

C++ Grafik Kullanıcı Arabirimlerinde Taşınabilirlik

Serdar TUĞCU, Mete Alpaslan KATIRCIOĞLU

Portakal Teknoloji

serdar.tugcu@portakalteknoloji.com, mete.alpaslan@portakalteknoloji.com

Özet; Grafik arayüzleri tasarlamak ve programlamak, geliştirilen yazılımın kullanılması açısından en önemli konulardan biridir. Arayüz programlarken dikkat edilmesi gereken noktalar, istenilen görünümü elde edebilmek, arayüzle programın iletişimini kolay sağlamak, hızlı çalışması ve farklı işletim sistemlerinde kullanılabilir olmasıdır. Bildirimizin temel konusu taşınabilirlik, yani farklı işletim sistemlerinde çalışabilen ara birimler tasarlayabilmektir. Taşınabilirliğin temel prensiplerinden yola çıkarak yaygın olarak kullanılan arayüz programlama araçlarını karşılaştırdık, kendi projelerimizden elde ettiğimiz tecrübeleri aktardık, teknik ve pratik örneklerle sonuçları sunduk.

Anahtar Kelimeler: C++, Arayüz Programlama, Taşınabilirlik, Qt, Wxwidgets

Portability in C++ Graphical User Interfaces

Abstract: Programmers are people who have analytical thinking ability. They focus on problem solving and algorithm finding. This may be the reason why they usually do not like interface programming. But on the contrary, although it may be built on a very good designed software architecture, a program with a weak interface will not be accepted by the users.

C++ and Java are the most popular languages for interface programming. There are some tools for interface programming with C++. While choosing one of these tools, programmers pay attention mostly on easiness in designing, and portability. Two of the tools, which we also used for our projects, are Qt and wxWidgets. The use of these tools, comparison of them and explaining portability issues are the main topics of this paper.

Key Words: C++, Interface Programming, Portability, Qt, Wxwidgets

1. Giriş:

Arayüz hazırlamak sadece teorik yazılım bilgisinin ve yazılımın gerektirdiği yeteneklere sahip olmanın tek başına yeterli olmayacağı bir iştir. Geliştirdiğiniz yazılımın kullanıcı tarafından nasıl en rahat şekilde kullanılacağını tahmin etmeli, hazırladığımız pencerele- rin, butonların ve benzeri arayüz elemanları- nın mantıklı ve göze hoş görünen bir dizayn içinde olmasını sağlamalısınız. Üstelik işiniz bununla da bitmez. Hedef kitleniz sadece belli bir işletim sistemi kullanıcılar değilse, programınızı farklı işletim sistemlerine rahatlıkla entegre edilebilir özellikte olmalı. Küçük çap-

lı programlarda bunu başarmak görece daha kolay olsa da projenin kapsamı büyüdükçe ve siz standart kütüphaneler yanında başka kütüphaneler de kullanmak zorunda kaldıkça bu iş zorlaşır. Arayüzler de belirli kütüphanelerle çalıştığı için, bir işletim sisteminde hazırladığımız arayüzü diğer bir işletim sistemine entegre etmek zor bir iş haline gelir.

Bu sorunu çözmek için arayüz hazırlamak üzere kullanılacak araçlar iyi seçilmeli. Sektörde özellikle iki araç bu amaca yönelik yazılım geliştiren profesyonellerce tercih edilmektedir. Bunlar Qt ve wxWidgets'tir. Qt daha kolay olması, geçmişten günümüze daha dikkatli bir

şekilde gelişmesi ve ticari hamlelerini de daha akıllıca atması sonucunda wxWidgets'a göre bir adım önde görünmektedir. Fakat wxWidgets da kendince üstün özelliklere sahip olduğu için gittikçe artan bir kullanım yaygınlığı kazanıyor.

Bu bildirgenin amacı, arayüz hazırlamak konusunda bazı teorik ve pratik bilgiler vermenin yanında, özellikle bu iki aracı karşılaştırmak, yazılımcının önceliklerine göre hangi tercihi seçmesi gerektiği konusunda veriler sunmaktır.

2. Kullanıcı Ara birimi Programlama nedir?

Microsoft Windows, Apple Macintosh, OS/2 ve UNIX altındaki X Windows işletim sistemleri kullanıcı ara birimli işletim sistemleridir. Bunlar DOS ya da UNIX gibi yazı tabanlı sistemler değil, çok-işli ve özellikle UNIX için aynı zamanda çok-kullanıcılı sistemlerdir. Bu işletim sistemlerinde görüntü, klavye, fare, disk sürücüler ve iletişim portları programlar tarafından kullanılır. Bu ara birimleri kullanan programlar da kullanıcı ara birim programlarıdır. Bu programları geliştirme işi de kullanıcı ara birimi programlama olarak adlandırılabilir. Kullanıcı ara birimi programlama, pencereler, düğmeler, kontroller gibi görüntülerin oluşturulup, kullanıcıların programla iletişimini sağlanması işini yapar.

2. Ara birim Programlama Modeli

Ara birim programlama modeli, komut satırı programlarından farklı bir modeldir. Kullanıcı her an bir düğmeye basabilir, bir metin kutusunu doldurabilir ya da başka herhangi bir nesneyi kullanabilir. Programın bunu anlaması ve algılaması için olay yapısını kullanması gerekir. Kullanıcı arayüzden herhangi bir iş yaptığında, bir olay tetiklenir. Bu olay, yazılan kodlarla değerlendirilir ve sonucu gösterilir.

2. Taşınabilirlik Nedir?

Yazılımın yaygınlaştığı ilk dönemlerde, firmalar sistemlerini tek tip yapma, yani bütün makinelerinde tek bir işletim sistemi kullanma eğilimindeydiler. Çünkü homojen bir sistemin birçok problemi ortadan kaldıracağına inanılırdı. Fakat zamanla ihtiyaçlar arttı ve görüldü ki her ihtiyaç için en uygun olan sistemler farklıdır. Bu da homojen sistem kurma merakını geride bırakıp ihtiyaca göre sistemler kullanma gereksinimini ortaya çıkardı. Fakat bu gereksinimi karşılamamanın bir bedeli vardır: Taşınabilir programlar.

Taşınabilir program yazmak demek, farklı sistemlerde sorunsuz çalışabilen yazılımlar geliştirmek demektir. Eğer sadece veritabanı ya da ağ programlama gibi bir iş yapıyorsanız belli standartlara uyararak bu taşıma işlemini rahatlıkla çözebilirsiniz. Fakat hemen hemen bütün yazılımlar bir kullanıcı arayüzüne sahiptir ve taşınabilirlik konusunda en çok dikkat edilmesi gereken konulardan biri, hazırlanan kullanıcı arayüzünü farklı işletim sistemlerinde çalışabilir hale getirmektir.

Taşınabilirlik, yazılımın değerini arttırdığı gibi, kullanan kişi sayısı ile orantılı olarak ömrünü de arttırır. Temel olarak taşınabilirlik, bir sistemde düzgün olarak çalışan yazılımı, başka bir sistemde de çalışabilir hale getirmektir. Bu da her sistem için sıfırdan yazılım geliştirmeyle karşılaştırıldığında çok daha ekonomik bir durumdur.

Taşınabilirlikte yazılımın bütün bölümleri dikkate alınmak zorunda değildir. Sadece ortam değiştiğinde değişmek zorunda kalan kısımların (yerel kütüphaneler gibi) değiştirilmesi yeterli olmalıdır. Yazılımınızın sisteme bağımlı kısımlarını belirlemeli, hedeflediğiniz yeni sistem için hangi değişikliklere ihtiyacınız olduğunu ortaya koymalı ve bunun yapılabilirliğini ölçmelisiniz. Cevaplanması gereken

sorular şunlardır:

1. Hangi sınıflarınız sisteme bağlı olarak değişebilir?
2. Değişik sistemlere göre bu sınıflar ne kadar değiştirilmelidir?
3. Bu değişikliğin getireceği ekstra maliyet ne kadar kabul edilebilir?
4. Değişikliklerin performans getireceği olumsuz etki ne kadar kabul edilebilir?

Bu sorular, taşınabilirliğin gerekliliğini ve bize ne ölçüde faydalı olacağını göstermek için başvurduğumuz sorulardır. Eğer sorular size yazılımınızı taşımamanın yeni sistemler için yeniden yazılmasından daha pahalı olacağını gösteriyorsa, büyük ihtimalle ilk tasarımınızda taşınabilirliği göz önünde bulundurmamışsınızdır.

Taşınabilirlik konusunda belirli kavramlar vardır. Bunlar:

- Yazılım birimleri
- Çevre
- Çevreye bağlı sınıflar
- Taşınabilirliğin derecesi
- Maliyet ve kar
- Taşınabilirliğin evreleri

4.1 Yazılım birimleri

Bir yazılım birimi, bir bileşen, program, alt sistem ya da sistem olabilir.

4.2 Çevre

Çevre, yazılım birimlerinin iletişim halinde olduğu önemli harici elemanların toplamına verilen isimdir. Çevre ifadesi genellikle işletim sistemi ve ilgili donanımları ifade etmek için kullanılır.

4.3 Çevreye bağlı sınıflar

Taşınabilirlikte yapılması gerekenler her hedef platform için farklı kodlar yazmak değil,

platformun kullanacağı sınıfları belirlemek ve sadece bu sınıfları uygun hale getirmektir.

4.4 Taşınabilirliğin derecesi

Taşınabilirlik, bir yazılımı farklı platforma taşımakla farklı platformlar için yazılımı yeniden inşa etmek arasında bir seçim gerektirir. Bu seçimin yapılabilmesi için de belirli metrikler kullanarak ölçümler yapılmalıdır. Taşınabilirliğin derecesi de bu ölçümler sonucu belirlenir.

4.5 Maliyet ve Kar

Bir yazılımın taşınabilir bir yazılım olmasının yazılımcıya getireceği karın yanında karşılamak zorunda olduğu bir maliyeti de vardır. Maliyeti düşünürken, taşınabilirlik sağlamak uğruna fazladan harcanan zaman, performans kaybı, sisteme özel avantajların bir kısmından vazgeçmek gibi konular düşünülme zorundadır. Kar ise gelecekte daha az çaba harcayarak farklı platform kullanıcılarına hitap etme ve güvenilirlik istekleri varsa elde edilir.

4.6 Taşınabilirliğin evreleri

Taşınabilirliğin iki evresi vardır. Taşıma ve adaptasyon. Taşıma; fiziksel olarak bir ortamdan diğer ortama geçiştir. Adaptasyon ise yeni ortamda çalışabilmesi için gereken ve genellikle derleyici tarafından yapılan değişiklikler ve modifikasyonlardır.

4.7 Taşınabilirlik yöntemleri

Arayüzlerin taşınabilirliği konusunda çeşitli yöntemler kullanılmıştır. Bunlardan bazıları aşağıdaki gibidir:

- **Çalıştırılabilir Emülasyon:** Bu yöntemle, uygulama yeniden derlenmeye gerek duymadan farklı bir ortamda çalışabilir. Bu da bir yazılım emülatörünün donanım ve yazılımı taklit etmesiyle mümkündür.

- **Katmanlı Araçlar:** Katmanlı bir araçla, geliştirici uygulamalarını Uygulama Programlama Ara birimi(API) kullanarak

hazırlar, ve bu araçlar hedef platformun API'lerini çağırır. Başka bir ifadeyle, bu yaklaşım uygulamayla platform arasında bir katman kullanır ve bu katman bir API-den diğerine geçişi sağlar.

Bu yaklaşımın en önemli avantajı, uygulamanın hedef platformlara özgü yazılmış gibi davranabilmesidir. Platform değişikliğinde tek yapılması gereken, yeni platformda uygulamanın tekrar derlenmesidir. Dezavantajı ise uygulamaların platformlardaki ortak özelliklerle kısıtlı olmak zorundalığıdır.

- **Emülasyon araçları:** Bu yaklaşımın katmanlı araçlar yaklaşımıyla ortak özelliği, platformlar için aynı API'leri kullanıyor olmasıdır. Farkı, ve avantajı, ise hedef platformların görünüşünü emüle edebilir. Böylece kullanılan ara birim elemanının platformda yerel olarak bulunmuyor olması bir sorun oluşturmaz. Fakat buna rağmen hedef platformda yerel olarak bulunmayan bir arabirim elemanının düzgün görüntülenebileceği garanti edilemez.
- **Taşınabilir uygulama çatıları:** Taşınabilir uygulama çatıları, önceki iki grupta olduğu gibi kendi API'lerini kullanır. Buna ek olarak da çıktı alma ve dosya yönetimi gibi bazı özelliklerin de kullanılmasına izin verir.
- **Taşınmış API'ler:** Bu yaklaşımı kullanan araçlar, bir platformun yerel API'lerini diğer platforma taşırlar. Böylece Platforma özgü özelliklerin farklı platformlarda sorun oluşturması olasılığı ortadan kalkar. Fakat bu işi yapabilen bir araca ihtiyaç olduğu için, en pahalı çözümlerden biri taşınmış API kullanmaktır.

4.7 Taşınabilirliğin kuralları:

Aşağıda anlatılanlar, taşınabilirlik konusunda dikkate alınması gereken kurallardır.

- **Modüler programlar:** Bir kodu taşımak için, kodun iyi bir şekilde bölümlenmiş olması gerekir. Arayüz sınıflarıyla diğer sınıfların ayrılması, platforma özgü kodların ayrı şekilde yazılarak değişikliklerin kolaylaştırılması gibi modüler yaklaşımlar, taşıyacak kodun yönetilmesini ve hataların ayıklanmasını mümkün kılar.

- **Derleme kontrolü:** Bir platformda derleme sırasında sadece uyarı veren kod, başka bir platformda düzeltilmesi zor bir sorun haline gelebilir. Bu nedenle de özellikle derleyicinin verdiği uyarıların dikkatle incelenmesi gerekmektedir.

- **Platforma özgü özelliklerden kaçınma:** Farklı platformlarda özellikle görüntü oluşturma konusunda sıkıntı yaşamamak için, platforma özgü özellikler yerine her platformda bulunan ortak özellikler kullanılmalıdır.

- **Erken ve sık taşıma:** Eğer geliştirici uygulamasının farklı platformlarda çalışmasını hedefliyorsa, henüz geliştirme sürecindeyken hedef platformların her birinde derleme ve çalıştırma süreçlerini denemeli ve hataları erkenden fark etmelidir.

2. C++ Grafik Kullanıcı Ara Birimlerinde Taşınabilirlik:

Grafik arayüzü programlayan geliştiriciler çeşitli programlama dilleri kullanmakla beraber, iki programlama dili ön plana çıkar. C++ ve Java. Bu ikisi arasında seçim yapılırken de Java'nın geniş sınıf kütüphaneleri ile C++'ın performans avantajı arasında bir tercih yapılır.

C++ platforma bağlı olarak arayüz programlama için çeşitli araçlar ve kütüphaneler kullanma imkanı sunar. Windows için MFC, Linux için KDE kütüphaneleri bunlara örnektir. Araçlar için ise Microsoft Visual Studio, Qt,

wxWidgets ön plana çıkan örneklerdir. Her üç araç da aynı zamanda kendi kütüphanelerine ve çalışma sistemlerine sahiptir. Taşınabilirlik söz konusu olduğunda ise özellikle iki araç ön plana çıkar: Qt ve wxWidgets.

a.Qt

Qt, çok-platformlu bir C++ uygulama geliştirme çatısıdır. Arayüz geliştirme, veri tabanı, XML, OpenGL, ağ uygulamaları gibi birçok alanı içeren, 400den fazla sınıfa sahiptir.

Birçok modüle ayrılmış olan Qt API'si, çekirdek sınıfları, GUI sınıfları, SQL veritabanı sınıfları, XML sınıfları, Ağ sınıfları, OpenGL 3B sınıfları ve daha birçok sınıf içerir.

Qt 4 temel araca sahiptir: Qt Designer, Qt Linguist, Qt Assistant, qmake

- **Qt Designer:** Qt Designer, sürükle-bırak yöntemiyle çalışan bir yerleşim ve form geliştirme aracıdır. Qt'nin yaygın kullanılmadığı en önemli sebeplerden biri, Qt Designer'in Microsoft Visual Studio .Net platformuyla olan benzerliğidir. Bu araç, hızlı ve kolay bir şekilde arayüz hazırlama olanak sağlar.

- **Qt Linguist:** Bu araç, geliştirilen yazılımın farklı konuşma dillerine destek sağlanması için tasarlanmıştır. Bütün diller için destek sağlar.

- **Qt Assistant:** Qt ile ilgili en kapsamlı yardım kaynağı olan bu araçta, bütün sınıfların açıklaması, örnek kodlar gibi programcıya yol gösterecek bilgileri barındırır.

- **qmake:** Bu araç, yazılımın bağımlılık listesini oluşturmak, platforma özel derleme yöntemlerini hazırlamak gibi işleri otomatikleştirerek, taşınabilirliği daha kolay hale getirir.

Qt Uygulama platform ara birimi her platform için aynıdır. Dolayısıyla, hangi platformda çalışılırsa çalışılsın, öğrenilmesi gereken sadece bir uygulama platform ara birimidir. Platformun kendi ara birimiyle uğraşmak Qt ara biriminin işidir. Bu hem kodlama kolaylığı, hem de performans avantajı sağlar. Ayrıca Qt sadece arayüz programlama için kullanılan bir araç değildir. Qt Console Edition, arayüzden bağımsız yazılımlar geliştirmek için tasarlanmıştır.

Qt'nin en önemli farkı, olay mekanizmasıdır. Qt, MFC'nin yaptığı gibi bir mesaj haritası kullanmaz. bunun yerine, 'signal' ve 'slot' kavramlarına sahiptir. Qt'nin görsel elemanlarına alet(widget) denir ve bunlar QWidget sınıfından türetilir. Bir olay gerçekleştiğinde, bu aletler bir sinyal alır. Örneğin bir butona basıldığında buton 'clicked' sinyalini alır. Eğer geliştirici bu sinyal alındığında bir iş yapılmasını istiyorsa, bu sinyali bir 'slot'la birleştirir. Sinyaller ve slotlar türe bağımlı olduğu için, MFC gibi hata durumunda çakılmak yerine tür hataları verir.

```
connect(loginButton,SIGNAL(clicked()),qApp,SLOT(LoginUser()));
```

MFC ile Qt arasında belirgin farklar olmasına rağmen, Qt geliştiricileri MFC'den Qt'ye geçişi kolaylaştırmak için teknikler üretmiştir. Bu amaçla da 'Windows Migration Framework' adında bir kütüphane oluşturulmuştur. Bir MFC uygulamasını Qt çatısında kullanırken, Qt aletlerini MFC mesaj haritası yapısında kullanmak mümkündür. MFC uygulamasının kontrollerini Qt yapabilir. Bu uygulamaya yeni kontroller eklemek istendiğinde ise MFC kontrolleri yerine Qt aletleri kullanılabilir.

Son olarak, Qt ile ilgili dikkat edilmesi gereken bir konu da lisanslama konusudur. Linux üzerinden ticari olmayan bir yazılım geliştiriliyorsa, Qt ücret ödemediği için kullanılabilir. Fakat

ticari uygulamalarda lisans ücreti ödenmesi gerekir. Ayrıca Qt Windows ya da gömülü bir sistem üzerinde çalıştırılacaksa lisans ücreti hukuksal bir zorunluluk haline gelir.

b. wxWidgets

wxWidgets, C++ kullanarak çok platformlu arayüz uygulamaları geliştirilebilen bir araçtır. Desteklediği platformlar, Microsoft Windows, Mac OS, Linux/UNIX(X11, Motif, GTK+), OpenVMS ve OS/2dir. Gömülü sistemler için olan bir versiyonu da halen geliştirme evresindedir. Lisanslama konusunda tamamen geliştiriciye özgürlük tanıyan, ücretsiz bir araç olan wxWidgets hakkında ilk söylenmesi gerekenlerden biri, platform-bağımsız bir araç olduğu ve bulunduğu platformun yerel kütüphanelerini kullandığıdır. Bu özellik sayesinde bir wxWidgets yazılımı Windows üzerinde çalıştırılırken diğer Windows uygulamalarıyla aynı görüntü özelliklerine sahip olacak, aynı yazılımı Linux üzerinden çalıştırdığımızda ise Linux'un görüntüsüne sahip olacaktır. Çünkü kendi görüntü kütüphaneleri yerine bulunduğu ortamın görüntü kütüphanelerini kullanmaktadır. Bu bir işletim sisteminde derlediğiniz kodun çalıştırılabilir çıktısını direkt olarak başka bir işletim sisteminde çalıştırabileceğiniz anlamına gelmez, fakat kaynak kodu uyumluluğu konusunda çok önemli bir avantaj sağlar.

wxWidgets, çalışma mantığı olarak MFC'ye çok benzemektedir. Örneğin wxWidgets, MFC ile çok benzer bir mesaj haritası yöntemi kullanmaktadır. Bir MFC uygulamasında mesajların kullanılacağı, BEGIN_MESSAGE_MAP ve END_MESSAGE_MAP makrolarıyla belirtilir. Diğer bütün makrolar (ON_BN_CLICKED, ON_COMMAND, ON_WM_PAINT vs.) bu iki makro arasına yazılır. Sınıfların sonunda ise DECLARE_MESSAGE_MAP makrosu kullanılarak, sınıf içerisinde

de deklare edilen makroların haritasının oluşması sağlanır. wxWidgets da bunlara benzer bir yapıdadır. BEGIN_EVENT_TABLE ve END_EVENT_TABLE makroları, MFC'nin BEGIN_MESSAGE_MAP ve END_MESSAGE_MAP makrolarıyla aynı işi yapar. Ayrıca bu iki makro arasına EVT_MENU, EVT_BUTTON ve EVT_INIT_DIALOG gibi makrolar yerleştirilir.

wxWidgets ile MFC arasındaki benzerlikler core sınıfların karşılaştırılmasında da gözlenir. Tablolar ve şekiller bölümünde Şekil 1 bu benzerliği ortaya koyar. Bu benzerlikler, kontrollerde de karşımıza çıkar. Şekil 2, buton kontrollerindeki benzerliği göstermektedir. Son olarak, iki çatının yardımcı sınıfları da aralarındaki benzerliği ortaya koymaktadır.

Yukarıda anlatılanlar da göstermektedir ki wxWidgets özellikle MFC ile benzerliğinden dolayı, Windows kullanıcılarının taşınabilir kod yazmalarını kolaylaştırmaktadır. Fakat her platformda platformun kendi yerel kütüphanelerini kullandığı için, herhangi bir platformdan diğerine uygulama taşıma konusunda oldukça çekici bir araçtır.

wxWidgets, açık kaynak kodlu ve ücretsiz bir araçtır. Bu özellikleri, aracın sadece düşük bütçelilere yönelik olduğunu göstermez. Aksine, ücretsiz olması kullanıcı sayısını artırır, açık kaynak kodlu olması da her kullanıcının aynı zamanda bir geliştirici olması anlamına gelir.

wxWidgets, benzer birçok çatıdan daha fazla derleyiciyle derlenebilir. Symantec C++ hariç bütün popüler Windows derleyicileri kullanılabilir, ya da Windows üzerinden Cygwin ya da Mingw32 ile de derlenebilir. Unix üzerinden de, bilinen bütün C++ derleyicileri Motif ve GTK+ platformlarını destekler.

c. wxWidgets Qt karşılaştırması

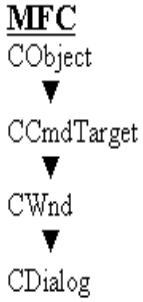
C++ kullanarak ara birim hazırlayan geliştiricilerin en sık kullandığı iki çatı olan Qt ve wxWidgets, her konuda birini diğerine tercih ettirecek avantajlara sahip değildir. Bazı açılardan Qt çok daha avantajlı görünürken, bakış açısı değiştirildiğinde wxWidget öne geçebilmektedir. Bu bölüm, taşınabilirlik açısından bu iki çatıyı değerlendirmektedir.

- Qt ve wxWidgets date/time, ağ, veri tabanı ve OpenGL gibi arayüzle ilgili olmayan birçok sınıfa sahiptir.
- Qt3, Mac ve GNU/Linux üzerinde geliştirilen açık kaynak kodlu yazılımlar için GPL ile lisanslanmıştır. Ticari uygulamalar ve Windows üzerinde geliştirilen uygulamalar için ise QPL ile lisanslanmıştır. Qt4 de Windows için GPL ile lisanslanmıştır. Buna karşın, wxWidgets her durum ve platform altında değiştirilmiş bir LGPL lisansı ile lisanslanmıştır. Bu da wxWidgets geliştiricilerini lisans ücreti konusunda rahatlatır.
- Qt, sinyal/slot mantığına göre çalışır. bu da arayüz elemanının projedeki metot ve fonksiyonlara bağlanmasını oldukça kolay bir hale getirir.
- Qt, wxWidgets gibi gerçek anlamda bir yerel porta sahip değildir. Her ne kadar gerçekçi görüntüler oluştursa da aslında Qt her ortamda kendi aletlerini göstermektedir. Olay işleme, sonucunda görsel geri besleme ve alet yerleşimi tamamen Qt tarafından kontrol edilmektedir. Buna benzer bir sistem, wxUniversal ile gerçekleştirilmektedir.
- Qt gibi kendi aletlerini çizmek yerine, wxWidgets bulunduğu platformun

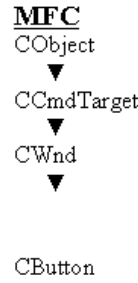
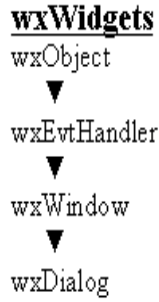
yerel aletleri üzerinde bir soyutlama yapar(örneğin OS X üzerinde Aqua, Linux üzerinde GTK ya da Motif vb.). Bu da wxWidgets uygulamalarının taşınacak uygulamanın performans açısından çok daha verimli olmasını sağlar.

- Qt, dokümantasyon açısından güçlü bir çerçeve olmasına rağmen, wxWidgets bu açıdan zayıf kalmaktadır. Bu da büyük çaplı projelerin kontrolü için ek araçlara ve ek çabaya mal olur.
- Qt büyük yazılım projeleri için daha çok tercih edilir. KDE ve Opera bunlara örnektir. Fakat wxWidgets da bu çapta kullanılmaya başlamıştır. Örneğin AOL Connector projesinde wxWidgets kullanılmıştır.
- Qt ile çok daha fazla sanal fonksiyon kullanımı görülür. Örneğin bitin aletlerin temel sınıfı olan QWidget, 50 den fazla sanal fonksiyona sahiptir. Bu da wxWidgets'a göre Qt'nin nesne yönelimli programlama mantığına daha yakın olmasını sağlar. Dolayısıyla, Qt aynı yazılımı daha az kod satırıyla yapabilir.
- Her ne kadar Qt daha az kod satırına ihtiyaç duysa da, ek özellikleri ile birlikte projenin boyut ve yapım açısından daha kompleks olmasına sebep olur. Bu komplekslik de Qt'nin wxWidgets'a göre daha yavaş çalışmasına sebep olur.
- Qt'de, yoğun bir makro kullanımı vardır. Bu her ne kadar kompleksliği düşürse de, Qt uygulamalarının taşındığı ortama eklenmesini zorlaştırır.
- Qt için tek bir arayüz geliştirme ortamı vardır (Qt Designer). wxWidgets için ise tercihler daha fazladır.

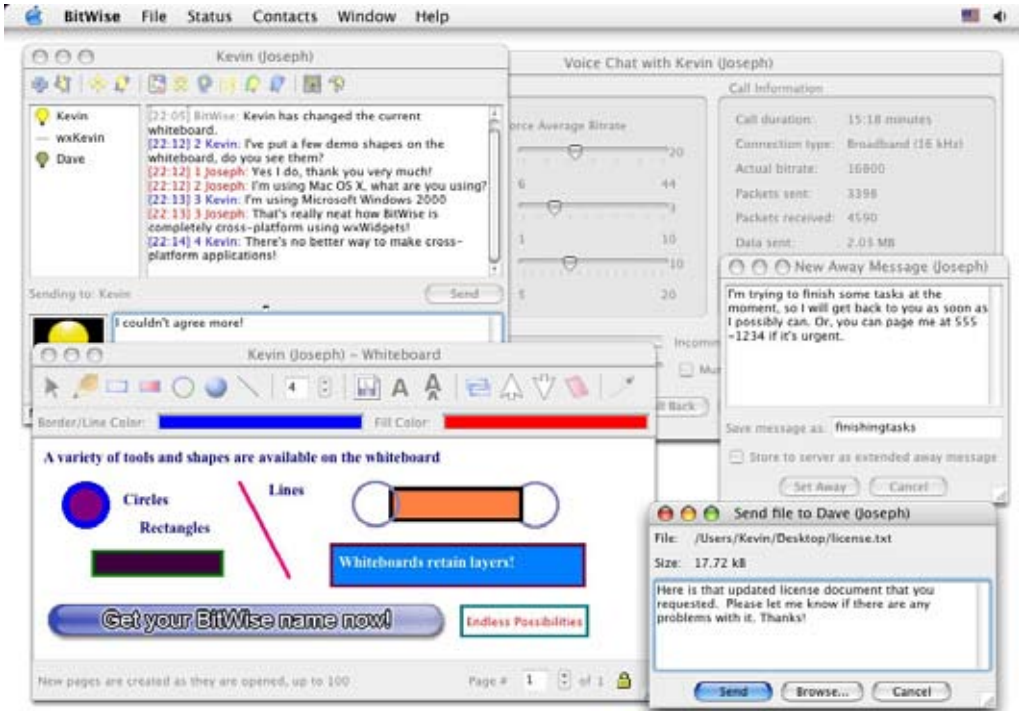
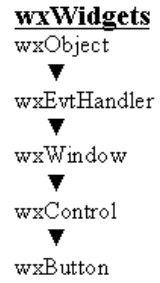
2. Tablolar ve Şekiller



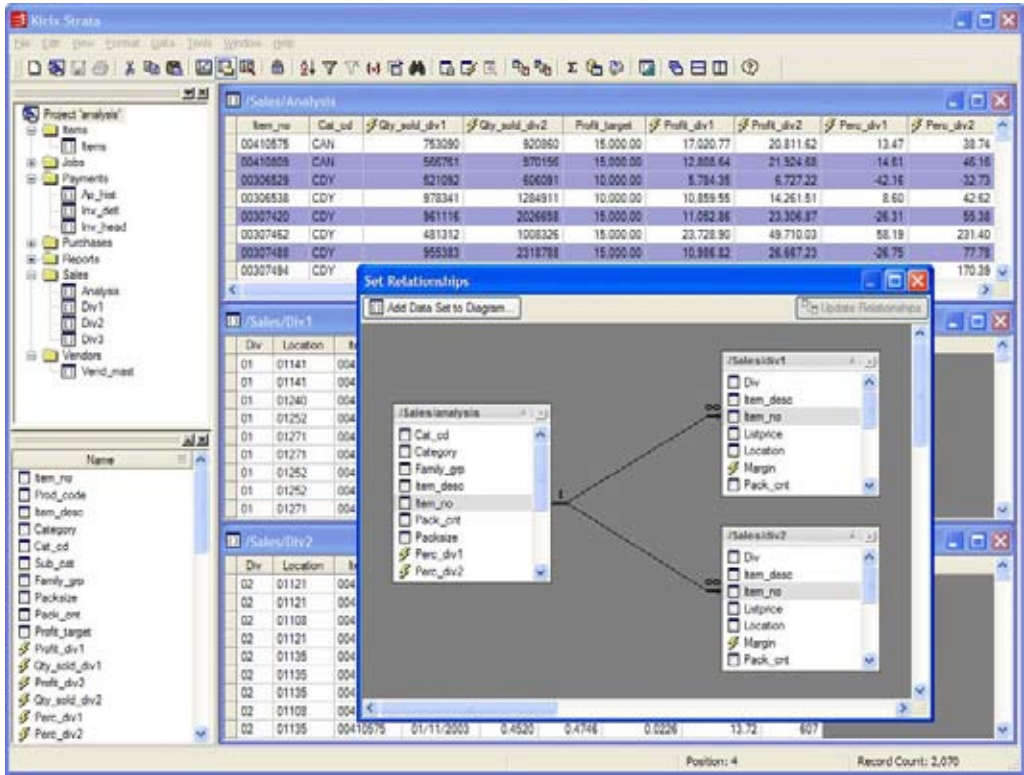
Şekil 1



Şekil 2



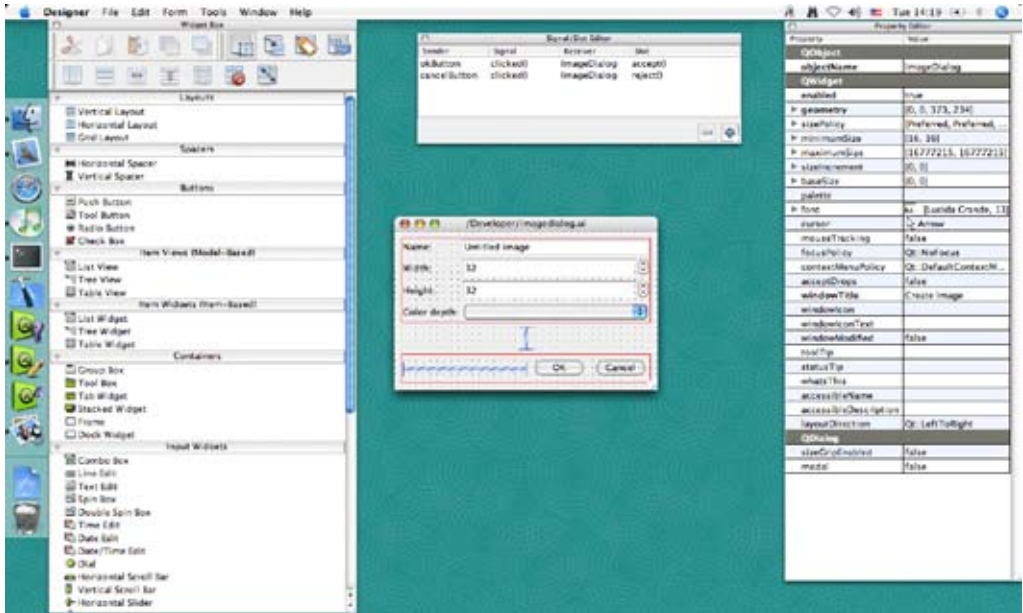
Şekil 3: Mac OS üzerinde wxWidgets ile geliştirilmiş bir mesajlaşma aracı(Bitwise).



Şekil 4: Windows üzerinde wxWidgets ile geliştirilmiş bir dinamik veritabanı programı(Kirix Strata).



Şekil 5: Linux üzerinde wxWidgets ile geliştirilmiş bir dinamik veritabanı programı(Audacity).



Şekil 8: Qt Designer on Mac OS

2. Kaynakça

1. Cool Solutions: Porting Windows MFC applications to Linux

<http://www.novell.com/cool solutions/feature/11244.html>

2. WxWidgets Compared To Other Toolkits

http://www.wxwidgets.org/wiki/index.php/WxWidgets_Compared_To_Other_Toolkits

3. Graphical User Interface Portability -Feb 1997

<http://www.stsc.hill.af.mil/crosstalk/1997/02/graphical.asp>

4. Cross-Platform GUI Programming with wxWidgets, Smart J., Hock K., Prentice Hall, 2006

5. C++ GUI programming with Qt 4, Blanchelette J., Summerfield M., Pearson Hall in association with Trolltech Press, 2006

6. The C programming language : including ANSI C, portability, and software engineering, Troy D. A., Kiper D. J., Scott, Foresman, 1989