GÖMÜLÜ YAZILIMLARDA ÇOK AMAÇLI GRAFİK ARAYÜZÜ: *VISION GDI*

Soner ÇINAR[1],Merve Özkardeş[1], Recep Bora ÇALIŞKANBAŞ[1],

Burak ÜNALTAY[1]

[1]Gömülü ve Gerçek Zamanlı Yazılım Tasarım Müdürlüğü, SST Sektör Bşk.
ASELSAN A.Ş.

{scinar, mozkardes,bcaliskanbas, bunaltay}@aselsan.com.tr

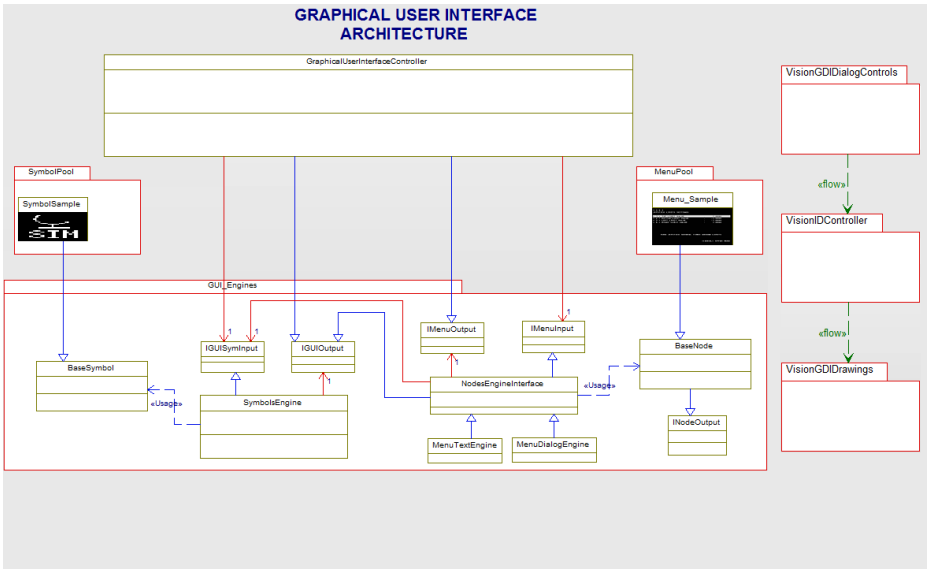
Özet. Gömülü sistemlerde geliştirilen projelerin sürekli değişen grafik altyapı ihtiyaçlarını en etkili şekilde karşılamak amacıyla Çok Amaçlı Grafik Arayüzü: *VisionGDI* tasarlanmıştır. *VisionGDI* temelde kullanıcı arayüzü, menü ve sembol motorları ve grafik çizim yeteneklerine sahiptir. Bu sayede projelerde kullanılacak kullanıcı arayüzlerinin her türlü otomasyon ve altyapısı hızlı bir şekilde sağlanabilmektedir. İçerisindeki katmanlar farklı yetenekleri gerçekleştirmek üzere tasarlanmıştır. Kullanıcı arayüzü katmanı da grafik arayüzünü kullanarak kullanıcı arayüz objelerini (diyalog kutuları, metin kutuları) oluşturur. Grafik çizim katmanı 2 boyutlu grafiklerin ve textlerin çizilmesi yeteneklerini kapsar. Bu çalışmada tasarlanan katmanların beraber çalışarak grafik desteği olmayan gömülü sistemlerde 2 boyutlu grafik ve menü gibi görsel objeleri oluşturması anlatılacaktır.

Anahtar Kelimeler: Semboloji, Menü, Gömülü Yazılım, Yeniden Kullanım, Grafik Arayüzü,Kullanıcı Arayüzü

1 Giriş

Literatürde gömülü sistemler için mevcut grafik arayüzü çözümleri olarak WindML(WindRiver Media Library) ve QT'dir. QT hem gömülü platformlarda hem de Windows üzerinde çalışabilen bir üründür. WindML ise sadece vxWorks işletim sistemi ortamında çalışmak üzere tasarlanmıştır.
QT oldukça geniş kapsamlı grafik arayüzü çözümleri sunabilen bir ürün olmasına rağmen vxWorks 6.9 ve sonrası vxWorks sürümleri için destek verdiğinden gömülü yazılım üreten bizim gibi ekipler için bir opsiyon olmaktan çıkmıştır[1]. Ayrıca vxWorks 6.9 sürümü ve sonraki sürümler için bir çözüm olarak düşünüldüğünde dahi geliştirici başına 5000 Amerikan Doları gibi bir lisans ücreti gerektirmesi diğer bir olumsuz yönüdür[2].
WindML ürünü sadece vxWorks gerçek zamanlı işletim sistemli ürünler için grafik arayüzü çözümleri sunan bir üründür. QT ürününde olduğu gibi geliştirici başına

getirdiği lisans ücretinin yanı sıra ürünün sunduğu özellikler platformdan platforma değişiklik göstermekte, her donanım konfigürasyonunda her özellik kullanılamamaktadır. Ayrıca kapalı kaynak kodlu bir yazılım olduğundan, ürün proje gereksinimlerine göre değiştirilememekte ve optimize edilememektedir[3].
Bizim projelerimizde ise bir grafik arayüzünden beklenen öncelikle sistemlerin değişen ihtiyaçlarını çabuk ve etkili bir biçimde karşılaması, bütün VxWorks versiyonları ile uyumlu olması ve hatta Windows da da çalışabilmesi, kullanıcı dostu ve kolay anlaşılabilir kod geliştirilebilir olması olmuştur. Bu yüzden Vision GDI, silah sistemleri ve görüntü işleme projelerinde ihtiyaç duyulan grafik arayüzü yeteneklerini karşılamak amacıyla tasarlanmıştır. Ayrıca tasarımında yeniden kullanabilirlik ve verimli hafıza kullanma gibi özellikler temel alınmıştır. Vision GDI, kullanıcı arayüzü katmanı, menü ve sembol motorları katmanı ve grafik çizim katmanı olmak üzere üç ana katmandan oluşmaktadır. Mimariye ilişkin detaylı gösterim Şekil 1'de mevcuttur.



Şekil. 1 : Vision GDI Mimarisi.

2 Kullanıcı Arayüzü Katmanı

Kullanıcı arayüz katmanı, kullanıcı isteklerini sembol ve menü motorlarına aktaran katmandır. Kullanıcı isteklerine örnek olarak; menü oluşturma, sembol çizdirme, yer değiştirme, boyut güncellemeleri ve mevcut menülerin içeriklerinin değiştirilmesi verilebilir. Buradaki amaç kullanıcının minimum seviyede bilgi kullanarak sadece isteklerini belirtmesi suretiyle kolay menü ve sembol oluşumunu sağlamasıdır. Bu

yüzden kullanıcı çizdireceği sembolün detayları hakkında bilgi sahibi olmak zorunda değildir. Bu özellik Vision GDI'ın nesne yönelimli tasarımının bir sonucudur.

3 Sembol ve Menü Katmanı

Bu bölüm sembol ve menü motorları ve havuzları olmak üzere iki ana yapıdan oluşmaktadır. İlerleyen bölümlerde bunlar ile ilgili detaylar verilmektedir.

3.1 Sembol ve Menü Motorları

Sembol ve menü motorları kullanıcı arayüzlerinin oluşturulabilmesi için her türlü otomasyon ve altyapıyı oluşturur. Menü motoru sembol motorunun yeteneklerini de kullanır ve bu yeteneklerle sembol tabanlı tüm menüler için gerekli altyapıyı oluşturur. Bu motorlar tek başına kullanılabilen, belli bir platforma veya işletim sistemine bağımlılığı olmayan, farklı geliştirme ortamlarıyla uyumlu, geliştirilebilir ve konfigüre edilebilir motorlardır.

3.1.1 Sembol Motoru

Sembol motoru, sembol sınıflarının koordineli bir şekilde ekranda gösterilmelerini sağlar. Bütün sembol sınıfları, BaseSymbol sınıfından türetilir ve bu sınıfın yeteneklerini kullanır. Her sembolün özellikleri ilgili sınıfın yapıcısında ilklenir. Bu özelliklere örnek olarak; sembol pozisyonu, sembol adı, sembol içeriğinin sabit ya da güncellenebilir olması, sembolün menüdeyken aktif olup olmaması, sembolün yanıp sönebilmesi, sembolün belirli bir süre ekranda kalması ve pozisyonunun değişebilmesi verilebilir.

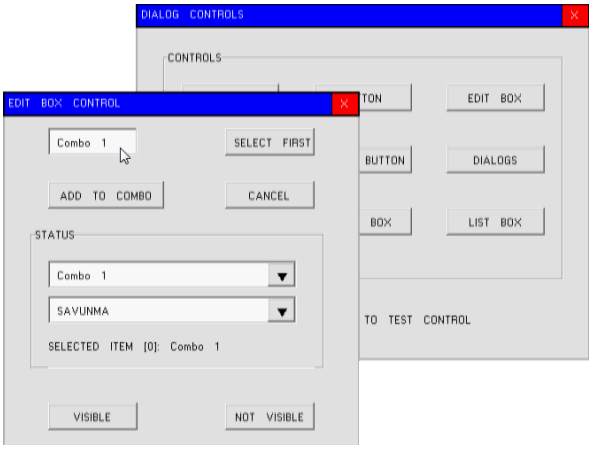
Sembol motoru, sembol havuzundaki sınıfları BaseSymbol sınıfı üzerinden görür. Sembol motoru sistemin durumunu kullanıcı arayüzüne yansıtmak ve sonuçlarını kullanıcı sınıfa iletmek amacıyla IGUIInput ve IGUIOutput arayüz sınıfları ile durum değişikliklerini yapar. Sembolojide gösterilen bir sembolle ilgili verilerin değişmesi durumunda da bu değişikliği ilgili sembol sınıfına bildirir. Fakat sembolün ne zaman çizilip silineceğine sembol motoru karar verir.

3.1.2 Menü Motoru

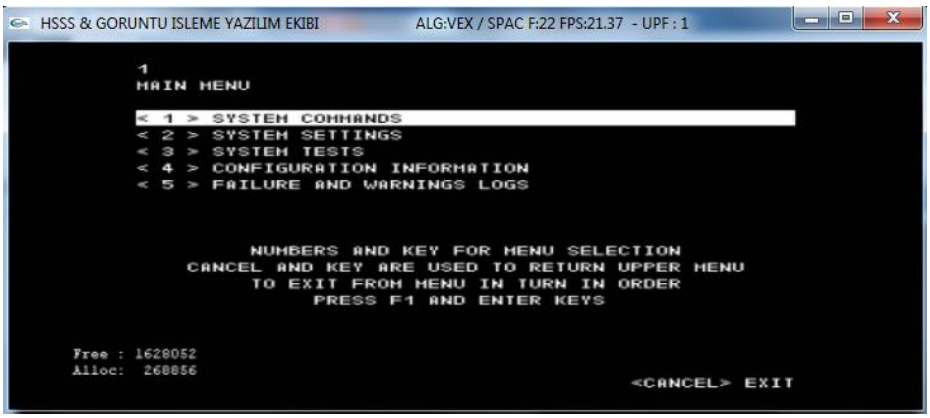
Bu seviyede diyalog tabanlı ve sembol(text) tabanlı olmak üzere iki farklı menü motoru vardır. Bunların örnekleri şekil 2 ve 3'te görülebilir. Bu iki menü motoru isteğe göre farklı kullanıcı arayüzleri sağlar.
Menü motoru, menü sınıflarının koordineli şekilde ekranda gösterilmesini ve sistem ayarlarının yapılabilmesini sağlar. Menü hiyerarşisi ağaç yapısı şeklinde belirlenir. Menü ağacının konfigüre edilebilir olması sayesinde kullanıcı menülerinin yerlerini belirleyebilir. Aynı zamanda, bir menü sınıfı menü ağacında birden fazla yerde kullanılabilmektedir.

Bütün menü sınıfları, BaseNode sınıfından türetilir ve bu sınıfın yeteneklerini kullanır. Menü motoru, menü sınıflarını INodeOutput arayüz sınıfı ile koordine eder. Menü sayfasının aktivasyonunun ve deaktivasyonunun sağlar. Bunun yanında kullanıcının sınıftan aldığı sistem durumlarının ve kullanıcı komutlarının aktif menü sınıfına iletilmesini sağlar.

Kullanıcı, menü sınıflarını doğrudan görür ve yaratır. IMenuInput arayüzü sınıfında bulunmayan ihtiyaçlarını, doğrudan menü sınıfına yaptırabilir. Bu durumda, menünün aktif olup olmadığı gibi kontrollerin menü motorunda değil, menü sınıfında ya da kullanıcı sınıfta yapılması gerekir.



Şekil 2 : Diyalog tabanlı menü örneği



Şekil 3: Text tabanlı menü örneği

3.2 Sembol ve Menü Havuzları

Sembol ve menü havuzları, sembol ve menü sınıflarını barındıran havuzlardır. Sembol ve menü motorları tarafından sağlanan altyapıyı kullanırlar. Her bir sembol ve menü ekranı projeden tamamen soyutlanmış ayrı ayrı sınıflardan oluşmaktadır. Sınıfların kendi ilk konfigürasyonları olmasına rağmen, istendiği takdirde kullanıcı tarafından da yapılandırılabilirler.

3.2.1 Sembol Havuzu

Sembol havuzu, sembol sınıflarını içinde barındıran havuzdur. Her bir sembol sınıfı, çizim ile ilgili olan tüm yeteneklerini, türemiş olduğu BaseSymbol sınıfını kullanarak gerçekler.

Bununla beraber her bir sembolün ayrı ayrı çalışma özellikleri olabilir. Bu özellikler; sembol pozisyonu, sembol adı, sembol içeriğinin sabit ya da güncellenebilir olması, sembolün menü aktifken ekranda olup olmaması, sembolün yanıp sönebilmesi, sembolün belirli bir süre ekranda kalması, pozisyonunun değişebilmesi ve sembolün sürekli değişip değişmediğidir. Her sembol sınıfının davranışı birbirinden bağımsızdır. Her sembol sınıfı, bu özellikleri kendi yapıcılarında ilklerler. Ancak konfigürasyon alt yapısı sayesinde, proje seviyesindeki istek ve taleplere göre yeniden yapılandırılabilirler. Bundan dolayı, kullanıcı arayüzü yeniden kullanılabilirliğe oldukça elverişlidir.
Sembol havuzundaki sembol sınıfları, sembol motoru tarafından BaseSymbol sınıfı üzerinden görülür. Sembolojide gösterilen herhangi bir değerin değişmesi durumunda, bu değişikliği sembol havuzunda bulunan ilgili sınıfa bildirilir. Fakat sembolün ne zaman çizilip silineceğine Semboloji Motor'u karar verir. Bu sayede soyutlama yapılmış olur.

3.2.2 Menü Havuzu

Menü havuzu, menü sınıflarını içinde barındıran havuzdur. Her bir menü sınıfı menülerle ilgili tüm yeteneklerini, türemiş olduğu BaseNode sınıfını kullanarak gerçekleştirir.

Menü sınıfları menü motoru tarafından aktive edilir. Aktive edilen menü sınıfı, oluşturacağı menüyü nesne yönelimli yaklaşıma göre tasarlanmış INodeBase sınıfı üzerinden oluşturur. Bu aşamada int, char, float, double, struct gibi temel veri yapılarının kullanımı engellenmiştir. Menü sınıflarında kullanıcıya gösterilen veya üzerinde değişiklik yapılan her parametre nesne yönelimli yaklaşımlara göre hazırlanmış değişkenler ile ifade edilmektedir. Bu sayede menü sınıflarında parametre girişi, sınır kontrolü, kullanıcıya verilen uyarılar, dönüşümler, hizalama, ondalık hassasiyeti ve gösterim modu menü motorunun alt sınıfları tarafından otomatik olarak sağlanmaktadır.

Benimsenen nesne yönelimli yazılım sayesinde menü havuzundaki menü sınıfları kullanılan menü motoruna göre farklılık göstermez. Menü sınıfları menülerin yazı ya da diyalog tabanlı olmasından etkilenmez, yazılımı farklılık göstermez. Bu kısım menü motorunun sorumluluğundadır. Menü sınıfları menüleri belirler, oluşturur ancak kullanıcı arayüzünün nasıl olacağıyla ilgilenmez. Nesne yönelimli yaklaşım sayesinde her bir menü motoru için aynı menü sınıfının farklı varyasyonlarına ihtiyaç duyulmaz.

4 Grafik Çizim Katmanı

4.1 VisionGDI Drawings

Grafik çizim katmanı olarak adlandırılabilir. 2 boyutlu grafiklerin ve textlerin çizilmesinden sorumludur. Şekillerin çizilmesi, yazıların yazılması, text fontları gibi yetenekler bu katmanda yer alır. Ayrıca şekillerin içlerinin doldurulması, yatay, düşey veya merkeze göre simetriğinin alınması ve renklerin tersine çevrilmesi gibi yetenekler de bu katmanda kullanılır. Bu katman sayesinde pek çok font, italik yazma, pek çok farklı şekli çizebilme gibi yetenekler kullanıcıya sunulur.

Üst katmanlar yapılmasını istedikleri bütün çizim işlemlerini VISION GDI Drawings aracılığıyla yaptırırlar. Basılacak ekranı doldurma işlemi burada gerçekleşir. Üst katmanlardan gelen istekler doğrultusunda hareket eden VISION GDI Drawings katmanı, istenen resmi oluşturduktan sonra bunu bir render'a (openGL, openCL vs.) iletir ve kullanıcı ekranına basılmasını sağlar.

VISION GDI Drawings katmanı ihtiyaç doğrultusunda, bunları en iyi şekilde karşılamak amacıyla oluşturulmuştur. Farklı platformlarda sorunsuz çalışabilmesi sağlanmıştır. Farklı platformlarda çalışabilir olması sayesinde platformdan bağımsız olarak kullanıcının ihtiyaçlarını her ortamda ve her projede karşılayabilecek düzeydedir. VISION GDI Drawings katmanı ihtiyaç duyulmayan özellikler eklemekten kaçınılması sayesinde kullanılmayacak öğelerden arındırılmıştır. Yetenek kalabalığındansa ihtiyacı tam karşılayacak ve hafıza, zaman gibi her türlü kaynağın gereksiz kullanımından kaçınacak, en optimize şekilde ihtiyaca karşılık verecek şekilde tasarlanmıştır.

Gerçek zamanlı olarak çalışan sistemlerde zaman en önemli kısıtlamalardandır. Kullanılabilecek hafıza da kısıtlı olduğu için hafızayı verimli ve efektif kullanmak zorlayıcı bir diğer konudur. Bu nedenle zamanın ve kullanılan hafızanın iyi yönetilmesine, VISION GDI Drawings'in olabildiğince optimize olmasına çok dikkat edilmiştir.

4.2 VisionID Controller

VisionID Controller paketi, 2 boyutlu grafiklerin, text objelerinin ve 2 boyutlu animasyonlu grafik objelerinin kontrollerinin yapılması ve bu grafikleri gösterecek yapıya gereken komutların verilmesi amacıyla kullanılmaktadır.

Paket hâlihazırda bulunan grafik objelerine erişerek bunları sembol olarak kullanmaya izin vermekle beraber; hazır sembollerin modifikasyonuyla üretilen ya da Vision GDI'ın grafik yetenekleri kullanılarak baştan üretilen grafik objelerinin sembol olarak kullanılmasına imkân sunmaktadır. Mevcut semboller farklı isimlerde, birden çok kez ve farklı pozisyonlarda; ek bir hafıza kullanımına ve kopyalama işlemine gerek kalmadan kullanıcıya sunulmaktadır. VisionID Controller'ın bu yetenekleri nesne yönelimli yazılım sayesinde kazandırılmıştır.

Gömülü sistem yazılımlarında hafıza kullanımı önemli bir nokta olmakla beraber VisionGDI'ın genelinde olduğu gibi VisionID Controller paketinde de bu konuya önem gösterilmektedir. Sistemin dizininde bulunan semboller ve kullanıcı tarafından tanımlanan semboller iki ayrı yapıda tutulmaktadır. Dizinde bulunan semboller sistem açılışında yapılara aktarılır, kullanıcı tanımlı semboller ise sistemin çalışma sürecinin herhangi bir anında eklenebilir ya da çıkarılabilirler. Bunlarla beraber, VisionID Controller sembollerin birbirleriyle etkileşimi olmayan katmanlara çizilebilmesini sağlamaktadır ve bu katmanların kontrolünden sorumludur. Özet olarak VisionID Controller, Vision GDI'ın verdiği PNG, Bitmap, APNG, JPEG gibi medya destekleri sayesinde oluşturduğu sembollerin yönetimini ve kontrollerini yapan birimdir.

5 Sonuç

Bu makalede gömülü sistemlerde çalışan, çok amaçlı grafik arayüz yeteneklerini karşılayan Vision GDI yazılımı anlatılmıştır. Yazılımın genel bileşenlerinden bahsedilmiş ve mevcut yetenekleri açıklanmıştır. Vision GDI, sistemlerin şu anki ihtiyaçlarını tamamen karşılamakla beraber, halen geliştirilmekte olan bir yazılımdır.
Vision GDI yazılımı her platformda çalışabilecek bir biçimde tasarlanmıştır. QT'nin aksine bütün vxWorks sürümleriyle uyumludur. Her platformda çalışabildiği için Windows ortamında debug yeteneği sağlamaktadır. Kompleks sembol ve menüler kullanıcı dostu geliştirme arayüzü sayesinde birkaç satırda gerçeklenebilmektedir. Kaynak koduna erişim sayesinde Vision GDI, her platforma optimize çözümler sunabilmektedir.

Kaynaklar

1. Jessica Miller, Wind River and Digia Collaborate to Extend Qt Commercial Support for VxWorks, http://www.windriver.com/news/press/pr.html?ID=10222, 2012.
2. http://www.qt.io/buy-professional-step-2/.
3. WindML SDK Programmer's Guide, 3.0,2002