu tarafından çağrılan YeniEkran() adlı fonksiyonun içerisinde gerçekleşenlerdir.

Bu fonksiyonda, ICG_DestroyWidget() adlı yeni bir komutla karşılaşmaktayız. Bu komut, ICG_main içerisinde yarattığımız kutucukları yok etmek için kullanılmaktadır. Yani metin düzenleme kutucuklarını ve butonu, onların kulplarını kullanarak yok eder. Böylelikle penceremiz üzerinde yeni baştan kutucuklar yaratıp uygulamamızın çehresini tamamiyle değiştirmemize olanak verir.

Kutucukları yok ettikten sonra, istersek ICGUI_Create() fonksiyonunda kullanmaya alışık olduğumuz uygulama penceresinin boyutlarını değiştiren ICG_MWSize() veya uygulamanın ismini değiştiren ICG_MWTitle() komutları, ve hatta yazı tipini dahi değiştirerek yolumuza devam edebiliriz. İkinci sayfada farklı olarak bir resim çerçevesi yaratılmakta, butona basıldığında ise bu çerçeve içerisine bir resim çizen ResimCiz() fonksiyonu çağrılmaktadır.

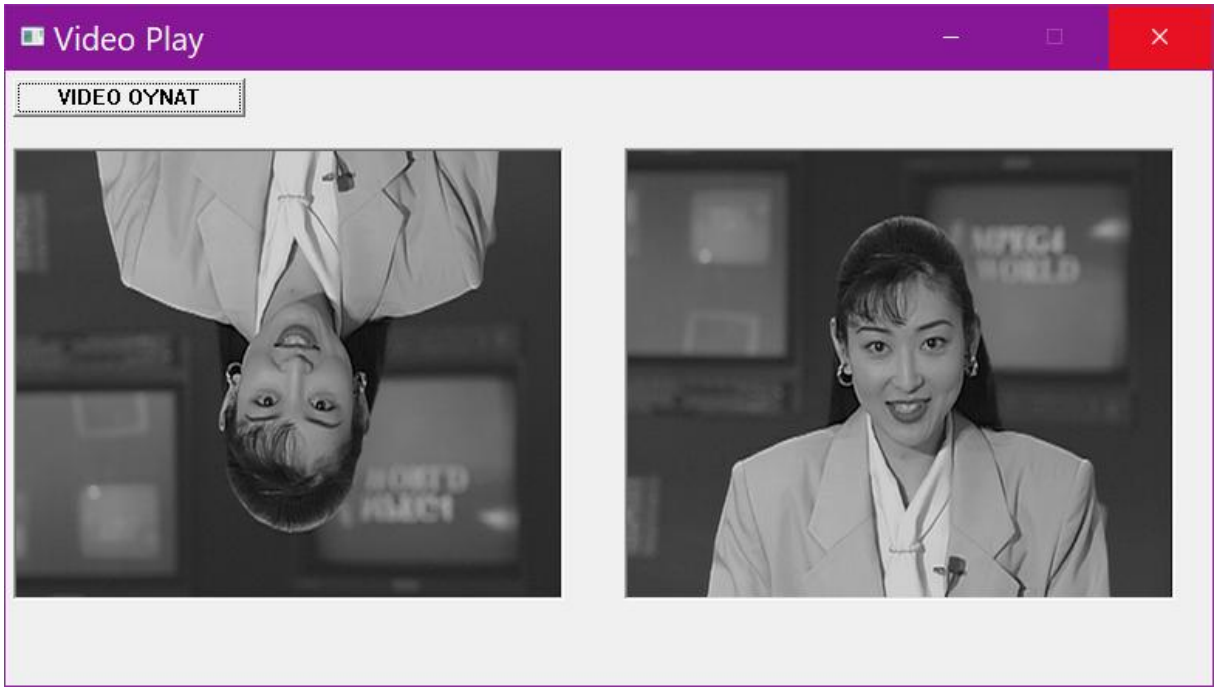
7.5. Video Oynatma

Video işlemeye giriş derslerinde genellikle öğrenciler sıkıştırılmamış video formatları üzerinde çalışırlar. Birçok bilimsel çalışmada da bu tür sıkıştırılmamış video formatları referans veri olarak kullanılır. Bu formatlardan en çok bilineni YUV (.yuv) formatıdır. Bu tür makalelerde sıklıkla kullanılan video örnekleri aşağıdaki web sitesinden indirilebilir.

https://media.xiph.org/video/derf

Bu web sitesindeki videolar ve resimler sayısız bilimsel makalede kullanılmıştır. Sadece-Baytlar-Görüyorum kütüphanesi ile YUV formatındaki videoları açmak çok kolaydır. Bunun için öncelikle bu videolardan birini (mesela akiyo_cif.yuv dosyasını) indirip projenizin çalışma klasörüne kopyalayınız. Sonra CreateFileReader() fonksiyonunu kullanarak bu video dosyasını bir cihaz olarak tanımlayın. Arkasından video dosyasındaki her kareyi birer birer resim matrisi içerisine okuyup ekranda göstermeniz gerekmektedir.

Aşağıda bu işlemleri yapan uygulamamızın ekran görüntüsü verilmiştir.



YUV formatında, her resmin önce parlaklık bilgisi yazılır. CIF formatı 352x288 pixel boyutlarında resimleri adlandırmak için kullanılır. Parlaklık bilgisi olan Y'nin arkasından renk biligileri olan U ve V katmanları gelir. Ancak renk katmanlarından her biri, bu formatta parlaklık bilgisininin dörtte biri kadar, yani 176x144 piksel boyutlarındadır. Videodaki her resim 29.97 milisaniyede bir ekranda gösterilmelidir. Biz bu uygulamada sadece parlaklık bilgisini göstereceğiz çünkü temel resim işleme algoritmalarının hemen hemen tamamı sadece parlaklık (gri tonlamalı) resimleri üzerine uygulanmaktadır. Aşağıda bu uygulamanın kodu gösterilmiştir.

```
#include"icb_gui.h"

int FRM1, FRM2;

void ICGUI_Create()
{
  ICG_MWTitle("Video Play");
  ICG_MWSize(800, 450);
}

void Goster()
{
  ICDEVICE dosya;
  ICBYTES Y,U,V,TersY;
  CreateImage(Y, 352, 288, ICB_CHAR);
  CreateMatrix(U, 176, 144, ICB_CHAR);
  CreateMatrix(V, 176, 144, ICB_CHAR);
```

```
  if (!CreateFileReader(dosya, "akiyo_cif.yuv"))
    return;//dosya açılamadı fonksiyondan çıkıyoruz
  while (ReadIntoMatrix(dosya, Y) == (352 * 288))
  {
    ReadIntoMatrix(dosya, U);
    ReadIntoMatrix(dosya, V);
    DisplayImage(FRM1, Y);
    FlipV(Y, TersY);
    DisplayImage(FRM2, TersY);
    Sleep(30);
  }
  CloseDevice(dosya);
}

void ICGUI_main()
{
  ICG_Button(5, 5, 150, 25, "VIDEO OYNAT", Goster);
  FRM1 = ICG_FrameSunken(5, 50, 360, 300);
  FRM2 = ICG_FrameSunken(400, 50, 360, 300);
}
```

Bu kaynak kodunun tamamını açıklamaya gerek yoktur. Önemli işlemlerin tamamı “VIDEO OYNAT” tuşuna basıldığında çağrılan Goster() fonksiyonu içinde gerçekleşmektedir. Bu fonksiyon içerisinde ismi dosya olan bir cihaz (ICDEVICE) tanımlanmaktadır. Bu cihaz sabit disk üzerindeki "akiyo_cif.yuv" dosyasına erişim için kullanılmaktadır. Bunu gerçekleştiren fonksiyon,

```
CreateFileReader(dosya, "akiyo_cif.yuv")
```

Komutu ile “dosya” cihazını bu dosyaya bağlamaktadır. Sonra renk bilgisinin her katmanı için birer matris oluşturulmaktadır: Y, U, V. Bu katmanlar video dosyası içerisinde şu şekilde yer almaktadırlar:

Y katmanını okumak için ReadIntoMatrix(dosya, Y) fonksiyonunu kullanmaktayız. Bu fonksiyon okuduğu bayt sayısını döndürmektedir. Dolaysı ile Y katmanını boyu olan 352x288 bayt’tan daha az okuma yaparsa video dosyasının

sonuna gelmişiz demektir. Y katmanını okuduktan sonra bunu resim çerçevesi-1'in içerisinde gösterdiğimizde resmin ters basıldığını görmekteyiz. Bunun sebebi grafik kartında yükseklik boyutu olan (y) boyutunun aşağıdan yukarı doğru artmasıdır. Oysa bir matriste bu boyut yukarıdan aşağıya doğru artar. Bu nedenle FlipV() fonksiyonu kullanarak bu resmi ters çevirip ekrana ikinci çerçeve içerisinde bastırdığımızda resim düzelmektedir. Y katmanını ekrana bastıktan sonra 30 milisaniye bekleyip ikinci filim karesini okumamız gerekir. Ancak ikinci Y katmanını okumadan evvel U ve V katmanlarını okumamız gerekir. İşte bu nedenle while döngüsü aşağıdaki gibi yazılmıştır:

```
while (ReadIntoMatrix(dosya, Y) == (352 * 288))
{
  ReadIntoMatrix(dosya, U);
  ReadIntoMatrix(dosya, V);
  DisplayImage(FRM1, Y);
```

Yani, Y katmanı okunduktan hemen sonra U ve V katmanları okunmakta, ancak o katmanlar ekranda gösterilmemektedir.

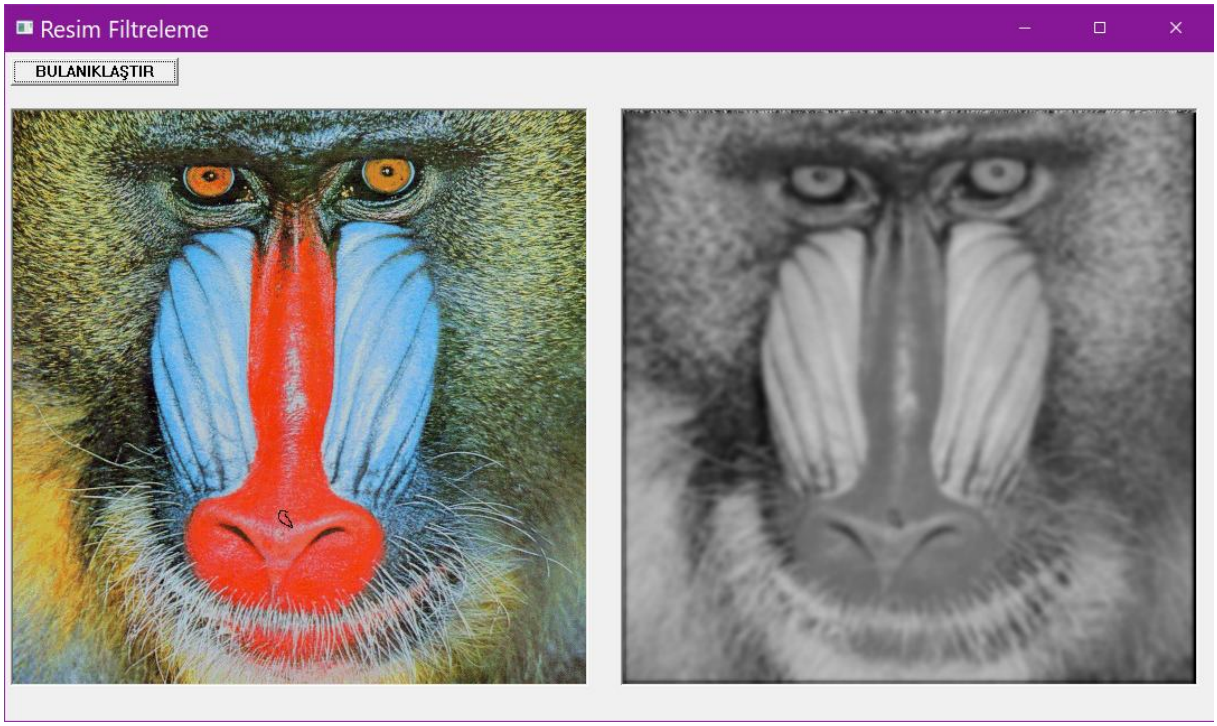
SORU: Y katmanı resim matrisi olarak yaratılmışken U ve V matrisleri sade matris olarak yaratılmıştır. Neden?

ALIŞTIRMA: Elinizdeki Y,U ve V katmanlarından RGB türünde renkli bir resim yaratıp bunu ekranda gösteriniz.

7.6. Resim Filtreleme

Resim işlemeye giriş derslerinde anlatılan ilk konulardan biri resim filtrelemedir. Çünkü bu işlem birçok başka algoritmanın ilk adımıdır. Mesela Canny kenar bulma algoritması resmin önce alçak geçirgen bir filtre ile işlenmesini gerektirir. Alçak geçirgen filtreleme resimdeki gürültüyü azalttığı gibi bulanıklaştırması sebebiyle gizliliği koruma uygulamalarında da kullanılır. Örneğin bir hastane yazılımı, hastaların vücut resmini gösterirken bunu bulanıklaştırması gerekebilir.

Diğer bir uygulama ise resim iyileştirmedir. Alçak geçirgen filtreler resmi gürültü ve bozukluklardan arındırırken yüksek geçirgen filtre ise resmin netliğini (keskinliğini) arttırır. Bu kitabın konusu resim işleme olmadığı için daha fazla detaya girilmeyecektir. Aşağıda alçak geçirgen bir filtre kullanarak bir resmi bulanıklaştıran bir uygulama gösterilmiştir.



Yukarıda resim işleme makalelerinde ve derslerde sıklıkla kullanılan "Mandril.jpg" resmini görüyorsunuz. Bu resmi internette rahatlıkla bulabilirsiniz. Sağdaki gri tonlamalı fotoğraftaki maymunun tüylerine dikkat ederseniz soldakine göre bariz şekilde bulanıklaştığını görebilirsiniz. Aşağıda bu uygulamanın kodu verilmiştir.

```
#include"icb_gui.h"

int FRM1, FRM2;

void ICGUI_Create()
{
  ICG_MWTitle("Resim Filtreleme");
  ICG_MWSize(1100, 650);
}

void Filtrele()
{
  ICBYTES RGB, gri, filitrelenmis;
  ICBYTES filtre = { 0.1,0.2,0.4,0.2,0.1 };
  ReadImage("mandril.jpg",RGB);
  DisplayImage(FRM1, RGB);
  Luma(RGB, gri);
```

```
  for (int x = 0; x < 5; x++)
  {
    FilterH(gri, filitrelenmis, filtre, ICB_UCHAR);
    FilterV(filitrelenmis, gri, filtre, ICB_UCHAR);
  }
  DisplayImage(FRM2, gri);
}

void ICGUI_main()
{
  ICG_Button(5, 5, 150, 25, "BULANIKLAŞTIR", Filtrele);
  FRM1 = ICG_FrameSunken(5, 50, 360, 300);
  FRM2 = ICG_FrameSunken(550, 50, 360, 300);
}
```

Olan biten herşeyin Filtrele() adlı fonksiyon içerisinde gerçekleştiği görülmektedir. Önce "filtre" isimli vektör içerisine alçak geçirgen filtremizin parametreleri yerleştirilmektedir. Daha sonra Mandril isimli resim dosyası okunup FRM1 kulpuna bağlı olan sol çerçeve içerisinde gösterilmektedir. Filreleme sadece resmin parlaklık birleşenine yapılabildiği için resim Luma() fonksiyonu kullanılarak gri tonlamalı hale getirilmiştir. Daha sonra bir for döngüsü içinde beş kere yatay ve dikey olarak filtrelenmiştir. Burada kullanılan FilterH() fonksiyonu "gri" akışkanı içindeki resmi, "filtre" vektörü kullanarak yatay yönde filtreleyip, "filtrelenmiş" akışkanına yazmaktadır:

```
FilterH(gri, filitrelenmis, filtre, ICB_UCHAR);
```

Sonucun "unsigned char" yani bayt cinsinden olmasını istemektedir. Daha sonra ortaya çıkan bu resmi dikey yönde filtreleyen FilterV() fonksiyonu sonucu tekrar "gri" isimli akışkana atmaktadır. Böylelikle döngünün başına döndüğümüzde yatay ve dikey yönde filtrelenmiş olan resim tekrar filtrelenmektedir. Bu şekilde beş kez filtrelenen resim FRM2 kulpuna bağlı olan soldaki çerçeve içerisinde gösterilmektedir.

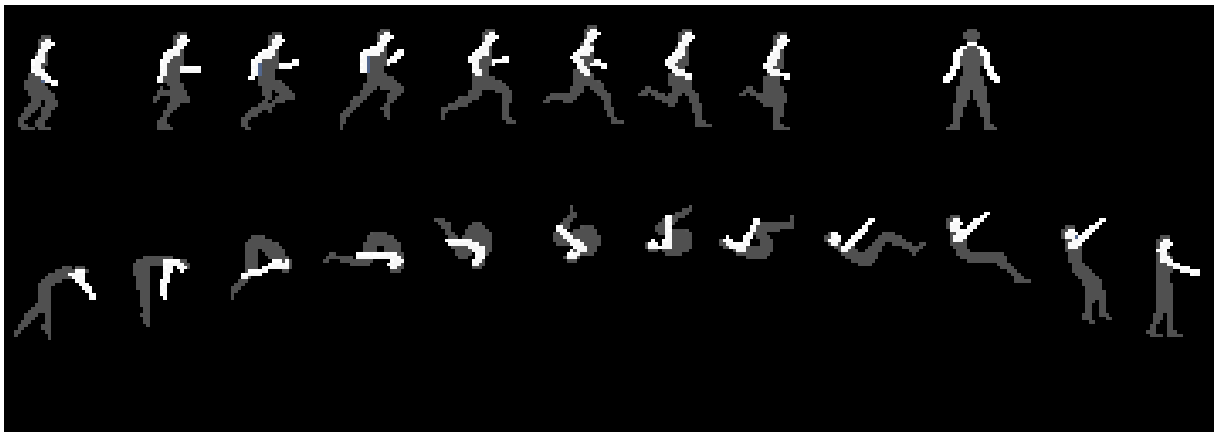
Bu örnekte basit parametreleri olan bir alçak geçirgen filtre kullanılmıştır. Merak edenler yüksek geçirgen bir filtre parametrelerinin ne olması gerektiğini öğrenip onunla resmi olduğundan daha net hale getiren bir uygulama da yazabilirler. Tek yapmaları gereken aşağıdaki satırda gösterilen parametreleri uygun şekilde değiştirmektir:

```
ICBYTES filtre = { 0.1,0.2,0.4,0.2,0.1 };
```

7.7. İleri Animasyon Teknikleri

Grafik animasyonları birçok uygulamada kullanılmaktadır. Bunların en çok kullanıldığı yerlerden biri de oyunlardır. Oyunlardaki hareketli nesneler yön değiştirdiklerinde farklı açılardan görülmekte veya uzaklaşıp yakınlaştıklarında büyüme ve küçülme animasyonları gerçekleştirmektedirler. Bu bölümde 1980'li yıllarda çok meşhur olmuş olan "Impossible Mission" isimli Commadore 64 bilgisayar oyununun nasıl yazılacağı anlatılacaktır.

Bu oyunda çılgın bir bilim adamının yeraltı sığınağındaki evinde dünyayı havaya uçuracak olan bombayı durduracak şifrenin anahtar parçalarını bulmaya çalışan bir ajan kontrol edilmektedir. Bu ajanın animasyon filim kareleri aşağıda gösterilmiştir.



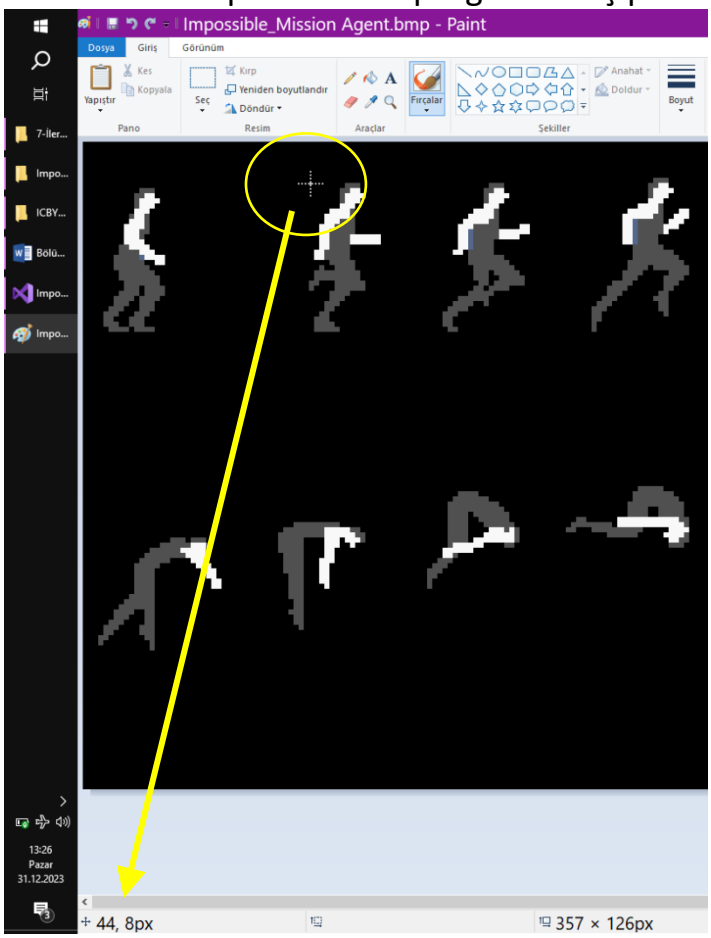
Bu tür oyun kahramanı animasyon film karelerinin bulunduğu resimlere "sprite sheet" denmektedir. Bu tür sprite sheet'ler, internette oyun programcıları veya eski oyun severler tarafından bol miktarda yayınlanmaktadır. Bizim yapmamız gereken her bir animasyon karesini ayrı bir resim matrisi olarak yükleyebilmektir. Eğer yukarıdaki resimden her kareyi teker teker kopyalayıp sonra da sırayla yapıştırabilirsek bir animasyon elde edebiliriz. Bunun için ICBYTES kütüphanesinde Copy() (kopyala), Paste() (yapıştır) ve PasteNone0() (sıfır olmayan yerleri yapıştır) gibi fonksiyonlar bulunmaktadır.

Aşağıdaki fonksiyon ajanımızın koşma animasyonu için gerekli olan kareleri 7 ayrı ICBYTES matrisine yükleyen kod örneğini bize göstermektedir.

```
ICBYTES AgentRun[7];

void LoadAgentRun()
{
      ICBYTES cordinat{ {45, 9, 13,30},{71,9,16,30},{100,8,20,30},
{130,8,23,30},{160,7,25,30},{189,8,22,30},{218,8,25,31} };
      for (int i = 1; i <= cordinat.Y(); i++)
      {
            Copy(Agent, cordinat.I(1,i), cordinat.I(2, i),
cordinat.I(3, i), cordinat.I(4, i), AgentRun[i-1]);
            PasteNon0(AgentRun[i-1], 33*i, 58, Corridor);
      }
      ICBYTES TV;
      MagnifyX3(Corridor, TV);
      DisplayImage(F1, TV);
}
```

Bu programa ilk bakışta, global bir array olarak tanımlanmış ICBYTES cinsinden bir değişken olan AgentRun[7] dizisinini görmekteyiz. Bu dizin koşmakta olan ajanın her bir filim karesi resmini sıralı olarak tutmak içindir. Fonksiyonun içerisinde ise adı "cordinat" olan 4x7 büyüklüğünde tam sayı bir matris görmekteyiz. Bunlar neyin koordinatıdır? Eğer yukarıdaki sprite sheet'i windows'daki paint isimli programla açıp fare imlecini yanda gösterildiği gibi



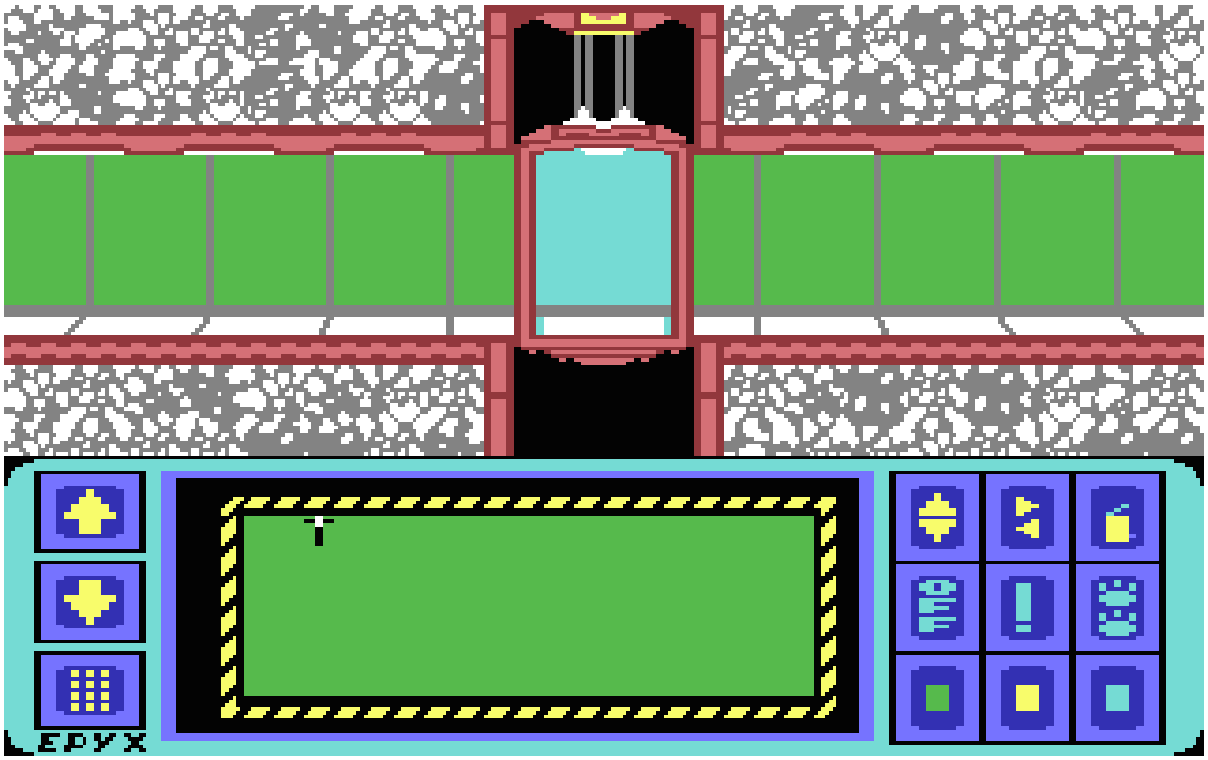
koşan ajanın ilk karesinin sol üst köşesine getirirsek, Paint programının sol alt köşesinde o noktanın resim üzerindeki x ve y koordinatını görürüz. Ancak Paint programında resmin sol üst köşe koordinatı (0,0) dan başlamaktadır. Diğer taraftan matrislerde sol üst köşe koordinatı (1,1) den başlar. Bu sebeple yukarıdaki kodda (44,8) yerine ilk iki koordinat:

```
ICBYTES cordinat{ {45, 9, 13,30},
```

Olarak tanımlanmıştır. 13,30 ise koşan ajanın ilk karesinin genişliği ve yüksekliğidir. Buradan, "cordinat" isimli matrisin geri kalan

elemanlarının koşan ajanın diğer karelerinin x ve y koordinatları ve genişlikleri olduğu anlaşılır.

LoadAgent() isimli fonksiyonun geri kalan satırlarında bir for döngüsü içerisinde 7 adet filim karesi sprite sheet'ten kesilip "Corridor" isimli resme yapıştırılmaktadır. Burada gösterilmemiştir ancak "Corridor" isimli ICBYTES akışkanı aslında aşağıdaki resmi taşımaktadır:

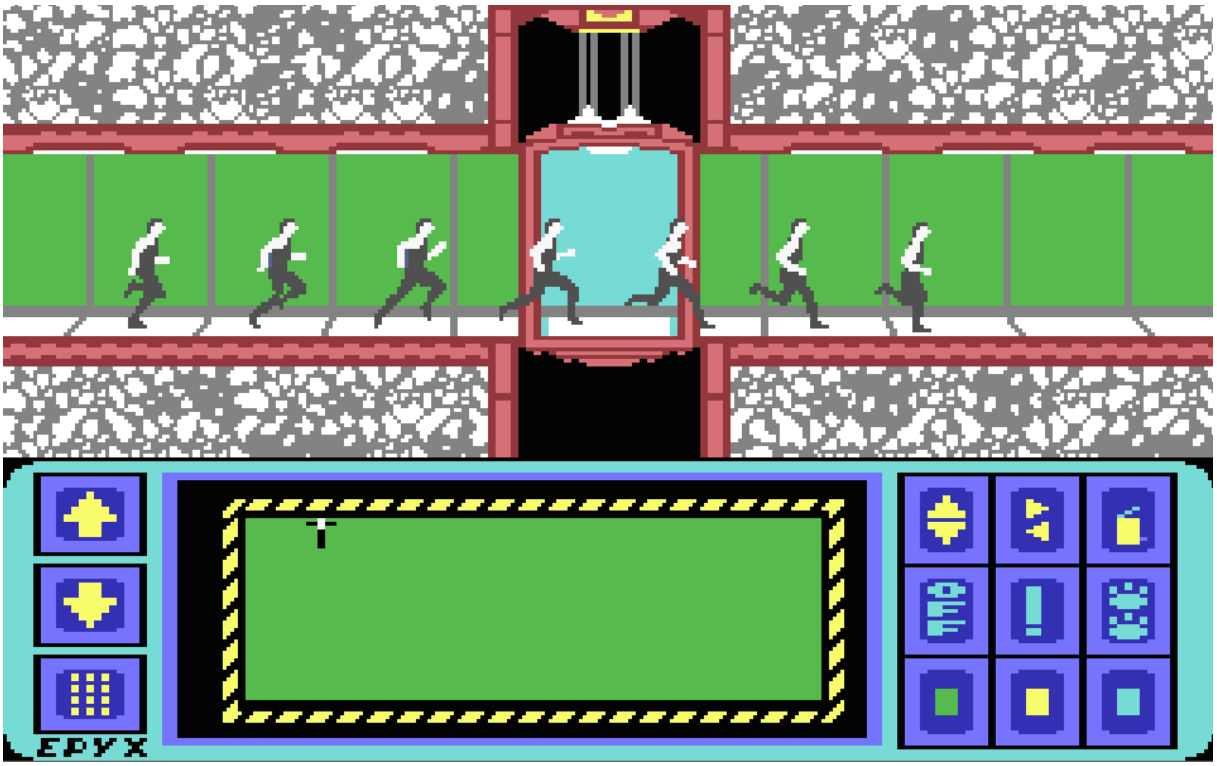



```
PasteNon0(AgentRun[i-1], 33*i, 58, Corridor);
```

Komutu ile ajanın koşar iken filim kareleri sıra ile 33 piksel aralıkla yukardaki resme yapıştırılmaktadır. Ancak ajanın resimlerinin bulunduğu "sprite sheet" resminde geri plan resmi siyahtır. PasteNone0() fonksiyonu yapıştırdığı resimde rengi tam siyah olan, yani Kırmızı yeşil ve mavi (KYM) kanallarının hepsinin sıfır olduğu pikseller yapıştırılmamaktadır. Daha sonra ise,

```
ICBYTES TV;
MagnifyX3(Corridor, TV);
DisplayImage(F1, TV);
```

Komutları ile tüm resim 3 kez büyütülüp ekrana bastırılmaktadır. Aşağıda ekrana bastırılan görüntü verilmiştir.



Eğer ajanın koşan halde animasyonunu yaptırmak istersek, ajan filim karesini yapıştırmadan evvel yapıştırılacağı bölgeyi geçici olarak bir matrise Copy() komutu ile kopyalayıp ondan sonra filim karesini oraya yapıştırmamız gerekir. Böylelikle ajan yana doğru kaydırılırken bu geçici matris tekrar o noktaya Paste() komutu ile yapıştırarak arka plan resmini bozmadan ajanı silmemiz mümkün olur.

7.8. Sürekli Çalışma Butonu

Daha evvel bahsettiğimiz gibi Windows işletim sistemi mesajlarla çalışır ve programınız bu mesajları işleyip sürekli olarak onlara tepki vermek zorundadır. Programınız bu mesajları işlemekte gecikirse programınız donmaya başlar. Mesajları işlemeyi bırakırsa ise tamamen tepkisiz hale gelir. Butonlar, menüler ve kademe çubukları gibi kutucukların hepsi tepkisiz hale gelir. Programınızın ana penceresini minimize edemez, yerini ve boyutunu değiştiremez, hatta kapatamaz hale gelirsiniz. Bu nedenle “7.3 Pacman’e Devam” adlı bölümde animasyonu sürekli kılmak için paralel çalışan bir sicim (thread) yaratmak zorunda kaldık.

Ancak sicimlerle uğraşmak ayrı bir uzmanlık gerektirmektedir ve çoğu bilgisayar mühendisi dahi bu konuda eğitim almamıştır. Bu nedenle kullanıcıyı sicimlerle uğraştırmak onlar açısından korkutucu, sonuçları açısından da tehlikeli

olabilmektedir. Bu konuda oluşabilecek sorunların önüne geçmek için I-See-Bytes kütüphanesi içerisinde farklı bir buton tanımı yapılmıştır. Bu butonun çağırdığı fonksiyon ana programa paralel olarak çalışan bir sicimdir aslında. Ancak kullanıcının bunu bilmesine gerek yoktur. Aç/Kapa butonu olarak adlandırdığımız bu buton aynı diğer butonlar gibi yaratılmaktadır:

```
ICG_TButton(5, 5, 120, 25, "BUTON", ButonFonksiyonu, void* arguman);
```

Bu buton kullanılarak sicimlerle (thread) uğraşmadan oyunlarda ihtiyacınız olan sürekli animasyonlar veya multimedya uygulamaları yaratabilirsiniz.

EK-1 ICGUI Fonksiyonları

Temel Fonksiyonlar:

Temel fonksiyonlar uygulamanın tamamını etkileyen fonksiyonlardır.

`void ICGUI_Create()`

Uygulamanın en dıştaki ana penceresini yaratan fonksiyon. Grafik arabirim olarak ICGUI kullanıyorsanız mutlaka çağırmanız gereken 2 fonksiyondan biri. Geneli ekleyen renk, font, gibi stiller; ana pencerenin yeri ve ebatları, uygulamanın pencerenin üst tarafındaki başlık çubuğuna yazılan ismi burada belirlenebilir.

`void ICGUI_main()`

Uygulamanın asıl çalıştığı kısım. Ana pencere üzerinde var olacak kutucuklar bu fonksiyon içerisinde yaratılır. C/C++'daki main gibi çalışır.

`void ICG_MWSize(int width, int height)`

Ana pencerenin genişlik (width) ve yüksekliğini (height) ayarlar.

`void ICG_MWPosition(int X, int Y)`

Ana pencerenin sol üst köşe koordinatlarını belirler.

`void ICG_MWTitle(const TCHAR* title)`

Ana pencerenin üst tarafındaki başlık çubuğuna yazılan ismi belirler. Bu fonksiyon kullanılmazsa pencere ismi demirbaş olarak ICB_GUI'dir.

`void ICG_MWColor(unsigned R, unsigned G, unsigned B)`

Ana pencerenin ve bazı kutucukların geri plan rengini belirler. Kırmızı (R), Yeşil (G), ve Mavi (B) tonlarının 0-255 arasındaki parlaklık düzeylerinin karışımı ile herhangi bir renk elde edilebilir.

`void ICG_MW_RemoveTitleBar()`
Ana pencerenin üst tarafındaki başlık çubuğunu ortadan kaldırır. Bu fonksiyon çok dikkatli kullanılmalıdır çünkü uygulamayı minimize ve maksimize eden ve sonlandıran düğmeler de ortadan kalkar. Programı sonlandırmak için başka mekanizmalar yerleştirilmiş olmalıdır aksi halde uygulamayı kapatamazsınız.

`HWND ICG_GetHWND(int handle)`

Windows işletim sisteminde her pencere ve kutucuk için bir kulp yaratılmaktadır. Bu kulp `HWND` cinsindendir. ICGUI kütüphanesinde ise tüm kutucukların kulpu tam sayıdır (`int`). Bu fonksiyon ICGUI tarafından yaratılan kutucukların Windows kulpuna ulaşmayı sağlar. WIN32 API'ı bilmiyorsanız kullanmamanız önerilir.

`bool ICG_SetFont(int H, int W, const char* fontname)`
Windows'un demirbaş yazı tipi yerine başka bir yazı tipi seçer. Bu fonksiyon çağrıldığı andan itibaren yaratılan kutucukları etkiler. İlk iki girdisi yazı tipinin yüksekliği (H) ve genişliğidir (W). Ancak genişlik 0 verilirse yüksekliğin aynısı kullanılır. Yazı tipi (fontname) MS Word programında listesi verilen "Times New Roman","Arial", veya şu anda kullandığımız yazı tipi "Calibri" gibi isimlerdir. Eğer istenilen yazı tipi sistemde yüklü değilse `false` döndürür.

`void ICG_SetSystemFont()`
Çağrıldığı andan itibaren yaratılan kutucukların yazı tipini yeniden Windows'un demirbaş yazı tipine çevirir.

`void ICG_DestroyWidget(int handle)`
Herhangi bir kutucuğu yok eder. Kutucuğun kulpu fonksiyonun girdisidir.

`bool ICG_SetWindowText(int handle, const char* string)`
Yazısı olan her türlü kutucuğun yazısını değiştirir.

`void ICG_SetOnMouseMove(void (*MouseFunc)(int, int))`
Fare her hareket edişinde çalıştırılacak fonksiyonu seçer.

`int ICG_GetMouseX()`
Farenin o anda ana pencerenin sol üst köşesine bağıl olarak yatay konumunu piksel cinsinden söyler.

`int ICG_GetMouseY()`
Farenin o anda ana pencerenin sol üst köşesine bağıl olarak düşey konumunu piksel cinsinden söyler.

`void ICG_SetOnMouseLClick(void (*MouseFunc)())`
Farenin sol tuşuna basıldığında çağrılacak fonksiyonu seçer.

void ICG_SetOnKeyPressed(void(*OnKeyPressedFunc)(int))
Klavyenin herhangi bir tuşuna basıldığında çalıştırılacak fonksiyonu seçer.

void SetText(int handle, ICBYTES& m)
ICBYTES akışkanı (m) içerisindeki karakter dizgisini girdi olarak verilen kulpa (handle) bağlı kutucuğun yazısı haline getirir.

void GetText(int handle, ICBYTES& m)
Girdi olarak verilen kulpa (handle) bağlı kutucuğun yazısını ICBYTES cinsinden akışkan içerisine kopyalar.

Metin Kutucuğu Fonksiyonları:

Aşağıda tanımları verilen fonksiyonlar, çeşitli tipte metin kutucukları yaratmak veya içlerine yazdırmak için kullanılır.

int ICG_SLEdit()
Çerçevesiz tek satırlı metin kutucuğu yaratır.

int ICG_SLEditBorder()
Çerçevesi ince siyah bir çizgi olan tek satırlı metin kutucuğu yaratır.

int ICG_SLEditThick()
Çerçevesi kalın dış bükey tek satırlı metin kutucuğu yaratır.

int ICG_SLEditSunken()
Çerçevesi içe göçük hissi veren tek satırlı metin kutucuğu yaratır.

int ICG_SLPasswordB()
Çerçevesi ince siyah bir çizgi olan tek satırlı şifre giriş kutucuğu yaratır.

int ICG_SLPasswordSunken()
Çerçevesi içe göçük hissi veren tek satırlı şifre giriş kutucuğu yaratır.

int ICG_IPAddressSunken( )
Çerçevesi içe göçük hissi veren tek satırlı internet adres giriş (IP versiyon 4) kutucuğu yaratır.

int ICG_IPAddressThick()

Çerçevesi kalın dış bükey tek satırlı internet adres giriş (IP v4) kutucuğu yaratır.

int Print(int handle, ICBYTES& i)

ICBYTES cinsinden olan "i" akışkanı içerisinde karakter dizgesi varsa bunu "handle" kulpuna bağlı metin kutucuğuna yazdırır. Eğer i matrisi tek satırlı ise yani bir vektörse tek satır yazar. Eğer i bir matris taşıyorsa matrisin her satırını alt alta yazar.

int ICG_printf(int handle, const char* format, ...)

Kulpu verilen (handle) metin kutucuğu içerisine C/C++'daki printf() fonksiyonunun davranışını taklit ederek yazdırır. Aynı format ve değişken tipi kurallarını sergiler.

void ICG_SetPrintWindow(int handle)

Yukarıdaki gibi sürekli metin kutucuğu kulpu seçmek istemiyorsanız, yani sürekli olarak aynı metin kutucuğuna yazdıracaksanız bunu baştan belirtip sonra kulp (handle) yollamadan ICG_printf() fonksiyonu çağırmak için önce bu fonksiyonu çağırmalısınız.

void ICG_SetPrintWindow(HWND c)

Bir önceki fonksiyonun aynısını yapıyor ancak ICGUI kulpu yerine Windows işletim sistemi kulpu kullanıyor.

int ICG_printf(const char* format, ...)

Baştan yazacağı metin kutucuğu belirlendiyse C/C++'daki printf() fonksiyonunun birebir aynısı.

EK-2 ICBYTES Matris Fonksiyonları

Matris Yaratma Yöntemleri:

Bu bölümde, ICBYTES akışkanı içerisinde matris yaratma yöntemleri gösterilecektir.

```
ICBYTES A = {{1,2,3},{4,5,6},{7,8,9}};
```
integer 3x3 matris yaratır.

```
ICBYTES A = {{1.0,2.0,3.0},{4.0,5.0,6.0},{7.0,8.0,9.0}};
```
double 3x3 matris yaratır.

CreateMatrix(A, X,Y,Z,W, ICB_TYPE)
XxYxZxW boyutlarında ICB_TYPE cinsinden bir matris yaratır.

CreateImage(A, X,Y,Z,W, ICB_TYPE)
XxYxZxW boyutlarında ICB_TYPE cinsinden bir matris yaratır. Ekranda gösterimi daha hızlıdır. Sade matrise göre yaratılması ve yok edilmesi daha yavaştır. Daha fazla RAM harcar.

CreateSound(i, x, y, ICB_TYPE, Hz)
X kanallı Y uzunluğunda sayısal ses örneği içeren, Hz örnekleme hızında çalınacak matris yaratır. Ses giriş/çıkış cihazlarında hızlı biçimde çalınmak için uygundur.

```
ICBYTES A;
```
A=5.3; veya A=-2;
A akışkanı içerisinde 1x1 matris yaratır. Matris cinsi ya double'dır (5.3) veya int olur (-2).

```
A = "ABCDEF";
```
A akışkanı içerisinde 6x1 char cinsinden matris yaratır ve içerisine (ABCDEF) kopyalanır.

Matris Sorgulama:

Bu kısımda matrisi kendisi hakkında bilgi edinmek için sorgulayabileceğiniz fonksiyonlar listelenmiştir.

```
bool IsFloating(ICBYTES& i)
```
Matris cinsi float veya double (floating point) ise `true` döndürür.

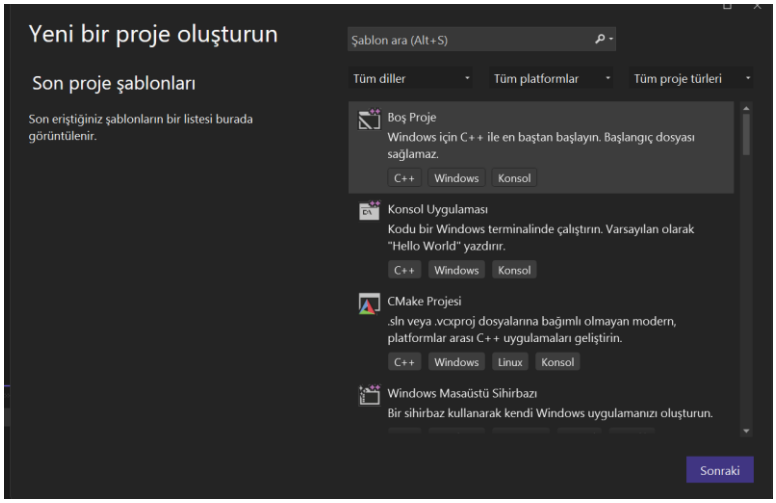
```
bool IsMatrix(ICBYTES& i)
```
Eğer girdi olarak yollanan akışkan matris taşıyorsa `true` döndürür.

```
bool AreDimsEqual(ICBYTES& i, ICBYTES& j)
```
Eğer girdi olarak yollanan matrislerin boyutları eşitse `true` döndürür.

```
bool AreEqualImage(ICBYTES& i, ICBYTES& j)
```
Eğer girdi olarak yollanan matrislerin boyutları ve renk derinlikleri eşitse `true` döndürür.

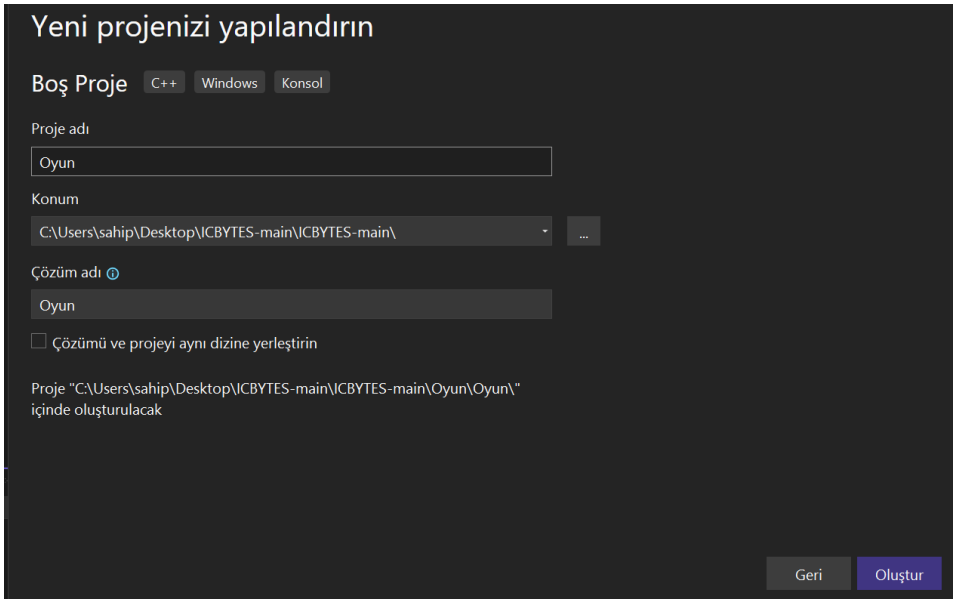
EK-3 ICBYTES Projesi Yaratma

Bu bölümde, I-See-Bytes kütüphanesi kullanarak yeni bir projenin nasıl yaratılacağı anlatılmaktadır. Öncelikle ICBYTES kütüphanesini indirip ziplenmiş dosyaları ayıklayıp (açıp) sabit diskinize kopyalayınız. Visual Studio'yu açtıktan

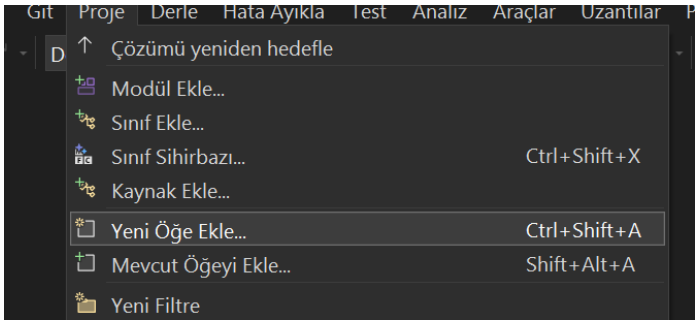


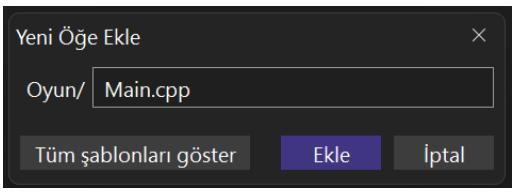
sonra Menüden yeni proje yaratmayı seçip, sonra da aşağıda gösterildiği gibi boş projeyi seçiniz:

Sonra proje klasörünü ICBYTES-main klasörü içerisinde seçip projeye bir isim verin. Mesela "OYUN" olsun:



Şimdi sırada projeye C/C++ dosyaları eklemek var. ICGUI_Main() fonksiyonumuzun veya GUI kullanmayacak isek main() veya WIN32 API kullanacaksak WinMain() fonksiyonumuzun olacağı en az bir C/C++ dosyası eklememiz zorunlu olduğuna göre bu dosyaya Main.cpp adı vereceksek, öncelikle, C++ dosyası eklemek için menüden Proje->YeniÖğeEkle seçmeliyiz. Ardından da önümüze gelen minik pencere içerisine "Main.cpp" (mesela) yazabiliriz:



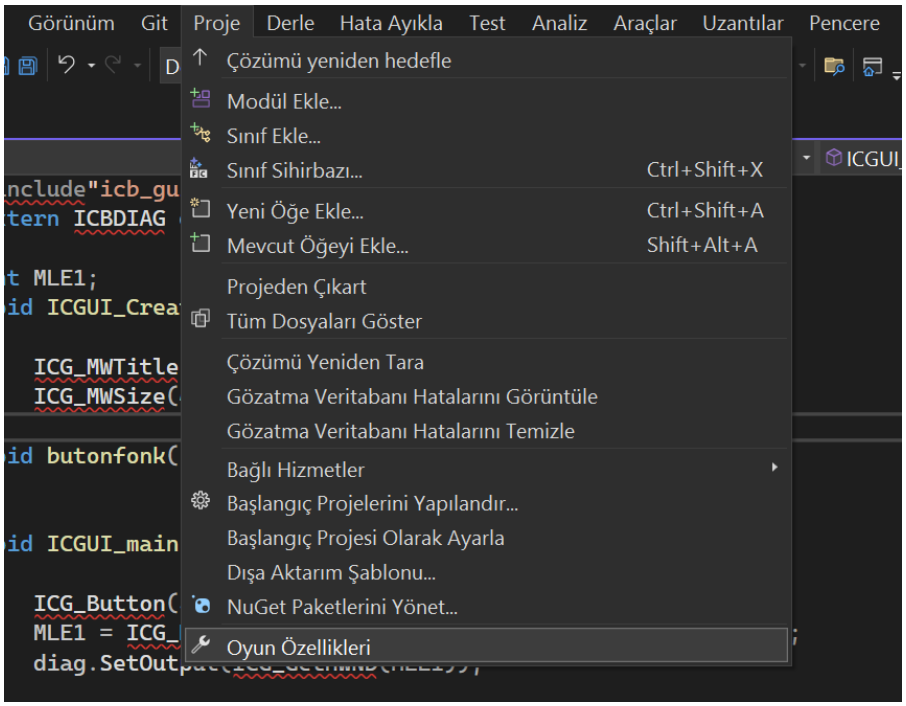


Sonra Main.cpp'nin içine bir şeyler yazalım:

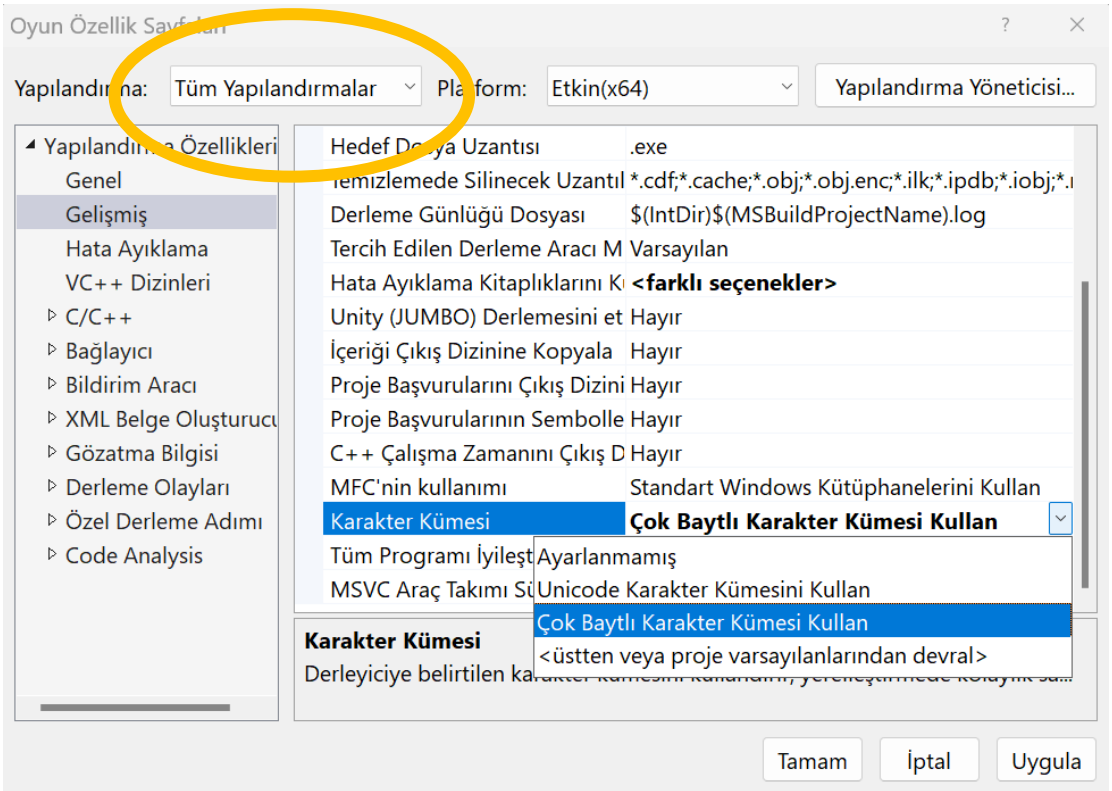
```
1   #include"icb_gui.h"
2   extern ICBDIAG diag;
3
4   int MLE1;
5   void ICGUI_Create()
6   {
7       ICG_MWTitle("OYUN");
8       ICG_MWSize(430, 300);
9   }
10  void butonfonk()
11  {
12  }
13  void ICGUI_main()
14  {
15      ICG_Button(5, 5, 120, 25, "Buton1", butonfonk);
16      MLE1 = ICG_MLEditSunken(5, 50, 250, 150, "", SCROLLBAR_V);
17      diag.SetOutput(ICG_GetHWND(MLE1));
18  }
```

Artık projemiz çalışmak için gereken dosyalara sahiptir. Şimdi sırada I-See-Bytes kütüphanesinin projeye dahil edilmesi ve proje ayarlarının yapılması var.

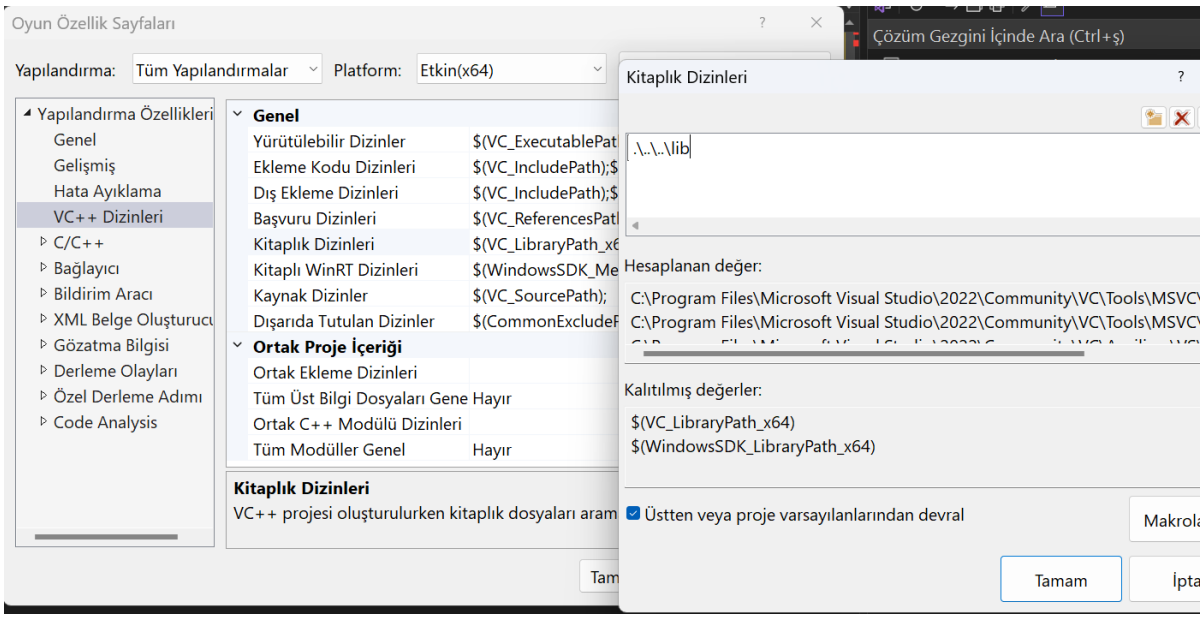
Proje özelliklerine girmek için Visual Studio menüsünden Proje->Oyun Özelliklerini seçelim:



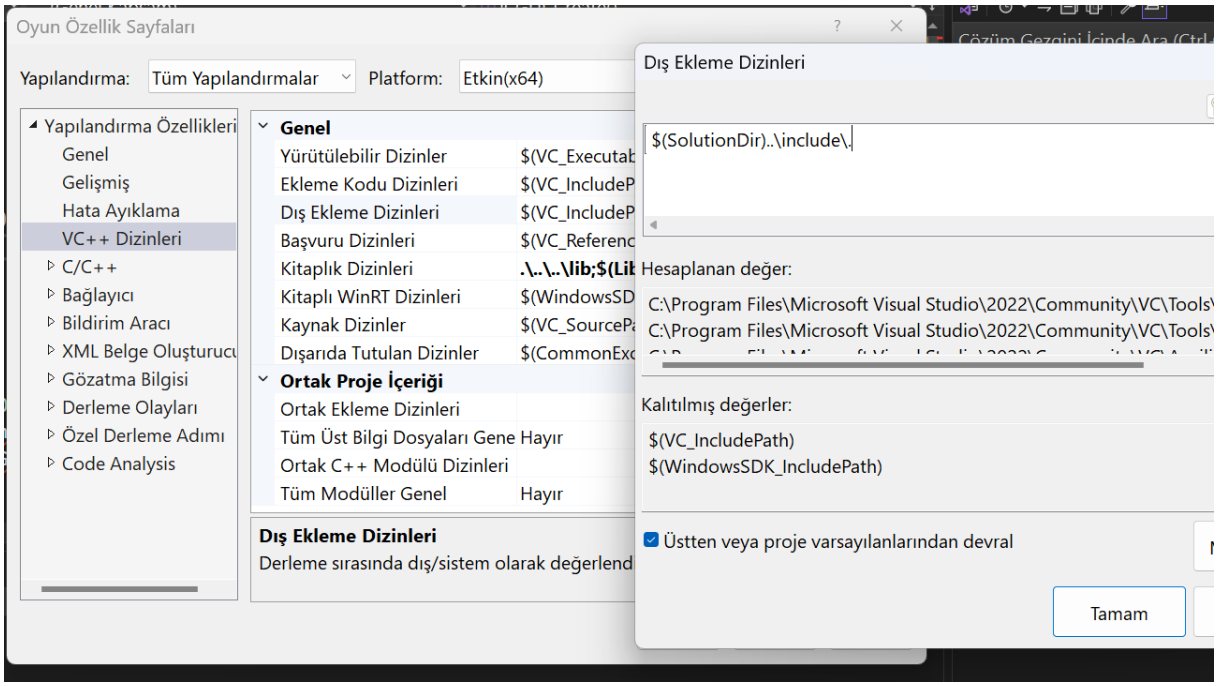
Ortaya çıkan pencerede, üst tarafta tüm yapılandırmaları seçip sonra da ”Çok Baytlı Karakter Kümelerini Kullanı” seçiniz.



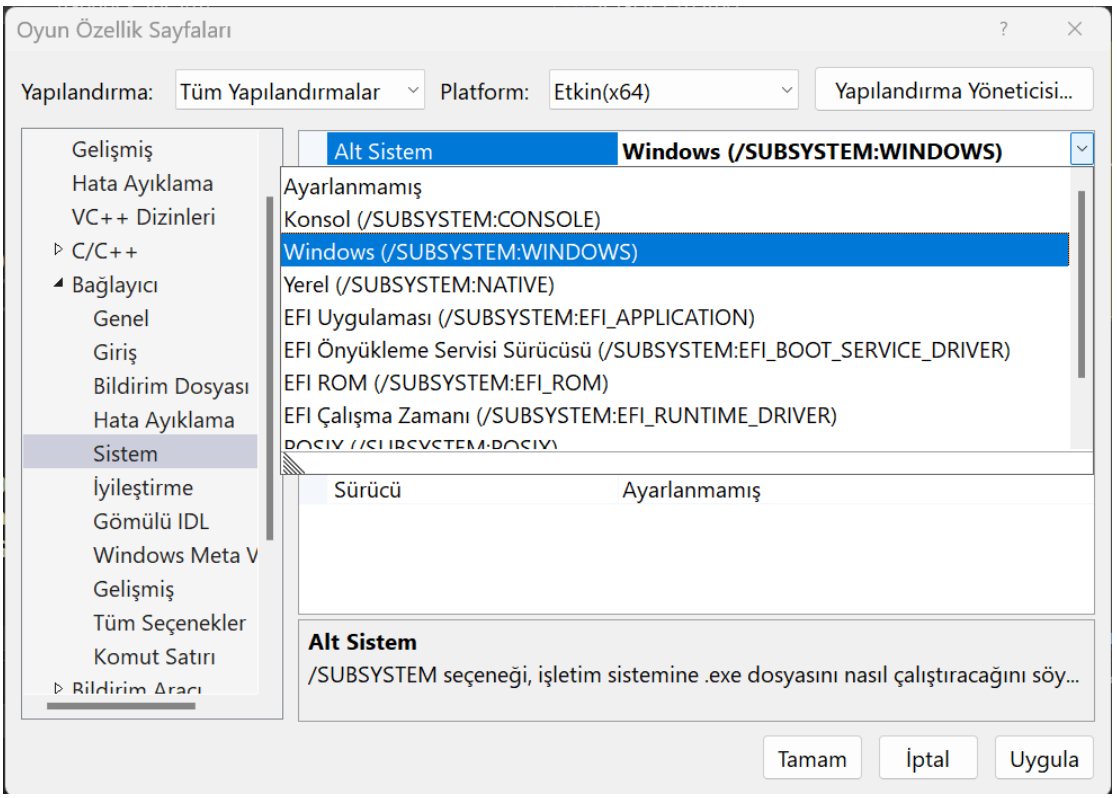
Sonra sırada kitaplık dizinlerinin derleyiciye bildirmesi var:



İnternetten indirmiş olduğumuz ICBYTES-main.zip dosyasını açtığımızda içerisinde “lib” ve “include” diye iki klasör olduğunu olduğunu farkedeceksiniz. Bu klasörler ICBYTES kütüphanesinin derlenmiş “.lib” ve “.h” dosyalarını içermektedir. Derleyiciye bu klasörlerin yerini bizim proje klasörümüze izafi olarak belirtmekteyiz. Böylelikle yazdığımız projeyi başka bir bilgisayarda derlenmesi gerekirse, mesela iş arkadaşlarımıza veya dersin asistanına ödev olarak yollamamız gerekiyorsa, onların bilgisayarında da sorunsuz derlenmesini sağlamış oluruz. Şimdi de “include” dosyalarının yerini söylüyoruz:



Ve son olarak konsol uygulaması değil, windows uygulaması yaratmak istediğimiz söylüyoruz. Tabi ki konsol uygulaması yapmak istersek bunu değiştirmemize gerek yoktur.



BU NOKTADA TAMAM'A BASIP PENCEREYİ KAPATIN.

Ancak henüz bitmedi. Sonraki sayfada devam etmektedir.

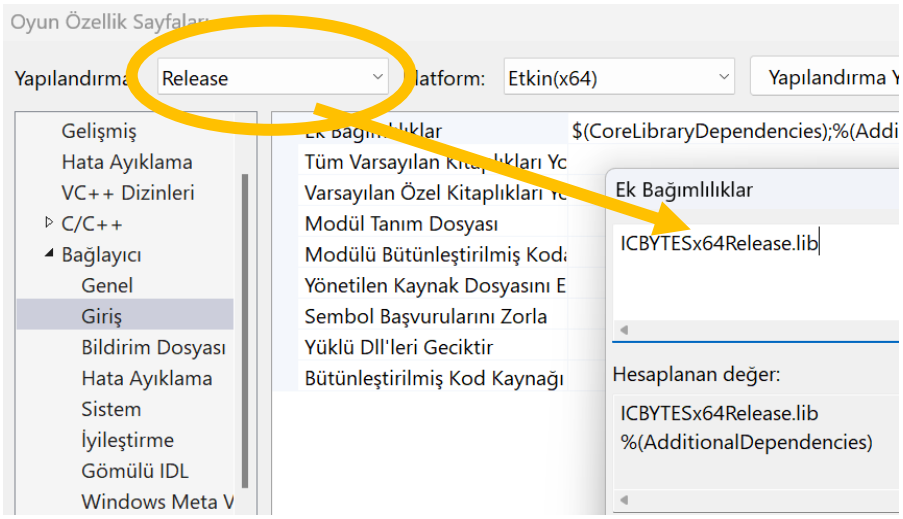
Şimdi sırada ".lib" dosyalarının eklenmesi var. Ancak "lib" klasörünün içerisinde gördüğümüz üzere iki farklı lib (library yani kütüphane) dosyası bulunmakta. Bunlar:

ICBYTESx64Debug.lib

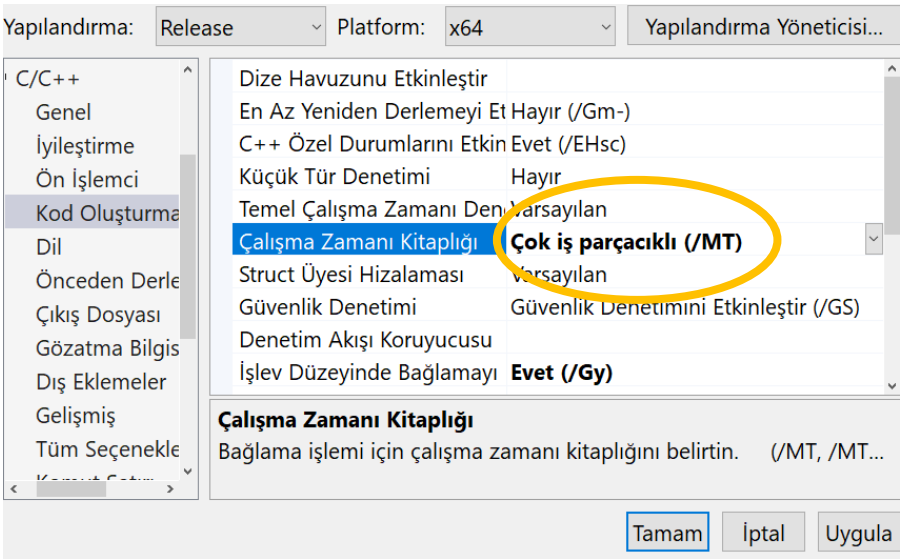
ve,

ICBYTESx64Release.lib

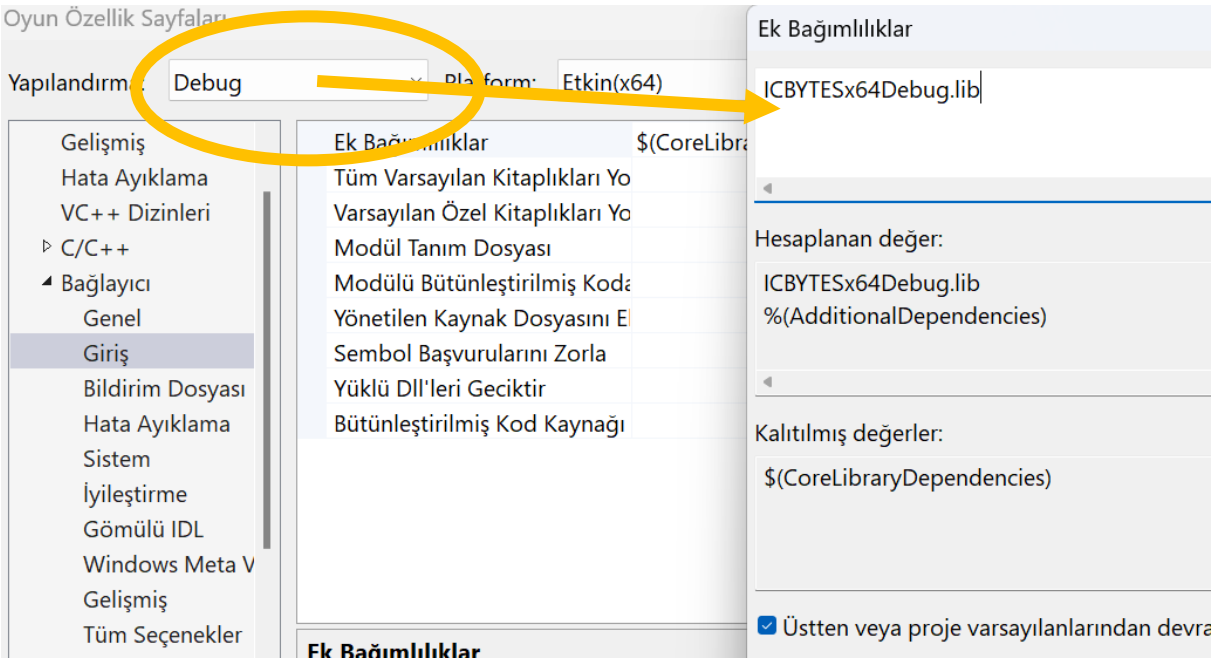
dosyalarıdır. Programımızı derlerken iki farklı modda derleyebiliyoruz. Bunlar Release ve Debug modlarıdır. Bu modlarda farlı kütüphanelerin ve farklı derleme seçeneklerinin seçilmesi gerekmektedir. Proje özelliklerine girmek için yeniden Visual Studio menüsünden Proje->Oyun Özelliklerini seçelim. Ancak bu kez üst sol köşedenRELEASE modunu seçip, release kütüphanesini kullanması için Bağlayıcı→Giriş→Ek Bağımlılıklar'ı seçip kütüphane dosyasının ismini yazıyoruz:



Ardından da C/C++ → kod oluşturmaya girip "Çalışma Zamanı Kitaplığı" seçeneğinini "**Çok iş parçacıklı (/MT)**" olarak değiştirip "Tamam'a" basıyoruz.



Debug modundadaki ayarları yapmak için Proje->Oyun Özelliklerine tekrar girmemiz gerekiyor. Sol üst köşedeki "Yapılandırma" seçeneklerinden bu kez "Debug" seçtikten sonra Bağlayıcı→Giriş→Ek Bağımlılıklar'ı seçip bu kez debug kütüphane dosyasının ismini yazıyoruz:



Release modunda olduğu gibi Debug modunda da C/C++ → kod oluşturmaya girip "Çalışma Zamanı Kitaplığı" seçeneğinini "**Çok iş parçacıklı (/MTd)**" olarak değiştirip "Tamam'a" basıyoruz. (Release'de **/MT** idi.) Artık projemiz derlenmeye hazırdır.