YUV (YCbCr) DECODER

Mehmet Oğuz Aktaş (150201170), Ebubekir Şit (150201245)

[oguzaktass@gmail.com](mailto:oguzaktass@gmail.com), [sitebubekir@gmail.com](mailto:sitebubekir@gmail.com)

Kocaeli Üniversitesi

Mühendislik Fakültesi

Bilgisayar Mühendisliği Bölümü

Özet

*Projede bizden istenen özellikleri bulunduran basit bir arayüz tasarladıktan sonra, YUV dosyasının kullanıcı tarafından seçilmesini sağladık. YUV dosyasındaki tüm byte değerlerini okuyup oluşturduğumuz byte tipindeki diziye attıktan sonra, YUV formatına ve resolution değerine göre render/parse işlemini ParseYUV() metodunu yazarak yaptık. Bu metot ile seçilen dosyanın YUV formatına göre frame sayısını da hesapladık. Daha sonra frame sayısı kadar Bitmap uzantılı dosya oluşturup bu dosyaları numaralandırarak frames dizinine kaydetme ve oluşturulan her Bitmap’i bir List’e ekleme işlemlerini SaveBitmaps() metodunu yazarak yaptık. Görüntüyü gösterme aşaması için de Frames formunu oluşturarak Constructor parametresi ile Bitmap listesini Main formundan aldık. Timer ile görüntü geçişlerindeki bekleme süresini (frame rate) ayarladıktan sonra Durdur, Devam Et ve Tekrar Oynat seçenekleri ile birlikte frame by frame görüntü oynatma arayüzünü yaparak projeyi tamamladık. Projede gereksiz kod tekrarları yapmamaya ve gereksiz yere bellekten alan ayırmamaya dikkat ettik, bu amaçla ReSharper Ultimate eklentisini kullanarak kod kalitesini arttırdık. Her aşamada kod yazarken algoritma karmaşıklığını ve programın hızlı çalışmasını hesaba kattığımız için yaptığımız projenin yeterince özgün olduğunu ve istenenleri fazlasıyla karşıladığını söyleyebiliriz.*

# Giriş

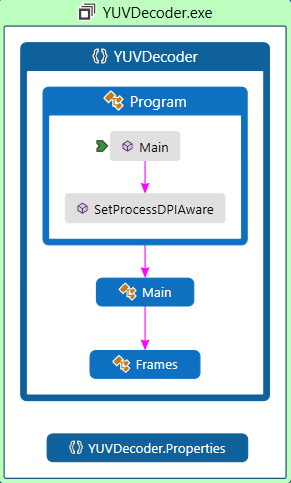
Bu projede, YUV renk encoding sistemiyle hazırlanmış 4:4:4 (Chroma yani renk kanallarında yatayda ve düşeyde alt örnekleme yapılmayan), 4:2:2 (Chroma kanallarında ½ oranında yatay alt örnekleme yapılan), 4:2:0 (Chroma kanallarında ½ yatay ve ½ düşey alt örnekleme yapılan) YUV formatlarındaki dosyaları okuduktan sonra (formata ve resolution’a göre) bileşenlerine ayırıp (render/parse işlemi) ile her frame’i sadece Y bileşenlerini seçerek RGB renk uzayına çevirip Bitmap dosyaları olarak kaydetmemiz ve daha sonra görüntüyü arayüz üzerinden frame by frame biçiminde oynatmamız istenmektedir.

Gerekli araştırmaları yapıp projeyi anladıktan sonra, bu projenin daha kapsamlı bir hali olan ticari amaçla yazılmış, <http://www.sunrayimage.com/yuvplayer.html> adresindeki YUVPlayer programını indirip arayüzünü inceleyerek projenin amaçları ve nasıl ilerleyeceğimiz hakkında ön fikir sahibi olduk. Bize verilen örnek YUV dosyaları dışında Internette bulduğumuz çoğu YUV dosyası packed image formatında (Her bir frame için bileşenler byte akışında teker teker Y, U, V sırasıyla gidiyor) olduğu için Ayarlar bölümünde YUV format ve Resolution’a ek olarak Pixel format seçeneğini ekledik. Bize verilen örneklerin hepsi planar image formatında olduğu için default olarak planar formatını seçili yaptık. Render/parse işleminde 4:2:0 YUV formatı sadece planar şeklinde bulunduğu için (Internetteki araştırmalarımız sonucunda bu bilgiye ulaştık.) diğer 2 YUV formatı için planar ve packed seçeneklerini ParseYUV() metodu içinde kodladık. Tüm istenenleri yaptıktan sonra test işlemleri ile son düzeltmeleri yaparak projeyi eksiksiz bir biçimde tamamladık.

# Temel Bilgiler

Programı yazmak yazmak için .NET Framework 4.6.1 platformu ile C# dilini, Visual Studio Enterprise 2017’yi ve JetBrains ReSharper Ultimate eklentisini kullandık. .NET Framework hazır kütüphaneleri, classları ve tasarım componentleri dışında NuGet ile veya farklı şekilde herhangi bir hazır kütüphane kullanmadık.

# Yazılım Mimarisi



*Şekil 1: Programda bulunan classlar*

Uygulamada Main ve Frames isimlerinde 2 tane Windows Form classı kullandık, onun dışında her Solution oluşturulduğunda hazır olarak gelen ve sadece Main metodunu bulunduran Program classını kullandık.

Dosya seçimi, dosya okuma ve parse işlemi sonrası Bitmap dosyalarının kaydedilmesi işlemlerini yaptığımız Main classı içinde kullandığımız metotlardan bazıları şunlar;

* CheckInputs(): Boolean tipinde ve parametre almayan bu metot ile dosyanın seçili olup olmadığını, ayarlarda boş alan bırakılıp bırakılmadığı, eğer özel resolution değerleri girilecekse width ve height boş bırakılıp bırakılmadığı, eğer boş bırakılmadıysa değerlerin 0’dan büyük olup olmadığı gibi kontrolleri yaptık. Metodu dosya okuma butonu içinde kullanarak eğer true dönüyorsa okuma işleminin yapılmasını, false dönüyorsa da ilgili hata mesajının görüntülenmesini sağladık.
* ParseYUV(): Integer tipinde 4 parametre alan (yuvFormat, width, height, pixelFormat) bu metot ile YUV dosyalarını okuyup Y bileşenleri byte değerlerini kaydettik (render/parse işlemi). Internette karşılaştığımız YUV örneklerinden dolayı planar image formatı (her frame için byte akışı önce tüm Y’ler sonra tüm U’lar sonra tüm V’ler şeklinde) dışında packed image formatı için de kodladık.
* SaveBitmaps(): Bu metot ile ParseYUV() metodunda her frame için ayırdığımız Y bileşenlerini döngü içinde Color classı hazır metodu olan FromArgb() ile sadece Y bileşeni değerlerini RGB’ye çevirip (siyah beyaz bir görüntü oluşuyor) her frame için Bitmap dosyalarını oluşturduğumuz Bitmap türündeki List’e ekledik ve aynı zamanda frames dizinine Bitmap classının Save() hazır metodu ile kaydettik.
* ButtonSelect\_Click(): Dosya Seç butonuna basıldığında çalışacak olan bu metot içinde yeni bir OpenFileDialog oluşturarak kullanıcı tarafından sadece .yuv uzantılı dosyaların seçilebilmesini sağladık. Eğer dosya seçimi başarılıysa FilePath TextBox’ına dosya yolunu yazdırdık.
* Main\_Load(): Bu hazır metot ile form açıldığında YUV formatı, resolution ve pixel formatı (image format, planar/packed) ComboBox’larında default olarak ilk değerlerin seçilmesini sağladık.
* TextBoxWidth\_KeyPress(): Bu hazır component metodu ile özel width değerleri girilirken sayı dışında bir değer girilmemesini kontrol ettik. Aynı şekilde height değeri için TextBoxHeight\_KeyPress() metodunu yazdık.
* ButtonRead\_Click(): Dosyayı Oku butonuna basıldığında çalışacak olan bu metot içinde, CheckInputs() true dönüyorsa ilk olarak frames dizinindeki önceki kayıtları ve bitmaps List’ini temizledik, daha sonra ParseYUV() ve SaveBitmaps() metotlarını çalıştırdık.
* ButtonShow\_Click(): Görüntüyü Oynat butonuna basıldığında çalışacak bu metot içinde, eğer bitmaps listesi boş değilse Frames formunun bitmaps List’ini parametre olarak açılmasını sağladık.

Görüntüyü oynatma işlemini yaptığımız Durdur, Devam Et ve Tekrar Oynat seçeneklerinin bulunduğu Frames classı içinde kullandığımız metotlardan bazıları şunlar;

* Frames(List<Bitmap> bitmap): Bu Constructor metodu ile bitmaps listesini Main classından aldık.
* Play(): Bu metot ile Timer’ı başlatarak (default 25 FPS) görüntünün oynatılmasını sağladık. Timer her çalıştığında görüntünün değişmesini de Timer\_Tick() hazır component metodu içinde ayarladık.
* Timer\_Tick(): Timer componenti hazır metodu olan bu metot ile PictureBox’taki Bitmap şeklinde framelerin FPS değerine göre (kullanıcı tarafından girilmediği sürece default 25) değişmesini sağladık. Eğer bitmaps List’indeki tüm dosyalar gösterilmişse Timer’ı durdurduk. Ayrıca gösterme devam ederken Tekrar Oynat butonuna basılma durumları için boolean tipindeki flag kontrol değişkenini tanımlayıp kullanarak bu durumlarda, önceki oynatmanın durdurulup yenisinin başlatılmasını sağladık.
* ButtonReplay\_Click(): Tekrar Oynat butonuna basıldığında çalışacak bu metot ile görüntünün girilen FPS değerine göre (eğer boş bırakılırsa default değer olan 25 alınacak) görüntünün tekrar oynatılmasını sağladık. FPS TextBox’ına 0 girilmemesi ve 1000 üstünde değer girilmemesi kontrollerini de bu metot içinde yaptık.
* TextBoxFPS\_KeyPress(): Bu metot ile FPS alanına sayı dışında değer girilmemesini sağladık.
* ButtonPause\_Click(): Bu metot ile Durdur butonuna basıldığında görüntünün o anki frame’de durmasını sağladık. Bu işlemi yapmak için Timer’ın enabled özelliğini false yaptık.
* ButtonResume\_Click(): Bu metot ile Devam Et butonuna basıldığında görüntünün o anki frame’den devam etmesini sağladık. Bu işlemi sağlamak için Durdur butonunda devre dışı bıraktığımız Timer’ı enabled özelliğini true yaparak tekrar aktif hale getirdik. Görüntü oynatmasının bitip bitmediğini kontrol etmek için de progress isminde boolean değişken oluşturduk.

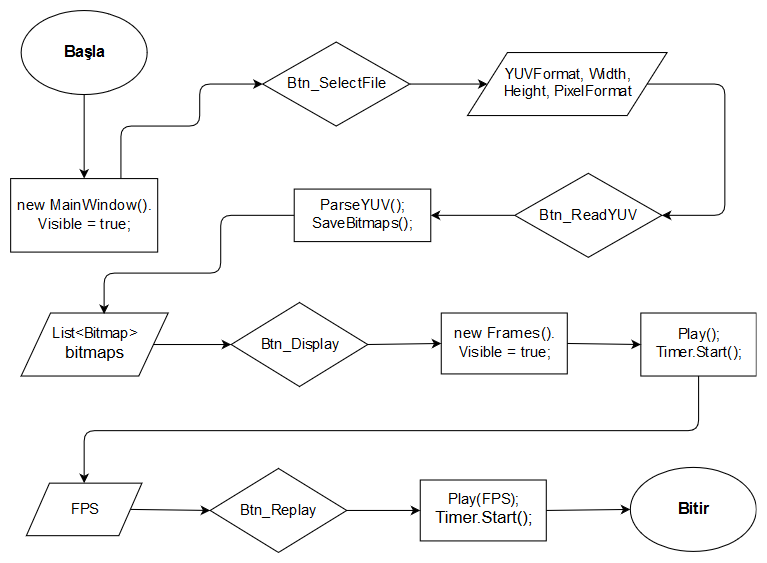
Son olarak, projeyi tamamladıktan sonra farklı bilgisayarda çalıştırdığımızda form ekranlarının bulanık göründüğünü fark ettik. Bunun Windows 10 işletim sisteminde çıkan DPI scaling ayarından kaynaklandığını anladık. Bu problemi çözmek için <https://stackoverflow.com/questions/4075802/creating-a-dpi-aware-application> adresindeki çözüm adımlarına bakarak Main metodunun bulunduğu Program classında Main metodunun içine,

if (Environment.OSVersion.Version.Major >= 6) { SetProcessDPIAware(); }

[System.Runtime.InteropServices.DllImport("user32.dll")]

kodunu ekledik. Böylece gerekli durumlarda (işletim sistemine göre) SetProcessDPIAware() metodunun çalışması için user32.dll Windows sistem dosyasını kullanmış olduk. Daha sonra Main ve Frames formlarının AutoScaleMode özelliğini Font yerine Dpi olarak değiştirerek bu sorunu çözdük.

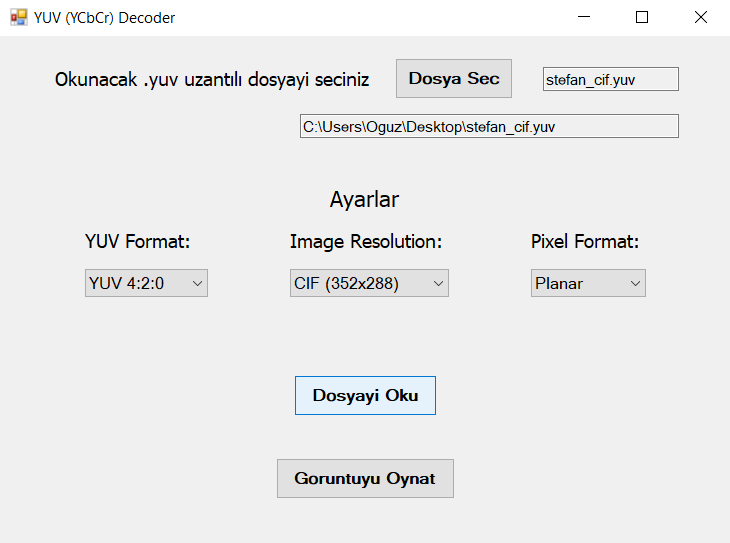
# Akış Şeması



*Şekil 2: Akış şeması*

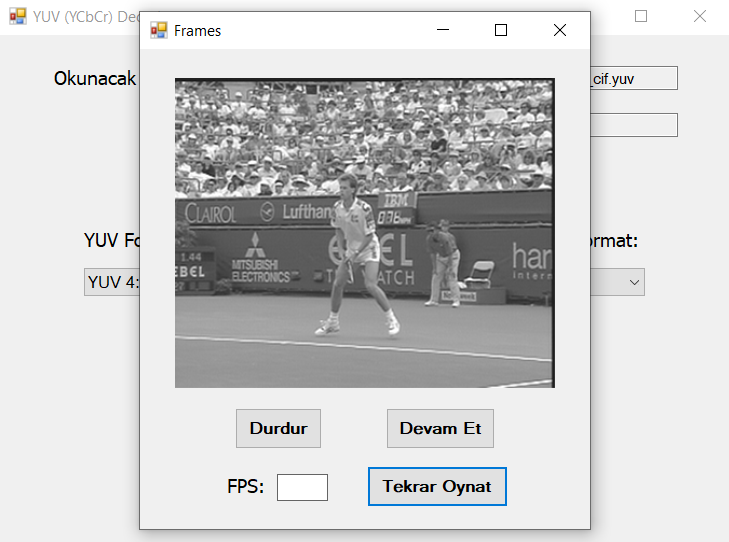
# Kullanıcı Kataloğu

Programı çalıştırınca ilk olarak dosya seçme, ayarlar, dosya okuma ve görüntüyü oynatma seçeneklerinin bulunduğu ana menü karşımıza geliyor.



*Şekil 3: Programın ilk çalıştırılması ve dosya okuma*

Dosyayı okuduktan sonra “Bitmap dosyaları başarıyla kaydedilmiştir, görüntüyü oynatabilirsiniz.” mesajı ekrana gelir. Görüntüyü Oynat’a basıldığında da Frames formu açılarak alttaki ekran gelir;



*Şekil 4: Frame by frame görüntü oynatılması*

Görüntü oynatma ekranında FPS (frame per second, saniye başına gösterilecek frame sayısı) ayarlanarak görüntü tekrar oynatılabilir. Durdur ve Devam Et seçenekleri de görüntü oynatılırken kullanılabilir. Eğer FPS girilerek görüntü oynatılma anında Tekrar Oynat butonuna basılırsa önceki görüntü oynatma işlemi durdurularak yeni FPS’e göre yeni görüntü oynatılır, bu işlemi sağlayabilmek için boolean tipindeki flag değişkenini tanımlayıp kullandık.

# Sonuçlar

Projeyi geliştirirken gereksiz kod tekrarları yapmamaya ve gereksiz yere bellekten alan ayırmamaya dikkat ettik, bu amaçla ReSharper Ultimate eklentisini kullanarak kod kalitesini arttırdık. Her aşamada kod yazarken algoritma karmaşıklığını ve programın hızlı çalışmasını hesaba kattığımız için yaptığımız projenin yeterince özgün olduğunu ve istenenleri fazlasıyla karşıladığını söyleyebiliriz. SaveBitmaps() metodu içinde Bitmap dosyalarını kaydederken SetPixels() hazır fonksiyonu yerine daha hızlı sonuç verecek bir yöntem bulup kullanabilirdik, onun dışında kafamızda soru işareti oluşturacak bir konu bulunmuyor.

# Kaynakça

[1] https://stackoverflow.com/

[2] <https://docs.microsoft.com/en-us/windows/desktop/medfound/about-yuv-video> (Erişim tarihi: 09.12.2018)

[3] <https://docs.microsoft.com/en-us/windows/desktop/medfound/recommended-8-bit-yuv-formats-for-video-rendering> (Erişim tarihi: 09.12.2018)

[4] <http://softpixel.com/~cwright/programming/colorspace/yuv/> (Erişim tarihi: 09.12.2018)

[5] <https://en.wikipedia.org/wiki/Chroma_subsampling> (Erişim tarihi: 14.12.2018)

[6] <https://www.dotnetperls.com/file-readallbytes> (Erişim tarihi: 14.12.2018)

[7] <http://videocodecs.blogspot.com/2007/05/image-coding-fundamentals_08.html> (Erişim tarihi: 16.12.2018)

[8] <https://www.rtings.com/tv/learn/chroma-subsampling> (Erişim tarihi: 17.12.2018)

[9] <http://www.ti.com/lit/an/spraak3b/spraak3b.pdf> (Erişim tarihi: 17.12.2018)

[10] <http://www.sunrayimage.com/faq.html> (Erişim tarihi: 18.12.2018)

[11] <https://scc.ustc.edu.cn/zlsc/sugon/intel/ipp/ipp_manual/IPPI/ippi_ch6/ch6_pixel_and_planar_image_formats.htm> (Erişim tarihi: 18.12.2018)

[12] <https://software.intel.com/en-us/ipp-dev-reference-pixel-and-planar-image-formats> (Erişim tarihi: 18.12.2018)

[13] <https://www.fourcc.org/yuv.php#Packed%20YUV%20Formats> (Erişim tarihi: 18.12.2018)

[14] <https://social.msdn.microsoft.com/Forums/vstudio/en-US/322df940-e277-4646-a177-113746dde364/rgb-byte-to-bitmap?forum=csharpgeneral> (Erişim tarihi: 19.12.2018)

[15] <https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.timer.interval?view=netframework-4.6.1> (Erişim tarihi: 19.12.2018)