İ.T.Ü. TEKNİK ÜNİVERSİTESİ ★FEN BİLİMLERİ ENSTİTÜSÜ

# X WINDOWS TUTORIAL PROGRAMI

YÜKSEK LİSANS TEZİ

Mat. Müh. Nuran ESEN




| | | |
|---|---|---|
| Tezin Enstitüye Verildiği Tarih | : | 27 MAYIS 1996 |
| Tezin Savunulduğu Tarih | : | 12 HAZİRAN 1996 |
| Tez Danışmanı | : | Prof. Dr. Metin DEMİRALP |
| Diğer Jüri Üyeleri | : | Prof. Dr. Galip TEPEHAN |
| | : | Doç. Dr. Gazanfer ÜNAL |

HAZİRAN, 1996

# ÖNSÖZ

Bana, kendisi ile çalışma olanağı veren ve çalışmamın her aşamasında sürekli olarak fikir ve yardımlarından yararlandığım değerli hocam Sayın Prof. Dr. Metin Demiralp'e en içten teşekkürlerimi sunarım.

Mayıs, 1996 Nuran METİN ESEN

İÇİNDEKİLER

# ÖZET

Bu projenin amacı, en basit şekliyle bir X kullanıcısına X'in yapısı ve programlanması hakkında bilgi sunan bir X Windows tutorial programının tasarlanıp gerçekleştirilmesidir. Projenin içerdiği konular şu şekilde özetlenebilir: X Windows için genel kavramların verilmesi, X rutinleri, grafik içeriği, pencerede yazı çizimi, ekranda karakterlerin basımı, olaylar ve renk kullanımı.

## SUMMARY

## X11 WINDOWS TUTORIAL PROGRAM

In September 1987, the Massachuttes Instute of Technology released the first snapshot of what may well become one of the most significant software technologies of the 1990s: Version 11 of the X Window System, commonly referred to as X11. X11 may not change the world, but it is likely to change the world of workstations.

The X Window System is being adopted as a standard by nearly every workstation manufacturer and should eventually replace or be supported under their proprietary windowing systems. Versions will also be available for personal computers and supercomputers.

For the first time, portable applications can be written for an entire class of machines rather than for a single manufacturer's equipment. Programmers can write in a single graphics language and expect their applications to work without significant modifications on dozens of different computers.

What's more, since X is a network-based windowing system, applications can run in a network of systems from different vendors. Programs can be run on a remote computer, and the results displayed on a local workstation. Proprietary networks have been around for a while. However, network cooperation of different computers has been held up by the lack of a common applications language. Now there is one.

Vendors hope that X will lead to a software explosion similar to the one that occured in response to the PC standard on microcomputers.

### X Window System Concepts:

The X Window System is complex, bur it is based on a few premises that can be quickly understood. This section describes these major concepts.

### Displays and Screens:

The first and most obvious thing to note about X is that it is a windowing system for bitmapped graphics displays. It supports color as well as monochrome and gray-scale displays.

A slightly unusual feature is that a display is defined as a workstation consisting of a keyboard, a pointing device such as a mouse, and one or more screens. Multiple screens can work together, with mouse movement allowed to

cross physical screen boundaries. As long as multiple screens are controlled by a single keyboard and pointing device, they comprise only a single display.

**The Server-Client Model:**

The next thing to note is that X is a network-oriented windowing system. An application need not be running on the same system that actually supports the display. While many applications can execute locally on a workstation, other applications can execute on other machines, sending requests across the network to a particular display and receiving keyboard and pointer events from the system controlling the display.

At this point, only TCP/IP and DECnet networks are supported by the X consortium and most vendors, though that may change before long.

The program that controls each display is known as a server. At first, this usage of the term server may seem a little odd – when you sit a workstation, you tend to think of a server as something across the network (such as a file or print server) rather than the local program that controls your own display. The thing to remember is that your display is accessible to other systems across the network, and for those systems, the code executing in your system does act as a true display server.

The server acts as an intermediary between user programs (called clients or applications) running on either the local or remote systems and the resources of the local system. The server (without extensions) performs the following tasks:

- Allows access to the display by multiple clients.
- Interprets network messages from clients.
- Passes user input to the clients by sending network messages.
- Does two-dimensional drawing–graphics are performed by the display server rather than by the client.
- Maintains complex data structures, including windows, cursors, fonts, and "graphics context" as resources that can be shared between clients and referred to simply by resource IDs. Server-maintained resources reduce the amount of data that has to be maintained by each client and the amount of data that has to be transferred over the network.

Since the X Window System makes the network transparent to clients, these programs may connect to any display in the network if the host thay are running on has permission from the server that controls that display. In a network environment, it is common for a user to have programs running on several different hosts in the netwrok, all invoked from and displaying their windows on a single screen.

In practice, each user is sitting at a server and can start applications locally to display on the local server or can start applications on remote hosts for display on the local server, if the remote hosts have permission to connect to

the local server. All other users in the network are in a similar situation – they can run applications on their own system or on yours, but they will, for the most part, be displaying on their own server. This use of the network is known as distributed processing. Distributed processing helps solve the problem of unbalanced system loads. When one host machine is overloaded, the users of that machine can arrange for some of their programs to run on other hosts.

One extreme of this arrangement is the PC server or X terminal. Because these single-task systems can run only the X server (and sometimes a window manager), a user sitting at one of these servers must run all clients on systems across the network, with their results displayed on the PC or X terminal screen. This makes the single-tasking PC or X terminal look and work just like X on a multitasking workstation.

**Window Management:**

Another important concept in X programming is that applications do not actually control such things as where a window appears or what size it is. Given multiprocessor, multiclient access to the same workstation display, clients must not be dependent on a particular window configuration. Instead, a client gives hints about how long and where it would like to be displayed. The screen layout or appearance and the style of user interaction with the system are left up to a separate program, called the window manager.

The window manager is just another program written with Xlib, except that it is given special authority to control the layout of windows on the screen. The window manager typically allows the user to move or resize windows, start new applicationss, and control the stacking of windows on the screen, but only according to the window manager's window layout policy. A window layout policy is a set of rules that specify allowable sizes and positions of windows and icons.

Unlike citizens, the window manager has rights but not responsibilities. Programs must be prepared to cooperate with any type of window manager or with none at all (there are fairly simple ways to prepare programs for these contingencies). The simple window manager twm does not enforce any window layout policy, but clients should still assume that there could be one. For example, the window manager must be informed of the desired size of a new window before the window is displayed on the screen. If the window manager does not accept the desired window size and position, the program must be prepared to accept a diffrent size or position or be able to display a message such as "Too small!"

X is somewhat unusual in that it does not mandate a particular type of window manager. Its developers have tried to make X itself as free of window management or user interface policy as possible. And, while the X11 distribution includes twm as a sample window manager, individual manufactures are expected to write their own window managers and user interface guidelines. In fact, two commercial window managers with user interface guidelines are

already becoming established. They are olwm, the OPEN LOOK window manager from AT&T and Sun, and mwm, the Motif window manager from Open Software Foundation. The OSF Motif window manager, mwm, and OPEN LOOK window manager olwm both can be configured to be real-estate-driven or click-to-type.

**Events:**

As in any mouse-driven window system, an X client must be prepared to respond to any of many different events. Events include user input (keypress, mouse click, or mouse movement) as well as interaction with other programs. (For example, if an obscured portion of a window is exposed when another overlapping window is moved, closed, or resized, the client must redraw it.) Events of many different types can occur at any time and in any order. They are placed on a queue in the order they occur and usually are processed by clients in that order. Event-driven programming makes it natural to let the user tell the program what to do instead of vice versa.

The need to handle events is a mojor difference between programming under a window system and traditional UNIX or PC programming. X programs do not use the standard C functions for getting charaters, and they do not poll for input. Instead there are functions for receiving events, and then the program must branch according to the type of event and perform the appropriate response. But unlike traditional programs, an X program must be ready for any kind of event at any time. In traditional programs the program is in control, asking for certain types of input at certain times. In X programs, the user is in control most of the time.

**Overview of X Windows:**

The heart of the X Windows system consists of a program called X which runs on a machine with a display, keyboard, and a mouse. It waits for other programs to tell it what to do or for something to happen to the pointer or keyboard. The programs can be running on the same machine as X is or elsewhere on the network, maybe on a machine that hasn't even got a display of its own. This 'network transparency' is one of the strengths of X. Graphics programs only have to know about X, not about the special low level graphics commands for each type of machine. The client programs communicate to X, the server, via a Protocol language that is common across machine types.

All a client program needs to do use the X display is to open up a connection with the server and then send Protocol requests to it. To simplify sending these, an extensive library of about 200 display subroutines is provided and it is this library, Xlib, which this thesis mainly decribes.

Many client programs can simultaneously use the same X server. To save each client having its own copy of fonts, color information, etc, (thereby wasting space and causing more data to be passed via the network), the server stores data on behalf of the clients, allowing sharing wherever possible. In order

to enable the client to reference these resources the server provides resource codes and these can be used in many of the routines to specify that certain data is to be used.

X is 'event driven'. For each window you create you can select what sort of events (key presses, re-exposure, etc) you want it to respond to. Typically, an X program consists of a set-up sequence followed by an 'event-loop' which waits for events to be reported by the server, determines what sort of event has happened and in which window, then processes the event.

There are 4 types of messages passing between the client and server;

- Requests – the client can ask the server to draw something, or ask for information.
- Replies – the server can reply
- Events – the server can surprize the client with something
- Errors – the server can report an error

Both display requests and events are buffered and the server executes asynchronously much of the time to meximise efficent use of the network. When the client wishes to establish syncronization it often has to ask for it, though working in syncronized mode can be up to 30 times slower.

X clients programs should be written mindful of the fact that other clients are likely to be running at the same time. Windows should be prepared to be covered over by others then exposed, and they have to redraw themselves. Neither should clients indescriminately use scarce resources like off-screen memory or monopolise the keyboard or pointer. If they can accept cut-and-paste operations, so much the better.

One client that is always likely to be running is a 'window manager'; a program which allows you to resize, move and re-stack windows, pop menus up, etc. Any X application you write will need to work under a window manager. Some window managers are more bossy than others (some won't let you raise windows, for instance), but there are X commands so that a client can at least make suggestions to the window manager.

X can support one or more screens containing overlapping (sub)windows and works on many types of hardware but it doesn't provide high level support. If you want to prefabricate dialog boxes form pre-defined scrollbars, buttons, etc then you should use a 'toolkit' that sits on top of X.

This thesis has 7 chapters. The information about the chapters are following:

Chapter 1 Introduction: describes about X system concepts, X windows versions, displays and screens, client-server models, windows management, events, extension to X, X windows system software architecture.

Chapter 2 X Routins: describes about opening a display server, finding out about the display, creating & destroying windows, mapping, moving & uncovering windows and defining bitmaps and pixmaps.

Chapter 3 Graphics Context: describes graphics context and how to use it.

Chapter 4 Rectangles, Lines, Dots, Arcs and Text: decribes about drawing rectangles, lines, dots, arcs and text in a window, finding the witdh of characters, printing characters on the screen, creating cursors.

Chapter 5 Events: describes about events, event types and keyboard encoding.

Chapter 6 Using Color: describes about colors, allocating color and read-only color cells.

Chapter 7 X11 Tutorial Program: describes about x11tutor program including source code information and how it is used.

# BÖLÜM 1. GİRİŞ

Massachusetts Institute of Technology (MIT) Eylül 1987'de, 1990'ların en iyi yazılım teknolojilerinden biri olacak X Pencere Sisteminin 11 numaralı versiyonunu X11 olarak piyasaya sürmüştür. X11 belki dünyayı değiştirmemiştir ama iş istasyonları(workstations) dünyasını değiştirdiği bir gerçektir.

X Pencere Sistemi, hemen hemen bütün iş istasyonu üreticileri tarafından bir standart olarak kabul edilmekte olup, üreticilerin özel pencere sistemleri altında desteklenebilir veya bunların yerini alabilir gözükmektedir.

İlk olarak, X Pencere Sisteminde bir tek üreticinin donatımından çok tüm makine sınıfları için portatif uygulamalar yazılabilmesine olanak sağlanır. Programcılar tek bir grafik dilinde programlarını yazabilirler ve uygulamalarını düzinelerce farklı bilgisayar üzerinde özel değişiklikler olmaksızın çalıştırabilirler.

Dahası, X; ağ-tabanlı bir pencereleme sistemi(network-based windowing system) olduğundan uygulamalar farklı marka satıcılarının ağ sistemlerinde koşabilir. Programlar uzak bir bilgisayar üzerinde koşturulabilir ve sonuçlar yerel bir iş istasyonunda görüntülenebilir. Bununla beraber farklı bilgisayarların ağ işbirliği(TCP/IP, NFS, RPC, Telnet gibi uygulamalar), genel uygulamalar dilinin eksikliği yüzünden geciktirilmişse de şu anda bu işbirliğini sağlayan bir X pencereleme sistemi ürünü vardır.

Marka satıcıları, X Pencere Sisteminin, yazılım geliştirilmesinde önderlik edeceğini beklemektedirler.

## 1.1 X Windows Sürümleri

X, ortaklaşa olarak MIT's Project Athena ve Digital Equipment Corporation tarafından diğer birkaç şirketin de katkılarıyla gerçekleştirilmiş ve Robert Scheifler ile MIT'deki meslektaşları tarafından yönetilmiştir.

X'in pekçok araştırma sürümü olmuştur. 1986'da sürümü yapılan ve X10.4 diye bilinen sürüm bazı ticari ürünler için de temel oluşturmuştur. Çoğu X10.4 ürünlerinin gelişimi kısıtlandırılmış, dahası bu ürünlerin sürüm 11 ile uyumlu olmadığı açıkça görülmüştür. 11.1 sürümü Eylül 1987'de, 11.2 sürümü Mart 1988'de, 11.3 sürümü Şubat 1989'da, 11.4 sürümü Ocak 1990'da ve 11.5 sürümü Ağustos 1991'de kullanılabilir duruma getirilmiştir.

Tam bir pencere programlama paketi olan X11 sürümü, desteklenmiş görüntü özellikleri alanlarında, pencere yönetim biçimlerinde ve çoklu ekranlar için destekte daha fazla esneklik teklif ederken aynı zamanda X10 sürümünden daha fazla performans da sağlamaktadır. X11 sürümünün X10 sürümüne göre üstünlükleri genişletilebilir fakat en önemlisi, X11 alt rutin kütüphanesi(subroutine library - Xlib)'nin birkaç yıl için değişmez kalacağı ve en azından bir endüstri standardı olacağının umut edilmesidir. Bunun anlamı, bu kütüphane ile yazılmış programların yazılım güncellemeleri yüzünden büyük değişmelere ihtiyaç duymayacağıdır. Bu kütüphaneye eklemeler olurken uyumsuz değişiklikler olmayacağı taahhüt edilmiştir.

X11.2 sürümü ile birlikte X'in kontrolü MIT'den -X standardını desteklemeyi planlayan, büyük bilgisayar üreticilerinin bir ortaklığı olan ve Ocak 1988'de oluşturulmuş olan- X konsorsiyumuna geçmiştir.

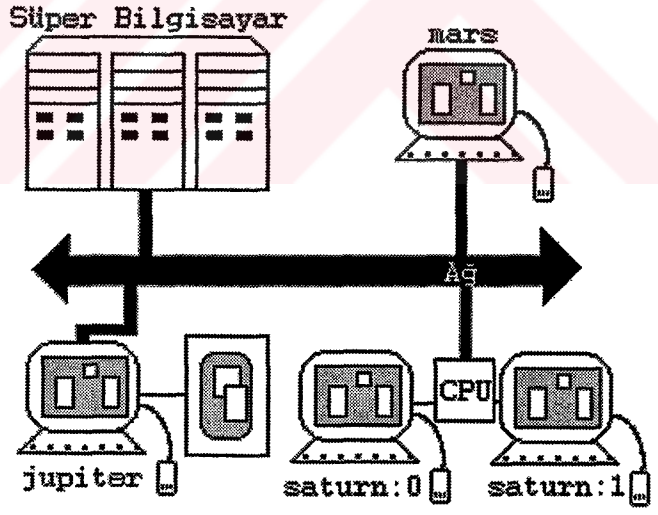
## 1.2 Görüntüler ve Ekranlar

X hakkında not edilebilecek ilk ve en açık şey, onun bitmapped grafik görüntüleri için bir pencere sistemi olduğudur. Bitmapped grafik'de pixel veya resim elemanı denilen ekran üzerindeki her bir nokta, hafızada bir veya daha fazla bite karşılık gelir. Grafik programları görüntüyü, görüntü hafızasına erişerek değiştirirler. Çoğu bitmapped görüntüler televizyon tipi tarama satırı

teknolojisini kullandıklarından dolayı bitmapped grafikler raster grafik olarak da adlandırılır. Bütün ekran sürekli olarak herhangi bir anda görüntünün yüzünün karşı tarafında bulunan bir elektron demet tarayıcısı tarafından bir tarama satırı veya raster olarak yenilenir. Bitmapped grafik terimi memory-mapped grafik'e göre daha geneldir, bu yüzden LCD ekranları gibi diğer nokta tabanlı(dot-oriented) görüntülere de uygulanır.

X, monokrom ve gri-ölçekli(monochrome and gray-scale) görüntüler kadar iyi olmak üzere rengi destekler.

Biraz olağandışı olan özelliği ise bir görüntünün, bir klavye, fare gibi bir işaretleme aygıtı ve bir veya daha fazla ekran içeren bir iş istasyonu olarak tanımlanmasıdır. Birçok ekran, fiziksel ekran sınırlarının aşılmasına izin veren fare hareketi yardımıyla birarada çalışabilir. Bu yüzden birden fazla sayıda ekran, tek bir klavye ve işaretleme aygıtı ile bir kullanıcı tarafından kontrol edildiği sürece Şekil 1.1'de görüldüğü gibi sadece bir tek görüntüden oluşur.




Şekil 1.1 X uygulamaları ağ üzerinde herhangi bir sistemde koşturulabilir

## 1.3 Sunucu-İstemci Modeli

X hakkında not edilebilecek bir diğer şey, onun ağ tabanlı bir pencereleme sistemi olduğudur. Bir uygulama, gerçekte görüntüyü destekleyen sistem üzerinde koşmak zorunda değildir. Birçok uygulama bir iş istasyonu üzerinde yerel

olarak çalışabiliyorken diğer bazı uygulamalar, ihtiyaçlarını ağ üzerinden özel bir görüntüye göndermek ve görüntüyü kontrol eden sistemden gelen klavye ve işaretçi olaylarını almak suretiyle başka makinelerde çalışabilmektedir.

Bu noktada; yakında değişebilme olasılığına rağmen, sadece TCP/IP ve DECnet ağlarının X konsorsiyumu ve birçok marka satıcısı tarafından desteklendiği belirtilmelidir.

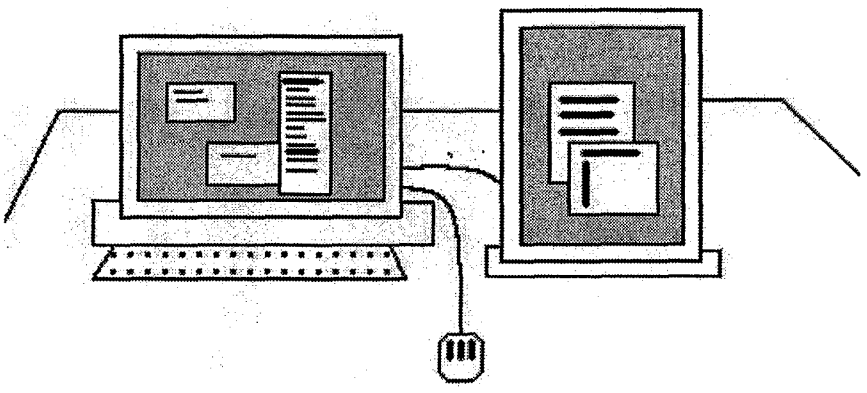
Her bir görüntüyü kontrol eden program, sunucu(server) olarak bilinmektedir. İlk seferinde sunucu teriminin kullanımı biraz garip gelebilir - Bir iş istasyonunun başına oturduğunuz zaman, sunucunun, görüntünüzü kontrol eden yerel bir programdan daha çok, (bir dosya veya bir yazıcı sunucusu gibi) ağ içindeki birşey olduğunu düşünürsünüz. Bu noktada hatırlanması gereken şey, görüntünüzün ağ üzerindeki diğer sistemlere ulaşabilir olması ve sizin sisteminizde diğer sistemler için çalışmakta olan kodun, gerçek bir görüntü sunucusu gibi hareket etmesidir.

Sunucu, yerel veya uzak sistemler üzerinde çalışan ve istemciler(clients) ya da uygulamalar diye bilinen kullanıcı programları ile yerel sistem kaynakları arasında bir aracı gibi davranır. Genişlemeleri olmaksızın sunucu aşağıdaki görevleri yerine getirir:

- Birçok istemci tarafından görüntüye erişime izin verir.
- İstemcilerden gelen ağ mesajlarını yorumlar.
- Ağ mesajları göndererek istemcilere kullanıcı girişi iletir.
- İki boyutlu çizim yapar. Grafikler, istemciden daha çok görüntü sunucusu tarafından gerçekleştirilir.
- İstemciler arasında paylaşılabilir kaynaklar olarak pencereleri, fontları ve 'grafik içeriği' içeren karmaşık veri yapılarını korur. Sunucu-korunmuş kaynaklar, her bir istemci tarafından korunulması zorunlu veri miktarı ile ağ üzerinde transferi zorunlu veri miktarını azaltır.

Bu yüzden X Pencere Sistemi istemcilere ağ şeffaflığı(network transparent) sağlar, eğer programların üzerinde çalıştığı makinenin görüntüyü kontrol eden sunucu tarafından verilmiş izni varsa bu programlar, ağ içinde bulunan

ve sunucunun kontrol ettiği herhangi bir görüntüye bağlanabilir. Bir ağ ortamında, Şekil 1.2'de görüldüğü gibi tek bir ekranda program pencerelerini görüntüleyerek ve kullanıcı girişlerini alarak bir kullanıcının ağ içindeki birkaç farklı makine üzerinde koşan programlara sahip olması olağandır.



Şekil 1.2 Birden fazla ekran içeren bir görüntü sistemi

Pratikte, her bir kullanıcı bir sunucu önünde oturur ve yerel sunucu üzerinde göstermek üzere yerel olarak uygulamaları başlatabilir veya yerel sunucu üzerindeki görüntü için uzak makinelerdeki uygulamaları eğer uzak makinelerin yerel sunucuya bağlanma izinleri varsa başlatabilir. Ağ üzerindeki diğer bütün kullanıcılar benzer durumdadır - kullanıcılar kendi sistemlerindeki veya sizin sisteminizdeki uygulamaları koşturabilirler, fakat birçok bölüm için bu uygulamalar kendi sunucularında görüntüleniyor olacaktır. Ağın bu kullanımı dağıtılmış işlem(distributed processing) olarak bilinir. Dağıtılmış işlem, dengesiz sistem yüklemesi probleminin çözümüne yardım eder. Bir makine fazla yükleme yaptığı zaman bu makinenin kullanıcıları kendi programlarının bir kısmını diğer makineler üzerinde koşturmak için düzenleyebilirler.

Bu düzenlemenin bir sakıncası PC sunucu veya X terminaldir. Çünkü bu tek-görevli sistemler(single-task systems) sadece X sunucuda ve bazen de bir pencere yöneticisinde koşturulabilir; bu sunuculardan birinde oturan kullanıcının, sonuçlarını X terminal ekranında veya PC üzerinde alabilmesi için ağ içindeki ilgili bütün istemcileri koşturması zorunludur. Tek görevli PC veya X terminal üzerinde çalışan görüntüleme sistemi, çok kullanıcılı iş istasyonu üzerinde X'in çalışması gibi çalışır. Örnek olarak (PC'ler için);

- PC-NFS, PC-TCP
- PC-XWindow, PC-XWindows on Windows3.1
- PC-NFS-Daemon(sunucu üzerinde çalışacak)

programları, X terminal için de bunlara benzer özel bir takım programlar, (yukarıda anlatıldığı gibi) sonuçların PC veya X terminal üzerinden alınabilmesi için çalıştırılmak zorundadır. İlk iki grup PC üzerinde, üçüncü grupsa sunucu üzerinde çalışan programları içerir. İlk iki grup için, herbir gruptan bir programın çalışması gerekmektedir.

NOT - Server için sunucu yerine sunumcu ifadesi de kullanılmaktadır.

## 1.4 Pencere Yönetimi

X programlamadaki bir diğer önemli kavram; uygulamaların, bir pencerenin göründüğü yer veya o pencerenin boyutunun ne olduğu gibi şeyleri gerçekten kontrol etmemesidir. Verilmiş çokluişlemci(multiprocessor), çok istemci erişimli aynı iş istasyonu görüntüsünde, istemciler özel bir pencere konfigürasyonuna sahip olmamalıdırlar. Bunun yerine bir istemci, kendisinin nerede ve ne kadar süreyle görüntülenmek istendiği hakkında bilgiler verir. Ekran planı veya görüntüsü ve kullanıcının sistem ile etkileşim biçimi pencere yöneticisi denen ayrı bir programa bırakılır.

Pencere yöneticisi, ekrandaki pencerelerin planını kontrol etmek için kendisine verilmiş olan özel yetkinin dışında sadece, Xlib ile yazılmış başka bir programdır. Pencere yöneticisi; tipik olarak kullanıcıya, pencereleri yeniden boyutlandırması veya hareket ettirmesi, yeni uygulamaları başlatması ve ekran üzerindeki yığılmış pencereleri kontrol etmesi için izin verir; fakat sadece pencere yöneticisinin pencere planı politikasına göre!. Bir pencere planı politikası, simgelerin(icons) ve pencerelerin izin verilebilir pozisyonları ile boyutlarını belirten bir kurallar kümesidir.

Pencere yöneticisi haklara sahip fakat sorumluluklara sahip değildir. Programlar, herhangi bir pencere yöneticisi ile işbirliğine hazır olmalı iken hiçbir pencere yöneticisi yokken de çalışabilmelidir (Bu olasılıklarda programları hazırlamak için kurallara uygun basit yöntemler vardır). Basit bir pencere

yöneticisi olan twm herhangi bir pencere plan politikasını uygulamaz, fakat istemciler hala böyle bir politika olduğunu kabul ederler. Örneğin, pencere yöneticisine, pencerenin ekran üzerinde görüntülenmesinden önce yeni bir pencerenin istenen ölçüsü haber verilmelidir. Eğer pencere yöneticisi istenen pencere ölçüsünü ve pozisyonunu kabul etmezse, program farklı bir ölçüyü veya pozisyonu kabul etmek için hazırlanabilmeli ya da 'çok küçük' gibi bir mesajı görüntüleyebilmelidir.

Eğer bu durumu hayalinizde canlandırmak sıkıntı veriyorsa hiçbir pencerenin çakışmasına(üst üste gelmesine) izin verilmeyen yerde bir pencere yöneticisi düşünün. Bu, örülü pencere yöneticisi(tiled window manager) diye bilinir. SIEMENS RTL örülü pencere yöneticisi, sadece geçici pencereleri - örneğin çıkıveren menüleri(pop-up menus) - üst üste çakıştırır. twm pencere yöneticisi başka bir deyişle mülk-sürücülü(real-estate-driven) olarak bilinir, çünkü klavye girişi, işaretleyicinin halihazırda içinde bulunduğu pencereye otomatik olarak atanır.

Dinleyici (Listener) veya yazmak-için-tıkla (click-to-type) diye bilinen en az bir diğer pencere yöneticisi çeşidi ile daha karşılaşılabilir. Bu çeşit pencere yöneticisinin ayırt edici özelliği, üzerinde işaretçinin tıklanması ile seçilen tek bir pencereye bütün klavye girişlerini atamasıdır. Bir dinleyici, pencerelerin üst üste çakışmasına izin verebilir veya vermeyebilir. Apple Macintosh kullanıcıları bu çeşit arayüzü tanıyacaklardır.

X içindeki alışılmamış birşey, X'in özel bir pencere yöneticisi tipine bağlı olmamasıdır. X'i geliştirenler X'in kendisini mümkün olduğunca pencere yönetiminden veya kullanıcı arayüz politikasından bağımsız yapmaya çalışmışlardır ve X11 sürümü, örnek bir pencere yöneticisi olarak twm'i içeriyorken bireysel üreticiler kendi pencere yöneticilerini ve kullanıcı arayüz rehberlerini yazmayı beklerler. Gerçekte, kullanıcı arayüz rehberli iki ticari pencere yöneticisi halihazırda yayınlanmış durumdadır. Bunlar, AT&T ve Sun firmaları tarafından üretilen Open Look Pencere Yöneticisi(OLWM - Open Look Window Manager) ile Open Software Foundation firması tarafından üretilen Motif Pencere

Yöneticisidir(Motif Window Manager - MWM). OSF Motif pencere yöneticisi ve Open Look pencere yöneticisinin ikisi de mülk-sürücülü veya yazmak-için-tıkla olarak düzenlenebilir.

## 1.5 Olaylar

Herhangi bir fare-sürücülü (mouse-driven) pencere sisteminde olduğu gibi, bir X istemci, birçok farklı olaydan birine cevap vermek için hazırlanmalıdır. Olaylar(events), diğer programlarla etkileşim kadar iyi olmak üzere kullanıcı girişini (klavyeye basılması, fare tıklanması veya fare hareketi) içerirler. Örneğin, eğer gizlenmiş (görülmesi güç) bir pencere kısmı, kendisiyle üst üste çakışan diğer pencere hareket ettirildiği, kapatıldığı veya tekrar boyutlandırıldığı zaman (ki bu durumda istemci pencereyi tekrar çizmelidir) açığa çıkar. Birçok farklı tipte olay herhangi bir zamanda veya herhangi bir sırada meydana gelebilir. Olaylar, meydana geldikleri sırada bir kuyruğa yerleştirilirler ve genellikle o sıradaki istemciler tarafından işlenirler. Olay-sürücülü (event-driven) programlama, kullanıcının programa hangi olayda ne yapması gerektiğini söylemesini doğal kılarken, bunun tam tersi de doğrudur.

Olayları idare etme ihtiyacı, bir pencere sistemi altında programlama ile geleneksel UNIX veya PC programlama arasındaki büyük bir farklılıktır. X programları, karakterleri elde etmek için standart C fonksiyonlarını kullanmazlar ve giriş için kayıt etmezler. Bunun yerine, olayları almak için fonksiyonlar vardır; o zaman program, olayın tipine göre kollara ayrılmalı ve uygun cevabı yerine getirmelidir. Fakat geleneksel programların aksine bir X programı, herhangi bir anda herhangi bir çeşit olay için hazırlanmalıdır. Geleneksel programlarda, kesin bir zamanda tipi kesin olarak bilinen girişlere karşılık vermesi için programlar kontrol altında tutulur. X programlarında ise çoğu zaman kullanıcı kontrol altındadır.

## 1.6 X'e Genişlemeler

X hakkında bilinmesi gereken son şey, X'in genişletilebilir olmasıdır. Kod, genişlemeleri birleştirmek için tanımlanmış bir mekanizma içerir, bu yüzden

özellikler eklendiği zaman uyumsuz şekillerde oluşan sistemleri küçültmesi için marka satıcılarına baskı yapılamaz. Bu genişlemeler, Xlib rutinlerinin içinde gibi kullanılır ve aynı seviyede çalıştırılır. Klavye ve fareden başka giriş aygıtlarını destekleyen X giriş genişlemesi (X input extension) ve dikdörtgensel olmayan pencereleri destekleyen şekil genişlemesi (shape extension) gibi bazı genişlemeler MIT X konsorsiyumunun standartlarıdır. Dahası PHIGS ve PEXlib olarak adlandırılan iki uygulamasıyla birlikte PEX (The PHIGS Extension to X) denilen standart bir üç boyutlu genişleme vardır.

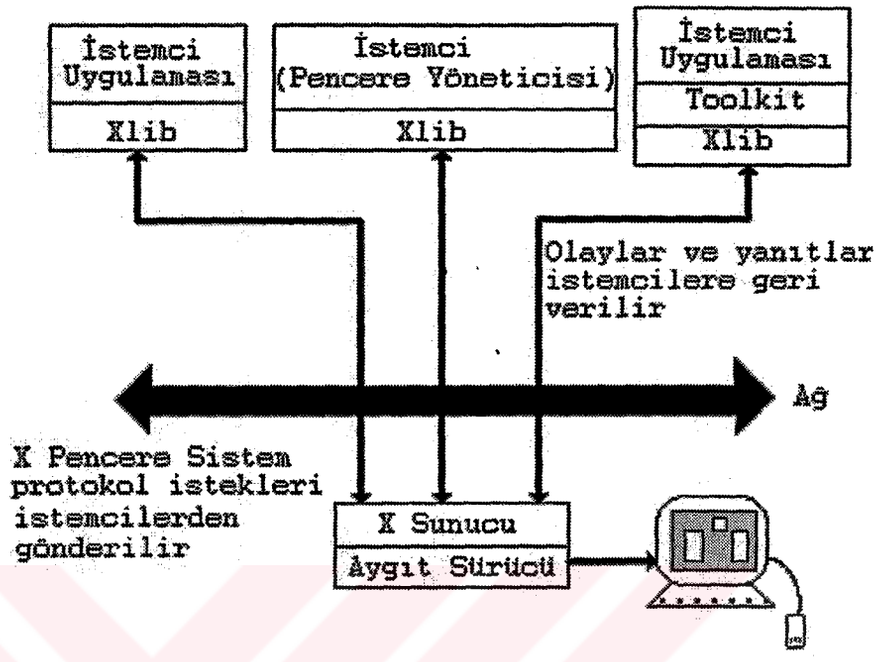
Genişlemeler hem istemci-yönü(client-slide) hem de sunucu-yönü(server-slide) kod içerirler. Bir sunucu satıcısına, bütün standart genişlemelere destek sağlaması için gereksinim duyulmaz. Bu yüzden bir genişlemeyi kullanmadan önce eğer genişleme destekleniyorsa sunucu sorgulanmalıdır.

## 1.7 X Pencere Sistemi Yazılım Mimarisi

Bir görüntü sunucusu, grafik görüntüyü, klavyeyi ve fareyi destekleyen her sistemde çalışan bir programdır (Bkz. Şekil 1.3). MIT'nin X sürümü; SUN, DEC, Hewlett Packard, IBM, Apple Macintosh ve diğer birçok sistemler için örnek monokrom ve renkli sunucular içerir. Ticari olarak geliştirilmiş sunucular hemen hemen bütün büyük iş istasyonu markaları tarafından kullanılabilir. Buna ek olarak Graphics Software Systems, Interactive Systems ve Locus Computing gibi şirketler, IBM uyumlu PC'ler için sunucu gerçeklemelerini teklif ederler. Sonuç olarak ROM(Read Only Memory) içinde çalışan bir X sunucu tarafından kontrol edilen ekranlar olan X terminalleri vardır. X terminalleri Visual, Network Computing Devices ve GraphOn gibi şirketlerden elde edilebilir.

Uygulamalar, Xlib olarak bilinen C dili rutinlerinin düşük seviyeli bir kütüphanesine çağrı yapmak suretiyle sunucu ile haberleşirler. Xlib, özel bir görüntü sunucusuna bağlanmak, pencere yaratmak, grafik çizmek, olaylara cevap vermek ve benzeri şeyler için fonksiyonları temin eder. Xlib çağrıları, ağ üzerinde ya yerel sunucuya ya da başka bir sunucuya TCP/IP yoluyla gönderilmiş protokol isteklerine çevrilir. X sürümü üzerindeki kullanılabilir birçok

örnek uygulamadan birkaçı xterm(bir terminal emülatörü), xcalc(bir hesaplayıcı), xmh(bir posta idarecisi), xclock(bir saat) ve bir troff öngösterimcisidir.




Şekil 1.3 İstemcilerin, Xlib çağrıları kullanarak sunumcu ile haberleşmesi

Pencere yöneticisi sadece, ekran üzerindeki pencerelerin planını kontrol etmek için kendisine verilmiş olan özel otoritenin dışında, X kütüphanesi ile yazılmış başka bir programdır.

İstemci, uygulama ile hemen hemen eş anlamlı olsa da uygulamadan biraz daha genel bir terimdir. Pencere yöneticisi hariç bütün istemciler uygulamalar olarak adlandırılırlar.

Uygulamalar ve pencere yöneticileri yalnızca Xlib ile veya Toolkit olarak bilinen daha yüksek seviyeli alt rutin kütüphanesinin bir takımı ile yazılabilir. Toolkitler, menüler veya komut tuşları(genellikle toolkit widgetleri olarak bilinirler) gibi kullanıcı arayüz özelliklerinin bir kısmını gerçekler ve nesneye dayalı programlama tekniği kullanarak uygulamaların bu özellikleri kullanmasına izin verirler. Toolkitin içeriği programcıların yeni widgetler yaratmasına izin verir.

X11 sürümlü dağıtılmış birçok toolkit vardır. Bunlar içinde en çok dikkat

çekenler digital ve MIT tarafından geliştirilen Xt Toolkit ve Stanford Üniversitesi tarafından geliştirilen Interviews Toolkittir. Xt, şu an X11 standardının resmi parçasıdır.

Toolkitler, çok çok kolay program yapabilmeyi ve projelerin daha eksiksiz bitirilmesini sağlarlar. Ayrıca bunlar, gömülü kullanıcı düzenlemesi ve kullanıcıyı pekçok sorundan kurtaran, pencere yöneticisi ile etkileşimli gömülü kod içerirler. X programlarının çoğunda bir toolkitin kullanımı tavsiye edilir. Bununla birlikte, C'de varolan bütün toolkitler Xlib kodunun kullanılmasına ihtiyaç gösterir veya izin verir. Bunlardan başka, toolkitler Xlib'i içeriden kullanırlar, böylece Xlib'i anlamak toolkitlerin nasıl çalıştığını anlamaya yardım eder.

Bir toolkit kullanmanın diğer sebebi kullanıcı arayüzü düzenini uygulama avantajının elde edilmesidir. OSF'nin Motif'i ve Sun'ın OPEN LOOK'u gibi pekçok kullanıcı arayüzü düzeni vardır. Eğer bütün X11 programlarında Xlib kullanılırsa ya OSF Motif gibi yerleşmiş düzenlerden bir tanesi yeniden gerçeklenme zorunluluğunda kalacak veya programlar kullanıcı beklentilerine uygun olmayan görünüş ya da tepkiler veren garip bir durumda olacaklardır.

Aynı zamanda toolkit kullanmak bir alışveriştir. Bunlardan biri toolkit kullanılarak yazılan bir programın büyüklüğünün Xlib kullanılarak yazılan aynı programınkinden daha büyük olacağıdır. Diğeri ise toolkitlerin yüksek somut kavramlardan yararlanması ve nesneye dayalı dizaynlarından dolayı titiz programlama düzeneğini gerektirmesidir. Bunları öğrenmek ise zaman alır.

### 1.8 Xlib'e Bakış

Önemli fonksiyonlarına göre Xlib rutinlerini aşağıdaki gibi gruplandırılmıştır.

**Renk:** Ekranda yorumlanan ve bir uygulamanın çizimlerinin renklerini değiştiren rutinlerdir.

**İmleçler:** Ekrandaki işaretçi izleri olan imajların şekillerini ve renklerini değiştiren rutinlerdir.

**Veri Yönetimi:** Pencereler veya sayılarla veri ilişkisi kurmak için kullanılan mekanizmalardır.

**Görüntü Bağlantısı:** Muhtemelen ağ üzerinde bir uygulamanın bir görüntüye bağlanmasını veya bağlantıyı koparmasını sağlayan rutinlerdir.

**Görüntü ve Sunucu Özellikleri:** Bir özel sunucu gerçeklemesi ve görüntü donanımı bağlantısı üzerine bilgi sağlayan makrolar ve eşdeğer fonksiyonlardır.

**Çizim:** Nokta, çizgi, dörtgen, poligon ve yay çizimleri ile bunlardan son üçünün doldurulmasına ilişkin rutinlerdir.

**Hatalar:** Bir hata oluştuğu zaman çağrılan fonksiyonları yerleştiren rutinlerdir.

**Olaylar:** Kullanıcıdan, diğer uygulamalardan ve diğer sunuculardan gelen girişleri alan rutinlerdir. X'de bütün bunlara olaylar adı verilir.

**Fontlar:** Mevcut fontların listelenmesini, fontların yüklenmesini ve karakteristiklerinin elde edilmesini sağlayan rutinlerdir.

**Geometri:** Geometri özelliklerini kullanan ve çeviren rutinlerdir.

**Grafik İçeriği:** Yorumlanan çizim isteklerinin özelliklerini yerleştiren rutinlerdir.

**İmajlar:** Ekran imajlarını elde eden, gösteren veya işleyen rutinlerdir.

**İstemciler Arası Haberleşme:** Herhangi bir istemcinin diğer istemcilerin okuyacağı bilgileri oluşturmasına izin veren rutinlerdir.

**Uluslararası Duruma Getirme:** Dilden bağımsız olarak kullanıcı girişi elde edimi ve yazı çizimi fonksiyonlarıdır.

**Klavye:** Klavye tasvirini kullanarak, klavye girişlerini düzenleyen fonksiyonlardır.

**İşaretçi:** İşaretçi girişlerini düzenleyen fonksiyonlardır.

**Bölgeler:** Poligonal bölgeler üzerinde matematik işlemler icra eden rutinlerdir.

**Kaynak Yönetimi:** Kullanıcı tercihlerinin yönetimini yapan ve komut satırı girişlerini kolaylaştıran rutinlerdir.

**Ekran Koruyucular:** Bir süre zarfında klavye ve işaretçi girişi olmadığı zaman ekranı karartan uygulamanın operasyonel karakteristiklerini yerleştiren rutinlerdir.

**Yazı:** Çizilecek olan karakter zincirinin belirlenmesi ve yazı çizimi rutinleridir.
**Kullanıcı Tercihleri:** Klavye tıklamalarının yerleştirimi ve elde edimi ile oto-tekrar(auto-repeat) yerleştirimlerini yapan rutinlerdir.
**Pencere Özellikleri:** Bir pencerenin o anki karakteristiklerini elde eden ve yerleştiren rutinlerdir.
**Pencere Hayatı:** Bir pencereyi yaratan ve yok eden rutinlerdir.
**Pencere Yönetimi:** Pencerenin ekranda büyüklüğünün değişmesi, görünülebilirliği, öteki pencerelerin üstte veya altta kalmasına göre görünür pozisyonları gibi işlemlere izin veren rutinlerdir.

Görüldüğü gibi Xlib pekçok fonksiyona destek vermektedir. X herhangi bir stilde kullanıcı arayüzüne izin verecek şekilde tasarlanmış olup birçok esnek rutine ihtiyaç duymaktadır. Fakat bütün rutinler normal uygulama yazılımları için zorunlu veya istenilen değildir. Rutinlerin çoğu pencere yönetimi için veya öteki özel amaçlar için istenilenlerdir [1], [2], [3], [4].

# BÖLÜM 2. X RUTİNLERİ

X üzerinde yazılan programlar, X rutinlerini çağırarak görüntü üzerinde grafik çıktısı elde ederler. Bu bölümde X kullanarak grafik çıktı elde etmek için kullanılan rutinler verilmektedir.

## 2.1 Bir Görüntü Sunucusu Açma

Herhangi bir pencere kullanılmadan önce, bir görüntü sunucusu ile bağlantı sağlanması gereklidir. Bir sunucu, üzerinde bulunduğu makinenin ismi ve görüntü numarası ile tanımlanır. Eğer sunucu ismi katarı `NULL` ise `DISPLAY` çevre değişkeninde(environment variable) bulunan değer, sunucunun kullanacağı görüntü olarak kabul edilir. Tam görüntü ismi ise şu formdadır:

`makine_ismi:görüntü_numarası.ekran`

Çoklu ekranlar aynı görüntü üzerinde bulunan tek bir X sunucusu tarafından kontrol edilebilir.Örneğin bu ekranlar tek bir fare ve tek bir klavye ile kontrol edilirler.

`Display *XOpenDisplay(name)`

İsimlendirilmiş sunucu üzerinde bir görüntü açar. Eğer başarılı olursa görüntü üzerindeki bütün ekranlar kullanılabilir.

`char *name;` (örnek; candy:0.0)

`XCloseDisplay(display)`

Bir görüntüyü tamamen kapatır.

Bir istemcinin bir görüntüye bağlantısı kapatıldığı zaman sunucu tarafından istemci adına tutulan kaynaklar serbest bırakılır. X sunucusuna olan son bağlantı kapatıldığı zaman çoğu kaynaklar serbest bırakılır ve sunucu bir sıfırlama(reset) yapar.

## 2.2 Görüntü Hakkında Bilgi Elde Etme

Bazı basit rutinler görüntünün genel karakteristiklerini keşfetmek için kullanılabilir.

`int DefaultScreen(display)`

Önceden belirlenmiş ekran numarasını geri döndürür. Yalnızca istenildiğinde kullanılır.

`int ScreenCount(display)`

Olası ekran numarasını geri döndürür.

`int DefaultDept()`

Bir grafik görüntüsü, bir pixelin etkilediği her katmanda yalnızca bir bit olmak üzere bit katmanlarının çakışması olarak düşünülebilir. Bu rutin, kullanılan katmanların sayısını döndürür.

`unsigned long AllPlanes()`

Katmanları temsil eden dönüş değerindeki bitler için kullanılır.

`int DisplayPlanes(display,screen_number)`

Olası katmanların sayısını döndürür.

`Colormap DefaultColormap(display,screen_number)`

`Colormap` olarak adlandırılan bir tabloda depolanmış o anki renkler hakkında bilgi verir.

`int DisplayCells(display,screen_number)`

Önceden belirlenmiş `colormap`'deki girişlerin değerini döndürür.

`unsigned long BlackPixel(display,screen_number)`

Bunlar kalıcıdır.

`unsigned long WhitePixel(display,screen_number)`

Siyah ve beyaz için önceden belirlenmiş `colormap`'deki girişlere yer açar. Bu rutinler renklere referans için kullanılan değeri geri döndürür.

`GC DefaultGC(display,screen_number)`

Bir grafik içeriği yapısından o anki ön fon rengi gibi detayları elde etmek için kullanılan bir grafik komutudur. Bu rutin önceden belirlenmiş bir grafik içeriği döndürür.

`Window DefaultRootWindow(display)`

Ekranı kaplayan ilk pencereyi geri döndürür.

`Visual *DefaultVisual(display,screen_number)`

Bazı çeşitli donanımlar görünmezliği destekleyebilir, bazı ayrıntılı donanım bilgisi görünüm yapısında depolanır.

`int DisplayHeight(display,screen_number)`

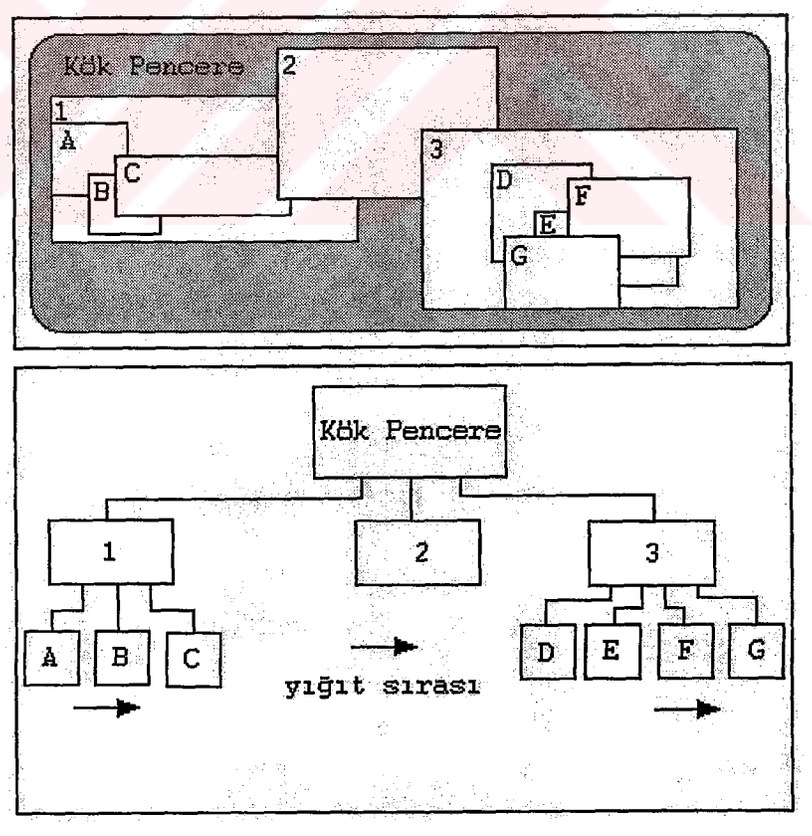
Pixel olarak görüntünün yüksekliğini geri döndürür.

`int DisplayWidth(display,screen_number)`

Pixel olarak görüntünün enini geri döndürür.

## 2.3 Pencere Yaratma ve Yok Etme

Sunucuyu kullanan her bir program, pencereye çizmeden veya yazmadan önce bir veya birden fazla pencere yaratır. Pencereler, bütün ekranı kaplayan kök pencereden aşağıya doğru bir ağaç yapısı şeklinde büyüyen yapılardır.




Şekil 2.1 Bir örnek pencere hiyerarşisinin ekranda ve şematik olmak üzere iki şekilde gösterilimi

Çoğu uygulamalar yalnızca, ebeveyni kök olan pencereleri kullanır. Eğer torun pencere yaratılırsa bu çocuk pencere ebeveyni sanki bütün fiziksel görüntüyü kaplıyormuş gibi davranır. Çocuk pencereye bütün çıkışlar ebeveynin sınırlarında kırpılır.

Hiçbir pencere tasvir edilmedikçe, bütün ataları tasvir edilmedikçe ve açığa çıkma olayı olmadıkça görüntüde görünmez. Bir pencereyi tasvir ederken bütün tamponlar boşaltılmalı veya ağ içinde tasvir komutunu görüntünün bir yerinde çiz denilmelidir. Bir pencereyi yalnızca yaratmak görüntüde gerçekten birşey görmek için yeterli değildir.

Eğer pekçok şey sıkıntılı şekilde yerleştirilmek istenmiyorsa pencereler ebeveynlerinin özelliklerini miras alarak yaratılabilir. Şöyle ki;

```
WindowXCreateSimpleWindow(display, parent, x, y, width, height,
borderwidth, border,background)
Display *display;
```

XOpenDisplay'den dönen değer

```
Window parent;
```

ebeveyn pencere muhtemelen kök pencere

```
int x, y;
```

pencere sınırının sol üst köşesi

```
unsigned int width, height;
```

pencerenin en ve boy bilgisi

```
unsigned int border;
```

pencerenin sınır pixeli

```
unsigned long background;
```

pencerenin arka fon pixeli

Daha sofistike pencere kullanımı için;

```
Window XCreateWindow(display, parent, x, y, width, height,
borderwidth, depth, class, visual, valuemask, attributes)
```

ile pencere yaratılır.

```
Window parent;
```

ebeveyn pencere muhtemelen kök pencere

`int x, y;`

pencere sınırının sol üst köşesi

`unsigned int width, height;`

pencerenin en ve boy bilgisi

`unsigned int borderwidth;`

pencerenin sınır genişliği

`int depth;`

`unsigned int class;`

GirişÇıkış(InputOutput),

SadeceGiriş(InputOnly),

EbeveyndenKopya(CopyFromParent) sınıfları

SadeceGiriş pencereleri şeffaftır.

`Visual *visual;`

EbeveyndenKopya vb.

`unsigned long valuemask;`

Özellik alanlarını belirler.

`XSetWindowAttributes *attributes;`

Özellikleri içerir.

Pencereler ilk yaratıldığı zaman aşağıdaki yapı kullanılır. Bu yapıda ilk kullanılan veri 'varsayılan değer'i, ikinci veri ise 'maske'yi ifade etmektedir.

```
typedef struct {
Pixmap background_pixmap;
    None, CWBackPixmap
unsigned long background_pixel;
    -, CWBackPixel
Pixmap border_pixmap;
    CopyFromParent, CWBackPixmap
unsigned long border_pixel;
    -, CWBorderPixel
int bit_gravity;
    ForgetGravity, CWBitGravity
```

```
int win_gravity;
    NorthWestGravity, CWWinGravity
int backing_store;
    NotUseful, CWBackingStore
unsigned long backing_planes;
    ~0, CWBackingPlanes
unsigned long backing_pixel;
    False, CWBackingPixel
Bool save_under;
    0, CWSaveUnder
long event_mask;
    0, CWEventMask
long do_not_propagate_mask;
    0, CWDontPropagate
Bool override_redirect;
    False, CWOverrideRedirect
Colormap colormap;
    CopyFromParent, CWColormap
Cursor cursor;
    None, CWCursor
} XSetWindowAttributes;
```

**background_pixmap:** Bu örüntülü arka fon, bir örüntülü arka fona sahip olunmasına izin verir. `background_pixmap`, pencere ile aynı köke ve aynı derinliğe sahip olmalıdır. Eğer değeri `None` ise arka fon yoktur. Eğer değeri `ParentRelative`(EbeveynİlişkiIi) ise ebeveynin arka fonu kullanılır fakat ebeveyn ve çocuk aynı derinliğe sahip olmalıdır.

**border_pixmap:** `border_pixmap` ve pencere aynı köke ve derinliğe sahip olmalıdır. Eğer değilse bir `BadMatch`(KötüEşleme) hatası oluşur. Eğer değeri `CopyFromParent` ise ebeveynin pixmap'i kopya edilir.

**bit_gravity:** Pencerenin yeniden boyutlandırıldığı bölgeleri tutmak için tanımlanır.

**win_gravity:** Bir çocuk pencerenin ebeveyni yeniden boyutlandırıldığında o çocuk pencerenin nasıl yeni pozisyon alacağını tanımlar.

**backing_store:** Değeri WhenMapped ise pencere tarafından kapatılan bölgeleri saklamasını sunucuya tavsiye eder. Değeri `Always` ise tasvir edilmedikçe bile içeriğin saklanmasını sunucuya tavsiye eder.

**backing_planes:** Hangi katmanların saklanacağını tanımlar.

**backing_pixel:** Saklanmayan katmanlara konan değeri tanımlar.

**save_under:** Eğer değeri `True` ise pencere tarafından kapatılan bölgeleri saklamasını sunucuya tavsiye eder.

**do_not_propagate_mask:** Atalara nakledilmeyen olayları tanımlar.

**override_redirect:** Eğer değeri `True` ise pencere yöneticisi pencere hakkında bilgiye sahip olmaz.

**colormap:** `colormap`, pencere ile aynı görünüm tipine sahip olmalıdır. Eğer değeri `CopyFromParent` ise ebeveynin renk tasviri kopya edilir.

Bir pencereyi yok etmek için bir çift çağrı vardır. Pencere yok edilmeden önce otomatik olarak tasviri çıkarılır (unmapped).

```
XDestroyWindow(display, w)
```

Kimlik bilgisi verilen pencere yok edilir.

```
XDestroySubWindows(display, w)
Window w;
```

Belirlenen pencerenin bütün alt pencereleri yok edilir. Pencerenin kendisi yok edilmez.

Bir pencerenin özellikleri aşağıdaki çağrı ve yapı kullanılarak elde edilebilir.

```
Status XGetWindowAttributes(display,w,window_attributes)
XWindowAttributes *window_attributes;
```

Pencerenin tamamlanmış yapısı elde edilir.

```
typedef struct {
int x, y;
```

pencerenin yeri

`int width, height;`

pencerenin eni ve yüksekliği

`int border_width;`

pencerenin sınır eni

`int depth;`

pencerenin derinliği

`Visual *visual;`

bağlantılı görünüm yapısı

`Window root;`

pencereyi içeren ekranın kökü

`int class;`

GirişÇıkış(InputOutput), SadeceGiriş(InputOnly) sınıfları

`int bit_gravity;`

bit çekim değerlerinden biri

`int win_gravity;`

pencere çekim değerlerinden biri

`int backing_store;`

`NotUseful, WhenMapped, Always`

`unsigned long backing_planes;`

eğer mümkünse korunmuş katmanlar

`unsigned long backing_pixel;`

katmanlar yerleştirilirken kullanılan değer

`Bool save_under;`

`True, False`

`Colormap colormap;`

pencere ile bağlantılı renk tasviri, renk bağlantılı görüntü tasviri

`Bool map_installed;`

renk tasviri halihazırda kurulu mu?

`int map_state;`

`IsUnmapped, IsUnviewable, IsViewable`

`long all_event_mask;`

herkesi ilgilendiren olaylar

```
long your_event_mask;
```

pencerenin olay maskesi

```
long do_not_propagate_mask;
```

yayılamayan olaylar

```
Bool override_redirect;
```

`override_redirect` için mantıksal değer

```
} XWindowAttributes;
```

## 2.4 Pencerelerin Tasvir Edilmesi, Hareket Ettirilmesi ve Ortaya Çıkarılması

Pencerelerin kullandığı tasvir fonksiyonları şunlardır:

```
XMapWindow(display, w)
XMapsubwindows(display, w)
XUnmapWindow(display, w)
XUnmapSubwindows(display, w)
```

Bir masa üzerindeki kağıt yığınındaki kağıtların üste ve alta karıştırılması gibi pencerelerin de etrafta bu şekilde hareket etmesi için kullanılan, biraz da kendini açıklayıcı çağrılar da vardır. Pencere yöneticisi isteklere göre müdahale edebilir.

```
XChangeWindowAttributes(display, w, valuemask, attributes)
XSetWindowAttributes *attributes;
XConfigureWindow(display, w, x, y, valuemask, values)
unsigned int valuemask;
```

değerin ilgili alanlarının seçimi

```
XWindowChanges *values;
```

Yeni detayları içerir.

```
XResizeWindow(display, w, width, height)
XMoveResizeWindow(display, w, x, y, width, height)
XSetWindowBorderWidth(display, w, width)
XRaiseWindow(display, w)
```

Pencereyi öne çıkarır.

`XLowerWindow(display,w)`

Pencereyi arkaya düşürür.

`XMoveWindow(display,w,x,y)`

Pencereyi hareket ettirir ve öne çıkarır.

`XCirculateSubwindows(display,w,direction)`

Yön değeri `RaiseLowest` veya `LowerHighest olabilir.`

`XCirculateSubwindowsUp(display,w)`

En altta tasvir edilmiş çocuğu en yukarı çıkarır.

`XCirculateSubwindowsDown(display,w)`

En yukarıda tasvir edilmiş çocuğu en aşağıya indirir.

## 2.5 Bitmapler ve Pixmaplerin Tanımlanması

Ekransız bellek(bir pixmap), grafik operasyonlarda daha sonra kullanılacak imajları tanımlamak için sıkça kullanılır. Pixmapler ayrıca pencere arka fonlarının, sınırların veya imleçlerin kullanımı için örgü veya örüntü tanımlamada kullanılır. Tek bit katmanlı pixmaplere bazen bitmap denir. Bir pixmap yaratmak için şu çağrı yapılır:

```
Pixmap XCreatePixmap(display, d, width, height, depth)
unsigned int width, height, depth;
```

Bir bitmapi bir dosyadan(belki de bitmap programı tarafından yaratılmış bir dosya) okumak için şu çağrı yapılır:

`int XReadBitmapFile(display, d, filename, width, height, bitmap, x_hot, y_hot)`

Mümkün dönüş değerleri `BitmapOpenFailed, BitmapFileInvalid, BitmapNoMemory, BitmapSuccess`'dir.

`int *width, height;`

genişlik ve yükseklik değerleri

`Pixmap *bitmap;`

bitmap görüntünün bulunduğu adresi gösteren değer

`int *x_hot,*y_hot;`

Bir bitmapi bir dosyaya yazmak için şu çağrı yapılır:

```
int XWriteBitmapFile(display, filename, bitmap, width, height,
x_hot, y_hot)
```

Mümkün dönüş değerleri `BitmapOpenFailed`, `BitmapNoMemory`, `BitmapSuccess`'dir.

```
int width, height;
Pixmap bitmap;
int x_hot, y_hot;
```

Programlarda bir bitmap dosyası #include komutu ile içeri alınabilir ve şu şekilde kullanılabilir:

```
PixmapXCreateBitmapFromData(display,d,data,width,height)
char *data;
```

Pixmapler fazla miktarda bellek kullandıkları için işleri bittiğinde mümkün olduğunca çabuk, şu çağrı yapılmalıdır:

```
XFreePixmap(display, pixmap)
```

# BÖLÜM 3. GRAFİK İÇERİĞİ

Yazı ve grafik çıkış rutinleri bir grafik içeriği (Graphics Context -GC-) içinde işlem görürler. Grafik içeriği değerleri aşağıdaki yapı ile verilmektedir. Bu yapıda ilk kullanılan veri 'varsayılan değer'i, ikinci veri ise 'maske'yi ifade etmektedir.

```
typedef struct {
int function;
    DXcopy, GCFunction
unsigned long plane_mask;
    ~0, GCPlaneMask
unsigned long foreground;
    0, GCForeground
unsigned long background;
    1, GCBackground
int line_width;
    0, GCLineWidth
int line_style;
    LineSolid, GCLineStyle
int cap_style;
    CapButt, GCCapStyle
int join_style;
    JoinMiter, GCJoinStyle
int fill_style;
    FillSolid, GCFillStyle
```

```
int fill_rull;
    EvenOddRule, GCFillRule
int arc_mode;
    ArcPieSlice, GCArcMode
Pixmap tile;
    -, GCTile
Pixmap stipple;
    -, GCStipple
int ts_x_origin;
    0, GCTileStipXOrigin
int ts_y_origin;
    0, GCTileStipYOrigin
Font font;
    -, GCFont
int subwindow_mode;
    ClipByChildren, GCSubwindowMode
Bool graphics_exposures;
    True, GCGraphicsExposures
int clip_x_origin;
    0, GCClipXOrigin
int clip_y_origin;
    0, GCClipYOrigin
Pixmap clip_mask;
    None, GCClipMask
int dash_offset;
    0, GCDashOffset
char dashes;
    4, GCDashList
} XGCValues;
```

**function:** Ekrana birşeyler çizileceği zaman makine, çizilmiş olan pixel değerlerini alır ve bir görüntü fonksiyonu kullanılarak ekranda halihazırda bulunan

pixelle birleştirir. Görüntü fonksiyonu, ekranda özel noktadaki yeni pixel haline gelen bir pixel çıkış değeri üretir. Ekrandaki eski pixel değeri "dst" ve çizilecek olan pixel de "src" olarak kabul edilirse Tablo 3.1'de, mevcut değişik görüntü fonksiyonlarının etkilerinin listesi görülebilir.

TABLO 3.1 - Grafik İçeriğinde fonksiyonlar ve etkileri

| Fonksiyon İsmi | Fonksiyon Etkisi |
|---|---|
| GXclear | 0 |
| GXand | src AND dst |
| GXandReverse | src AND NOT dst |
| GXcopy | src |
| GXandInterved | (NOT src) AND dst |
| GXnoop | dst |
| GXxor | src XOR dst |
| GXor | src OR dst |
| GXnor | (NOT src) AND NOT dst |
| GXequiv | (NOT src) XOR dst |
| GXinvert | NOT dst |
| GXorReverse | src OR NOT dst |
| GXcopyInverted | NOT src |
| GXorInverted | (NOT src) OR dst |
| GXnand | (NOT src) OR NOT dst |
| GXset | 1 |

**line_style:** `LineSolid, LineDoubleDash, LineOnOffDash`

**cap_style:** `CapNotLAst, CapButt, CapRound, CapProjecting`

**join_style:** `JoinMiter, JoinRound, JoinBevel`

**fill_style:** `FillSolid, FillTiled, FillOpaqueStippled`

**fill_rule:** `EvenOddRule, WindingRule`

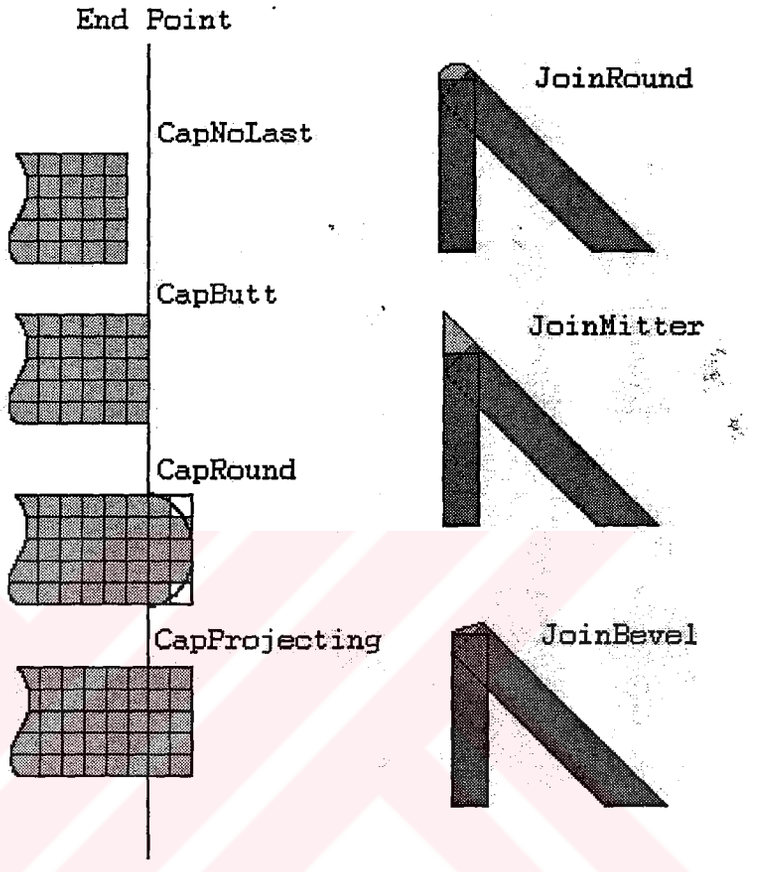
**arc_mode:** `ArcPieSlice, ArcChord`

**stipple:** Gölgeleme işlemlerinde kullanılan bir bitmaptir.

**ts_x_origin, ts_y_origin:** Örgü/gölgeleme işlemlerinin başlangıç yerleridir.

**subwindow_mode:** `ClipByChildren, IncludeInferiors`

**graphics_exposures:** Mantıksal bir değerdir, saklanmış bir alana çizim işlemi yapılırsa oluşan olaylar üretilmeli midir? sorusunun cevabını tutar.




Şekil 3.1 LineCap ve LineJoin stilleri

Grafik içerikleri yaratmak, değiştirmek veya serbest bırakmak için şu komutlar kullanılır:

```
GC XCreateGC(display, d, valuemask, values)
XGCValues *values;
XCopyGC(display, src, valuemask, dest)
GC src, dest;
XChangeGC(display, gc, valuemask, values)
XFreeGC(display, gc)
```

Aşağıdaki rutinler kullanılarak grafik içeriklerinin içerikleri teker teker veya hepsi birden değiştirilebilir.

XSetState(display, gc, foreground, background, function, planemask)

Bu rutin, aşağıdaki dört rutinden oluşan bir pakettir.

XSetForeground(display, gc, foreground)

XSetBackground(display, gc, background)

XSetFunction(display, gc, function)

XSetPlaneMask(display, gc, planemask)

XSetLineAttributes(display, gc, line_width, line_style, cap_style, join_style)

XSetDashes(display, gc, dash_offset, dash_list,n)

XSetFillStyle(display, gc, fill_style)

XSetFillRule(display, gc, fill_rule)

XQueryBestSize(display, class, d, width, height, rwidth, rheight)

XQueryBestTile(display, gc, d, width, height, rwidth, rheight)

XQueryBestStipple(display, gc, d, width, height, rwidth, rheight)

XSetTile(display, gc, tile)

XSetStipple(display, gc, stipple)

XSetFont(display, gc, font)

XSetClipOrigin(display, gc, clip_x_origin, clip_y_origin)

XSetClipMask(display, gc, pixmap)

XSetClipRectangles(display, gc, clip_x_origin, clip_y_origin, rectangles, n, ordering)

XSetArcMode(display, gc, arc_mode)

XSetSubwindowMode(display, gc, subwindow)

XSetGraphicsExposures(display, gc, graphics_exposures)

# BÖLÜM 4. DÖRTGENLER, ÇİZGİLER, NOKTALAR, YAYLAR VE YAZI

İlk olarak, bazı elementel yapılar şu şekilde verilmiştir:

```
typedef struct {
    short x, y;
} XPoint;

typedef struct {
    short x1, y1, x2, y2;
} XSegment;
typedef struct {
    short x, y;
    unsigned short width, height;
} XRectangle;
typedef struct {
    short x, y;
    unsigned short width, height;
    short angle1, angle2;
} XArc;
```

Aşağıdaki rutinler grafik içeriği kapsamlı olarak kullanır. Sıfır enindeki çizgiler farklı ve hızlı bir algoritma tarafından kalınlığı bir olmak üzere çizilir. Bu rutinler çizilebilirlere (grafik girişlerini alabilen pixmaplere veya pencerelere) yazılırlar.

```
XDrawPoint(display, d, gc, x, y)
XDrawPoints(display, d, gc, points, npoints, mode)
int mode;
    CoordModeOrigin veya CoordModePrevious
```

```
XDrawLine(display, d, gc, x1, y1, x2, y2)
XDrawLines(display, d, gc, points, npoints, mode)
int mode;
```

CoordModeOrigin veya CoordModePrevious

```
XDrawSegments(display, d, gc, segments, nsegments)
XDrawRectangle(display, d, gc, x, y, width, height)
XDrawRectangles(display, d, gc, rectangles, nrectangles)
XDrawArc(display, d, gc, x, y, width, height, angle1, angle2)
unsigned int width, height;
```

yayın büyük ve küçük eksenleri

```
int angle1;
```

yayın başlangıcı, saat 3'den itibaren saat yönlü, 64 ile ölçülendirilmiş

```
int angle2;
```

yayın yönü ve genişliği, açı1 (angle1) ile karşılaştırmalı (relative), 64 ile ölçülendirilmiş

```
XDrawArcs(display, d, gc, arcs, narcs)
XFillRectangle(display, d, gc, x, y, width, height)
XFillRectangles(display, d, gc, rectangles, nrectangles)
XFillPolygon(display, d, gc, points, npoints, shape, mode)
int shape;
```

Kompleks(Complex), Konveks(Convex), Konveks Olmayan(NonConvex);
En iyi doldurma algoritmasını seçmeye yardım eder.

```
int mode;
```

CoordModeOrigin veya CoordModePrevious

```
XFillArc(display, d, gc, x, y, width, height, angle1, angle2)
XFillArcs(display, d, gc, arcs, narcs)
XMoveArea(display, w, srcX, srcY, dstX, dstY, widht, height)
int srcX, srcY;
```

bölgenin sol üst köşesinin pozisyonu

```
int dstX, dstY;
```

Sol üst köşenin sonlanması gereken yerdir.

`int width, height;`

taşınabilir bölgenin büyüklüğü

`XCopyArea(display, src, dest, gc, src_x, src_y, width, height, dest_x, dest_y)`

`Drawable src_x, src_y;`

bölgenin sol üst köşesinin pozisyonu

`int src_x, src_y, dest_x, dest_y;`

Sol üst köşenin sonlanması gereken yerdir.

`unsigned int width, heiht;`

kopyalanabilir bölgenin büyüklüğü. Kaynak (source) ve hedefin (destination) derinliği aynı olmalıdır.

`XCopyPlane(display, src, dest, gc, src_x, src_y, width, height, dest_x, dest_y, plane)`

src(kaynak) ve dest(hedef) aynı kök ile çizilebilirdir.

`unsigned int width, height;`

`unsigned long plane;`

`XClearArea(display, w, x, y, width, height, exposures)`

`int width, height;`

Eğer width=0 ise width, window_width-x değerinde olur benzer şekilde y için düşünülürse eğer height=0 ise height, window_height-y olur.

`Bool exposures;`

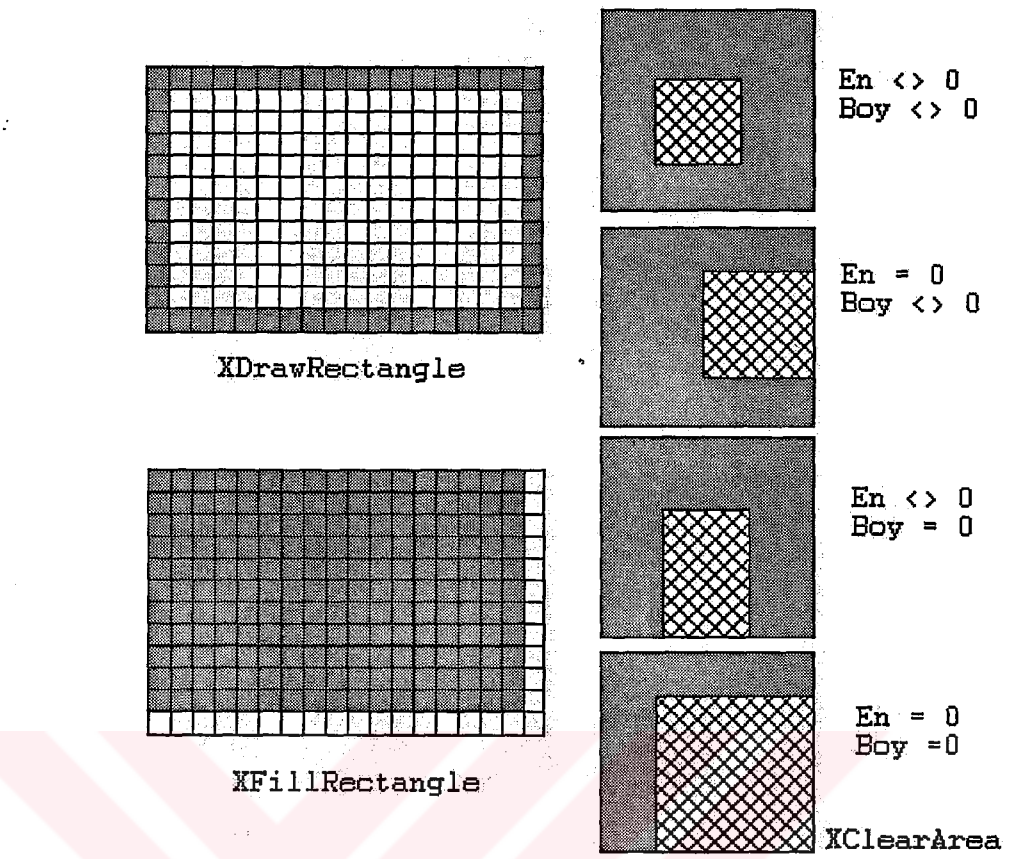
Eğer True ise exposure olayları üretilmiştir.

`XClearWindow(display, w)`

Pencereyi temizler.

## 4.1 Pencerede Yazı Çizimi

Fontu kullanmanın en iyi yolu, bir `XFontStruct` yapısını dolduran bir işaretçiyi geri döndüren ve fontu yükleyen `XLoadQueryFont` denilen rutinin çağırılmasıdır. fid alanı, halihazırdaki fontun yerleştirildiğinde kullanılan değişkendir.




Şekil 4.1 `XDrawRectangle()`, `XFillRectangle` ve `XClearArea()` fonksiyonlarının sonuçlarının gösterilimi

```
typedef struct {
XExtData *ext_data;
```

veri genişlemesine çengel

```
Font fid;
```

bu fontun font kimliği

```
unsigned direction;
```

çizilecek fontun yönü

```
unsigned min_char_or_byte2;
```

ilk karakter

```
unsigned max_char_or_byte2;
```

son karakter

```
unsigned min_byte1;
```

mevcut ilk satır

```
unsigned max_byte1;
```

mevcut son satır

```
Bool all_chars_exist;
```

Bütün karakterler mevcutsa bayrak kaldırır.

```
unsigned default_char;
```

Tanımlanmamış karakter için basılacak karakter

```
int n_properties;
```

Ne kadar özellik olduğunun sayısını tutar.

```
XFontProp *properties;
```

ek özelliklere işaretçi

```
XCharStruct min_bounds;
```

bütün mevcut karakterlerin minimum sınırı

```
XCharStruct max_bounds;
```

bütün mevcut karakterlerin maksimum sınırı

```
XCharStruct *per_char;
```

ilk karakterden son karaktere bilgi

```
int ascent;
```

boşluk için logaritmik artımlı taban çizgi

```
int descent;
```

boşluk için logaritmik azalımlı taban çizgi

```
} XFontStruct;
XFontStruct *XLoadQueryFont(display, name)
```

eylem için bir fontun hazırlanması

```
char name;
```

.onx'siz fontun ismi

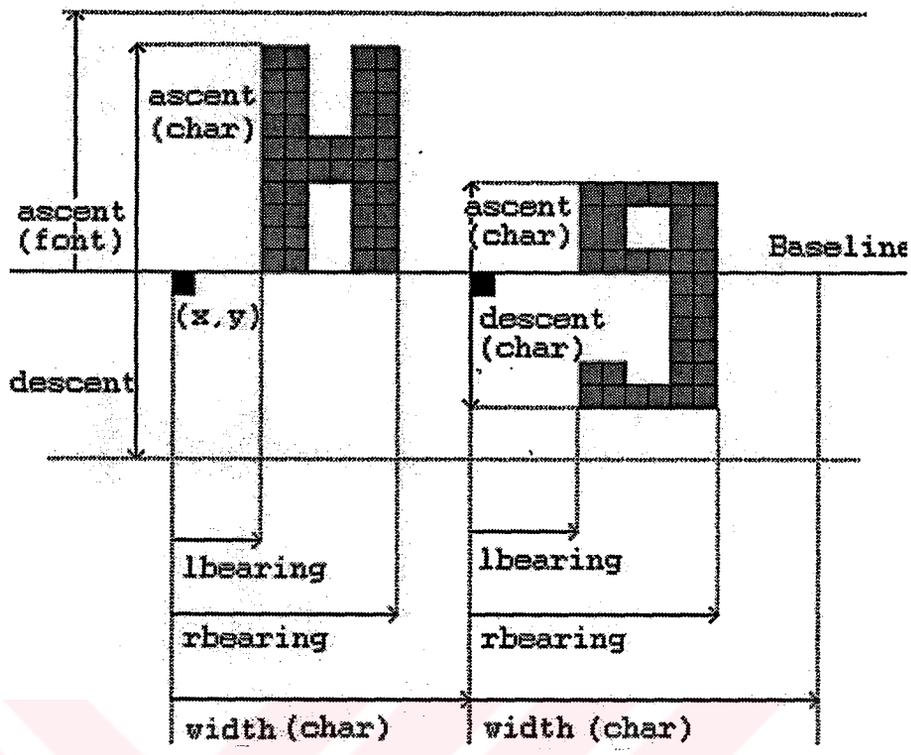
```
XCloseFont(info)
```

font tarafından kullanılan kaynakların bırakılması

```
FontInfo *info;
XFreeFont(display, font_struct)
```

Belirtilen fonta artık ihtiyaç olunmadığını, fontun kullanılmayacağını sunucuya söyler.

```
XFontStruct *font_struct;
```




Şekil 4.2 İki örnek karakterin metrikleri

## 4.2 Karakterlerin Eninin Bulunması

`XTextWidth` fonksiyonu yerel olarak 8 bitlik bir karakter katarının pixellerindeki genişliği verir. Bu fonksiyonun yapısı şu şekildedir:

```
int XTextWidth(font_struct, string, count)
XFontStruct *font_struct;
```

kullanılacak fontun font kimliği

```
char *string;
```

null bitimli karakter katarı

```
int count;
```

katardaki karakter sayısı

`XTextWidth16` fonksiyonu yerel olarak 16 bitlik bir karakter katarının pixellerindeki genişliği verir. Bu fonksiyonun yapısı şu şekildedir:

```
int XTextWidth16(font_struct,string,count)
```

`XFontStruct *font_struct;`

kullanılacak fontun font kimliği

`char *string;`

null bitimli karakter katarı

`int count;`

katardaki karakter sayısı

`XTextExtends` fonksiyonu yerel olarak katar ve font metriğini döndürür.

`XTextExtends(font_struct, string, nchars, direction, ascend, descend, overall)`

`XFontStruct *font_struct;`

kullanılacak fontun font kimliği

`char *string;`

null bitimli karakter katarı

`int nchars;`

katardaki karakter sayısı

`int *direction;`

`XFontStruct`'ın yön elemanının değerini döndürür. Bu değerler `FontRightToLeft` veya `FontLeftToRight` olabilir.

`int *ascend;`

`XFontstruct`'ın font artım elemanını döndürür. Bu değer fontun maksimum artım değeridir.

`int *descend;`

`XFontstruct`'ın font azalım elemanını döndürür. Bu değer fontun maksimum azalım değeridir.

`XCharStruct *overall;`

katarın özelliklerini döndürür.

`XTextExtends16` fonksiyonu, `XTextExtends` fonksiyonunun 16 bitlik karakter katarları üzerine uygulanmış halidir.

## 4.3 Ekranda Karakterlerin Basımı

Aşağıdaki fonksiyonlar tek bir fontun katarları için kullanılır.

```
XDrawString(display, d, gc, x, y, string, length)
XDrawImageString(display, d, gc, x, y, string, length)
```

Arka fonu da çizer.

```
XDrawImageString16(display, d, gc, x, y, string, length)
```

Arka fonu da çizer.

Diğer durumlarda bastırılacak yazı, aşağıdaki yapıda tutulan bilgilere de ihtiyaç duyar:

```
typedef struct {
char *chars;
```

katara işaretçi

```
int nchars;
```

karakter sayısı

```
int delta;
```

katarlar arası aralık

```
Font font;
```

Basımda kullanılacak font `None` ise font değişmez.

```
} XTextItem;
XDrawText(display, d, gc, x, y, items, nitems)
XTextItem *items;
XDrawText16(display, d, gc, x, y, items, nitems)
XTextItem *items;
```

## 4.4 İmleç Yaratma

X, bir imlecin her nerede olursa olsun her bir pencere için kayıt edilebilmesi ve fare her ne zaman o pencerede bulunursa bulunsun imlecin doğru şekli alması gibi bir otomatik kolaylığı sağlar. Bir pencere için özel bir imleç kayıt edilmemişse fare, ebeveyn pencerenin imlecini kullanır.

Üç imleç yaratma rutini vardır. Birincisi en basitidir:

```
Cursor XCreateFontCursor(display, shape);
```

Faydalı bitmap, standart kütüphane içinde olmayan bir imleç olarak kullanılmak istendiğinde bitmap dizaynı için kullanışlıdır. Böyle imleçler iki bitmap ve iki pixel değeri terimleri ile tanımlanırlar. İmleç bitmapi; maske bitmapi, imlecin düzenlemesine izin verilen alanları tanımlarken görüntü fonksiyonlarını ve ön font ile arka fontları kullanarak imlecin şeklini tanımlar. Her imleç, X sunucusunun ekran üzerinde olmasını istediği yerdeki noktanın koordinatlarını tanımlayan bir veriye sahiptir. Bu noktanın göreceli koordinatları(relative coordinates) da bu düzeyde verilir.

```
Cursor XCreateGlyphCursor(display, sfont, mfont, schar, mchar,
scolor, bcolor)
Font sfont;
```

imleç için glyph fontu

```
Font mfont;
```

maske fontu

```
unsigned int schar;
```

imlecin şekli

```
unsigned int mchar;
```

karakter maskesi

```
XColor *scolor;
```

RGB(RedGreenBlue) ön font(ön plan)

```
XColor *bcolor;
```

RGB(RedGreenBlue) arka font(arka plan)

```
Cursor XCreatePixmapCursor(display, source, mask, scolor,
bcolor, x, y)
Pixmap source;
Pixmap mask;
XColor scolor;
XColor bcolor;
unsigned int x, y;
```

```
XFreeCursor(display, cursor)
```

Önceden yaratılmış bir imleci kaldırır.

```
Cursor cursor;
XDefineCursor(display, w, cursor)
Cursor cursor;
```

İşaretçi, pencere içinde olduğu zaman bu imleci kullanır.

```
XUndefineCursor(display, w)
```

Fare ebeveynin imlecini kullanacaktır.

## BÖLÜM 5. OLAYLAR

Sunucu klavyesi üzerindeki bir tuşa basılırsa veya bir pencere parçası örtülmezse sunucu, olaylar göndererek etkileşimli olan istemciye danışabilir. Olaylar genellikle pencerelere ulaştırılır ve her bir pencere kendisiyle ilgili olayları sunucuya söylemelidir. Sunucu bu kullanımı, `XSelectInput` çağrısı ile yapar. Eğer bir olay seçilmemişse, `do_not_propagate_mask`'ın nasıl kurulduğuna bağlı olarak o olay ya boşa çıkarılacaktır ya da başka bir pencereye geçilecektir.

Bir pencere için, klavye ve fareden bütün olayları almak mümkündür.

Bu olaylar, pencere sistemi ve alt rutinler tarafından kuyruğa alınır. Bir olayın ne anlama geldiğini sırayla bulup araştırmak için olay yapısının okunabilmesi gereklidir. Bir `XEvent` yapısı her zaman ilk giriş gibi type'a sahiptir. `XEvent` yapısı sadece olayın tipinin ne olduğunu tanımlar. İkinci giriş her zaman olayın okunduğu görüntüye bir işaretçidir. Üçüncü giriş her zaman bir pencere tipi veya başka birşeydir (Tuş tasvir olayları hariç! Bu olaylar hiç pencere içermezler.) Genel olaya işaretçi, yapı içindeki herhangi bir diğer bilgiye erişmek için kullanımdan önce tanıtılmalıdır.

`Xlib.h` kodu, `FocusIn`, `FocusOut`, `EnterNotify` ve `LeaveNotify` olayları hariç diğer tüm olayları yeterli açıklamaları ile içerir. Bu olayların (`FocusIn`, `FocusOut`, `EnterNotify` ve `LeaveNotify`) tam detayı içinse kullanım kitabına danışılmalıdır. `XEvent` bütün olay yapılarının birleşimidir. Bu olay yapılarından biri aşağıda verilmiştir:

```
typedef struct {
int type;
```

olayın tipi

```
Display *display;
```

olayın okunduğu görüntü

```
Window window;
```

olayın rapor edildiği ilişkili pencere

```
Window root;
```

olayın üzerinde gerçeklendiği kök pencere

```
Window subwindow;
```

çocuk pencere

```
unsigned long time;
```

milisaniye

```
int x, y;
```

olay penceresindeki işaretçinin koordinatları

```
int x_root, y_root;
```

kök penecereye göre koordinatlar

```
unsigned int state;
```

anahtar veya düğme maskesi

```
unsigned int keycode;
```

detay

```
Bool same_screen;
```

aynı ekran bayrağı

```
} XKeyEvent;
typedef XKeyEvent XKeyPressedEvent;
typedef XKeyEvent XKeyReleasedEvent;
```

Diğer yararlı bir olay düğme (button) basılması ile ilgilidir. Yapısı aşağıdaki şekildedir:

```
typedef struct {
int type;
```

olayın tipi

```
unsigned long serial;
```

sunucu tarafından işlenmiş son isteğin numarası

```
Bool send_event;
```

Eğer bu bir SendEvent isteğinden geldiyse değeri True'dur.

```
Display *display;
```

olayın okunduğu görüntü

```
Window window;
```

olayın rapor edildiği ilişkili olay pencere

```
Window root;
```

olayın üzerinde gerçeklendiği kök pencere

```
Window subwindow;
```

çocuk pencere

```
Time time;
```

milisaniye

```
int x, y;
```

olay penceresindeki işaretçinin koordinatları

```
int x_root, y_root;
```

kök penecereye göre koordinatlar

```
unsigned int state;
```

anahtar veya düğme maskesi

```
unsigned int button;
```

detay

```
Bool same_screen;
```

aynı ekran bayrağı

```
} XButtonEvent;
typedef XButtonEvent xButtonPressedEvent;
typedef XButtonEvent xButtonReleasedEvent;
```

## 5.1 Olay Tipleri

Olayların tam listesi Tablo 5.1'de verilmiştir;

TABLO 5.1 - Grafik İçeriğinde fonksiyon ve etkileri

| Maske | Olay Nedeni | Olay Tipi |
|---|---|---|
| NoEventMask | | |
| KeyPressMask | KeyPress | xkey |

| Maske | Olay Nedeni | Olay Tipi |
|---|---|---|
| KeyReleaseMask | KeyRelease | xkey |
| ButtonPressMask | ButtonPress | xbutton |
| ButtonReleaseMask | ButtonRelease | xbutton |
| EnterWindowMask | EnterNotify | xcrossing |
| LeaveWindowMask | LeaveNotify | xcrossing |
| PointerMotionMask | MotionNotify | xmotion |
| PointerMotionHintMask | MotionNotify | xmotion |
| Button1MotionMask | MotionNotify | xmotion |
| Button2MotionMask | MotionNotify | xmotion |
| Button3MotionMask | MotionNotify | xmotion |
| Button4MotionMask | MotionNotify | xmotion |
| Button5MotionMask | MotionNotify | xmotion |
| ButtonMotionMask | MotionNotify | xmotion |
| KeymapStateMask | KeymapNotify | xkeymap |
| ExposureMask | Expose | xexpose |
| VisibilityChangeMask | VisibiltyNotify | xvisibility |
| StructureNotifyMask | CirculateNotify | xcirculate |
| | ConfigureNotify | xconfigure |
| | DestroyNotify | xdestroywindow |
| | GravityNotify | xgravity |
| | MapNotify | xmap |
| | ReparentNotify | xreparent |
| | UnmapNotify | xunmap |
| ResizeRedirectMask | ResizeRequest | xresizerequest |
| SubstructureNotifyMask | CirculateNotify | xcirculate |
| | ConfigureNotify | xconfigure |
| | CreateNotify | xcreatewindow |
| | DestroyNotify | xdestroywindow |
| | GravityNotify | xgravity |

| Maske | Olay Nedeni | Olay Tipi |
|---|---|---|
| | MapNotify | xmap |
| | ReparentNotify | xreparent |
| | UnmapNotify | xunmap |
| SubstructureRedirectMask | CirculateRequest | xcirculaterequest |
| | ConfigureRequest | xconfigurerequest |
| | MapRequest | xmaprequest |
| FocusChangeMask | FocusIn | xfocus |
| | FocusOut | xfocus |
| PropertyChangeMask | PropertyNotify | xproperty |
| ColormapChangeMask | ColormapNotify | xcolormap |
| OwnerGrabButtonMask | *** | *** |
| (set in GC) | GraphicsExpose | xgraphicsexpose |
| (set in GC) | NoExpose | xnoexpose |
| (always) | SelectionClear | xselectionclear |
| (always) | SelectionNotify | xselection |
| (always) | SelectionRequest | xselectionrequest |
| (always) | MappingNotify | xmapping |
| (always) | ClientMessage | xclient |

Olaylara erişmek için genel çağrı şu şekildedir:

```
NextEvent(display,rep)
```

Sonraki olayı döndürür.

```
XEvent *rep;
```

Kuyruktan kaldırılmış sonraki olayı geri döndürür. Eğer kuyruk boşsa bir olay oluşuncaya kadar kuyruk bekler.

Geri dönmüş olayın, kendi alanlarına erişilmeden önce tanımlanmasına gereksinim duyabileceği unutulmamalıdır. Eğer, mesela, `XEvent` olayını bildirirseniz ve olayın tip alanına bakarak olayın bir `ButtonEvent` olduğunu bulursanız, `event.xbutton.x` yaparak `x` alanına doğru şekilde erişmek için C'nin

birleşim kavramını kullanabilirsiniz.

`XSelectInput(display, w, mask)`

edinilmesi istenen olayları belirten fonksiyon

`int mask;`

olay maskesi: her bir bit 1 olay tipine izin verir

`XFlush(display)`

Bütün çıkış tamponlarını boşaltır.

`XSync(display,discard)`

Çıkış tamponlarını boşaltır ve bütün olayların bitmesini bekler.

`int discard;`

Olayların oluştuğunu bildirir. Eğer discard 0'dan farklı ise bütün olaylardan vazgeçilir.

`XPeekEvent(display,rep)`

Çıkış tamponları boşaltılır,sonra olay ile yapı doldurulur.

`XEvent *rep;`

Sonraki olayı geri döndürür fakat kuyruktan yok olmaz. Eğer kuyruk boş ise bir olay oluşuncaya kadar bekler.

`int XQLength(display)`

Bekleyen olayların bulunduğu kuyruğun uzunluğu bir tamsayı olarak geri döner.

`XPutBackEvent(display, event)`

Bir olayı kuyruğun en üstüne geri koyar.

`XEvent *event;`

Geri konmak istenen olayı belirtir.

`XWindowEvent(display, w, mask, rep)`

Tamponları boşaltıp kuyrukta belli olaylar için arama yapar.

`long mask;`

Bu olay maskesi ile eşleşen olaydır.

`XEvent *rep;`

Arama sırasında bulunan ilk olayı geri döndürür.Eğer uygun olay yoksa bekler.

`int XCheckWindowEvent(display, w, mask, rep)`

Tamponları boşaltıp, maskeye ve pencereye uyan olaylar için kuyrukta arama yapar.

`long mask;`

Bu olay maskesi ile eşleşen olaydır.

`XEvent *rep;`

Arama sırasında bulunan ilk olayı geri döndürür. Eğer uygun olay yoksa 0 geri döndürür.

`XMaskEvent(display, mask, rep)`

Tamponları boşaltıp kuyrukta belli olaylar için arama yapar.

`int mask;`

Bu olay maskesi ile eşleşen olaydır.

`XEvent *rep;`

Arama sırasında bulunan ilk olayı geri döndürür.Eğer uygun olay yoksa bekler.

`XCheckMaskEvent(display, mask, rep)`

Tamponları boşaltıp kuyrukta belli olaylar için arama yapar.

`int mask;`

Bu olay maskesi ile eşleşen olaydır.

`XEvent *rep;`

Arama sırasında bulunan ilk olayı geri döndürür.Eğer uygun olay yoksa 0 geri döndürür.

`int XPending(display)`

Çıkış kuyruklarını boşaltır ve giriş kuyruğundaki hiçbir olayı geri döndürmez.

## 5.2 Klavye Çözümü

Herbir tuş [7,255] aralığında sabit bir tuş kodu üretir. Bir keysyms listesi herbir tuş kodu ile birleştirilir. Keysyms, eğer shift tuşlara vb. basılmış ise gözönüne alınır. Tanımlanmış KeySyms listesi X11/keysyms.h içindedir. Tuş kodlarından keysyms'ye olan tasvir sunucu-genişlemesine değiştirilebilir. Halihazırdaki tasviri görmek için komut satırından xmodmap-pk kullanılabilir.

Keysym'yi bir katara eşleyen tasvir sadece bir istemci için değiştirilebilir.

XLookUpString, hem bir keysym hem de bir ASCII katarı üreterek bir tuş olayını yorumlamak için kullanılır.

Halihazırdaki tasviri elde etmek için şu yapı kullanılır:

```
KeySym *XGetKeyBoardMapping(display, first_keycode, count,
keysyms_per_keycodex)
int first_keycode;
int count;
```

kontrol edilecek tuş kodlarının sayısı

```
int *keysyms_per_keycode;
```

geri dönüş değeri

Tasviri bütünüyle değiştirmek için şu yapı kullanılır:

```
XChangeKeyBoardMapping(display, first_code, keysyms_per_code,
keysyms, num_codes)
KeySym *keysyms;
```

Tasviri yerel olarak değiştirmek içinse şu yapı kullanılır:

Verilen bir tuş kodu için tuş takımı tasvirini değiştirir.

```
XRebindKeySym(display, keysym, list, mod_count, string,
num_bytes)
KeySym keysym;
```

tekrar bağlanması istenilen tuş kodu

```
KeySym *list;
```

Bunlar genişletilmiş olarak kullanılır.

```
int mod_count;
char *string;
```

tuş kodu ile ilişkilendirilmek istenen katar

```
int num_bytes;
```

katardaki byte sayısı

Tuş kodlarının genişletilmiş olarak kullanıldığını görmek için şu yapıya gereksinim vardır:

```
typedef struct {
int lock;
int shift_a, shift_b;
int control_a, control_b;
int mod1_a, mod1_b;
int mod2_a, mod2_b;
int mod3_a, mod3_b;
int mod4_a, mod4_b;
int mod5_a, mod5_b;
} XModifierKeys;
```

`GetModifierMapping(display, modifier_keys)`

Genişletilmiş tuşları geri döndürür.

```
ModifierKeys *modifier_keys;
```

`SetModifierMapping(display, modifier_keys)`

Genişletilmiş tuşları yerleştirir.

`KeyPressedEvent`lerin bir dizisini veren bir karakter katarını geri döndürür.

```
ModifierKeys *modifier_keys;
int XLookUpString(event, buffer, num_bytes, keysym, status)
XKeyEvent *event;
```

yorumlanacak olayların dizisi

```
char *buffer;
int num_bytes;
```

karakter katarında bulunan byte'ların sayısı

```
XComposeStatus *status;
```

Varsayılan tasvir `/usr/lib/keymap.txt` dosyasında bulunur. Fakat kullanıcı kendi tasvirini `~/.Xkeymap` dosyasında belirleyebilir.

# BÖLÜM 6. RENK KULLANIMI

Renkli bir görüntü üzerindeki X içindeki pencereler her zaman renk hücrelerinin bir kolleksiyonu olan(Örneğin Red, Green ve Blue değerlerinin üçlemesi) birleştirilmiş bir renk tasvirine sahiptir.

Bazı yüksek son görüntülerinde renk tasvirinin içeriklerini bir yöntemden daha fazla şekilde yorumlamak mümkün olabilir. Bu farklı yöntemler 'görünebilirler (Visuals)' olarak bilinirler. Görünebilir mümkün sınıflar şunlardır:

**Direkt Renk:** Bir pixel değeri 3 alana ayrılır ve herbiri Red, Green, Blue dizilerini indekslemek için kullanılır.
**Gerçek Renk:** Renk tasvirinin sadece okumalı, sunucu bağımlı, değerler olarak önceden tanımlanması dışında Direkt Renk gibidir.
**Sanki(Pseudo) Renk:** Red, Green, Blue değerlerini üretmek için pixel değeri renk tasvirine (üçlemenin bir dizisine) indekslenir.
**Sabit Renk:** Sadece okunan Sanki Renk'tir.
**GriÖlçek(GrayScale):** Sanki Renk'in, üçleme içinde R=G=B olduğu zamanki bozulmuş bir durumudur.
**Sabit Gri(StaticGray):** Sadece okunan GriÖlçek'tir.

Eğer varsayılan Görünebiliri bulmak için

```
Visual XDefaultVisual(display, screen)
```

fonksiyonu kullanılırsa çok büyük yanlışlara gidilmez. Varsayılan renk tasvirini elde etmek içinse

```
Colormap XDefaultColorMap(display, screen)
```

fonksiyonu kullanılır. Renk tasvirleri özel bir ekrana göre yereldir ve eğer renk tasvirleri sadece okunan iseler birçok uygulama arasında paylaştırılabilir.

Bugün birçok donanım yapısı tek bir renk tasvirine sahiptir, bu yüzden ilkel renk tasvirleri, uygulamalar arasındaki renk tasviri girişlerinin paylaşımını desteklemek için yazılırlar. Eğer görüntü içinde yeterli renk tasviri kaynakları yoksa bazı pencereler onların gerçek renklerinde görüntülenemez. Eğer isimlendirilen renkler istenirse halihazırda kullanılmış bir renk hücresi ile paylaştırılabilir.

Renkler RedGreenBlue değerleri veya isim ile belirlenebilir. Renk bilgisi aşağıdaki yapıda tutulur:

```
typedef struct {
```

Ekran için renk hücresi tasvirleri kurulmasında renk yapısı kullanılır

```
unsigned long pixel;
```

pixel değeri

```
unsigned long red, green, blue;
```

yoğunluk(0'dan 65535'e)

```
char flags;
```

KırmızıYap(DoRed), YeşilYap(DoGreen), MaviYap(DoBlue)

```
} XColor;
```

### 6.1 Sadece Okunan Renk Hücreleri

Aşağıdaki rutin, isimlendirilmiş renklerdeki RGB bilgisini içeren `/usr/lib/X11/rgb.txt` veri tabanına başvurur.

```
Status XLookupColor(display, cmap, spec, screen_def, exact_def)
char *spec;
```

önem verilmemiş durum

```
XColor *screen_def;
```

Renk tasvirinde gerçekten kullanılmış değerleri geri döndürür.

```
XColor *exact_def;
```

Donanım tarafından desteklenmiş en yakın değeri geri döndürür.

3 yararlı küçük rutin şunlardır:

```
StatusXParseColor(display, colormap, spec, screen_def)
char *spec;
```

spec katarının renk ismi

bir spec katarı bir hash işareti ile başlamalıdır ve o zaman boşluksuz 3,6,9 veya 12 hex dijit içerir. Bunlar renk için istenen RGB değerini vermede sırasıyla 1, 2, 3 veya 4 gruplarına alınır.

```
Color *screen_def
```

Donanım tarafından desteklenen en yakın RGB'yi geri döndürür.

```
XQueryColor(display, cmap, def)
XColor *def;
```

Pixel değerini destekler, RGB'yi geri döndürür.

```
XQueryColors(display, cmap, defs, ncolors)
XColor *defs[ncolors];
```

Pixel değerlerini destekler, RGB'yi geri döndürür.

Renk hücrelerini tahsis etmek için eğer mümkünse, sadece okunan (Read Only) hücreler kullanılmalıdır.

```
Status XAllocColor(display, cmap, screen_def)
XColor *screen_def;
```

RGB değerlerini destekler. Belirlenmiş RGB'ye ve gerçekten kullanılmış RGB değerlerine en yakın pixel değerini geri döndürür.

```
Status XAllocNamedColor(display, cmap, colorname, screen_def,
exact_def)
char *colorname;
```

Renk ismini destekler.

```
XColor *screen_def;
```

Renk tasvirinde gerçekten kullanılmış değerleri geri döndürür.

```
XColor *exact_def;
```

Donanım tarafından desteklenmiş en yakın değeri geri döndürür.

# BÖLÜM 7. X TUTORIAL PROGRAMI

Bu tezin konusu olan X tutorial programı ile herhangi bir X kullanıcısına kolay bir şekilde X'in yapısı ve programlanması hakkında bilgi sunulması amaçlanmıştır.

X tutorial programı, OSF-Motif arayüz programı kullanılarak hazırlanmış olup, sadece tezde sunulan X için değil herhangi bir tutorial programı için genel olarak tasarlanmıştır.

X tutorial programı üç kısımdan oluşmaktadır:

i– Tutorial programının kendisi ve özkaynakları

ii– Tutorial dizini

iii– Yazı, kaynak ve grafik olarak içerikleri

Bu bölümde X tutorial programının yapısı ve işlevi anlatılacaktır. Bu tezin konusu olan X, X11 sürümünü içerdiğinden dolayı buradan itibaren 'X tutorial' yerine 'X11 tutorial' ifadesi kullanılacaktır.

## 7.1 OSF-Motif'e Bakış

Open Software Foundation(OSF) tarafından üretilen OSF-Motif toolkit'i, X Windows tabanlı sistemlerde pekçok standart toolkit mekanizmasını içeren X Toolkit Instrinsics(Xt)'in üzerine kurulmuştur. Xt, 'widget'ler ve 'gadget'ler olarak adlandırılan pekçok kullanıcı arayüzünden oluşan bir kütüphaneyi sağlamaktadır. Bu kütüphane, X penceresi yaratma ve yok etme, renkpaleti, olaylar ve diğer görüntü elemanlarını içerir. Kısaca, X toolkit(Xt) tam bir uygulama geliştirmek için programcının ihtiyaç duyduğu yapıları sunar.

Fakat Xt tarafından desteklenen widgetler, kullanıcı arayüzü için tatmin

edici değillerdir. Görsel ve kavramsal olarak tam bir kullanıcı arayüzü için Xt kütüphanesi OSF-Motif gibi toolkit'lere ihtiyaç duymaktadır. OSF-Motif toolkit'i ayrıca kullanıcı arayüzü için widget yaratma ve kullanma fonksiyonlarını içeren bir kütüphaneyi içinde bulundurmaktadır.

## 7.2 X11 Tutorial Programının Tanıtımı

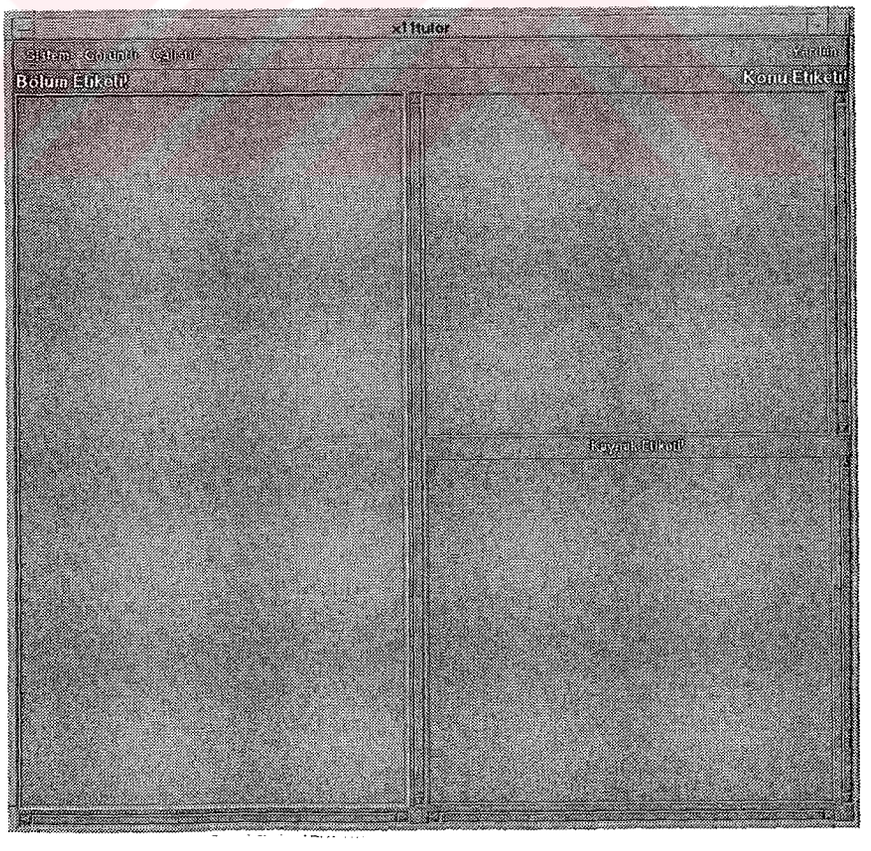
X11 tutorial programı, herhangi bir tutorial için genel olarak yazılmış olup iki kısımda incelenebilir:

i– Kullanıcı Arayüzü

ii– Kaynak Kodu

### 7.2.1 X11 Tutorial Porgramının Kullanıcı Arayüzü

X11 tutorial programı, sonuçta kullanıcıya bir sunum yapmak için yazılmış olduğundan programın en önemli yerini kullanıcı arayüzü oluşturmaktadır. Kullanıcı arayüzü dört ana kısımdan oluşmuştur (Bkz. Şekil 7.1):




Şekil 7.1 x11tutor programının widgetleri

i- Menü

ii- Konunun görüntülendiği kısım

iii- Grafiğin görüntülendiği kısım

iv- Kaynak kodunun görüntülendiği kısım

#### 7.2.1.1 X11 Tutorial Programının Menüsü

X11 tutorial programının menüsü dört ana menüden oluşmaktadır:

i- Sistem

ii- Görüntü

iii- Çalıştır

iv- Yardım

Sistem menü grubunun içinde bulunan menü maddeleri şunlardır:

i- Aç

ii- Çıkış

'Aç' menü maddesi seçildiğinde, X11 tutorial dizin dosyası okunur ve ilk bölümün ilk konusu görüntülenir. 'Çıkış' menü maddesi seçildiğinde ise X11 tutorial programından çıkılır.

Görüntü menü grubunda şu menü maddeleri bulunur:

i- Sonraki

ii- Önceki

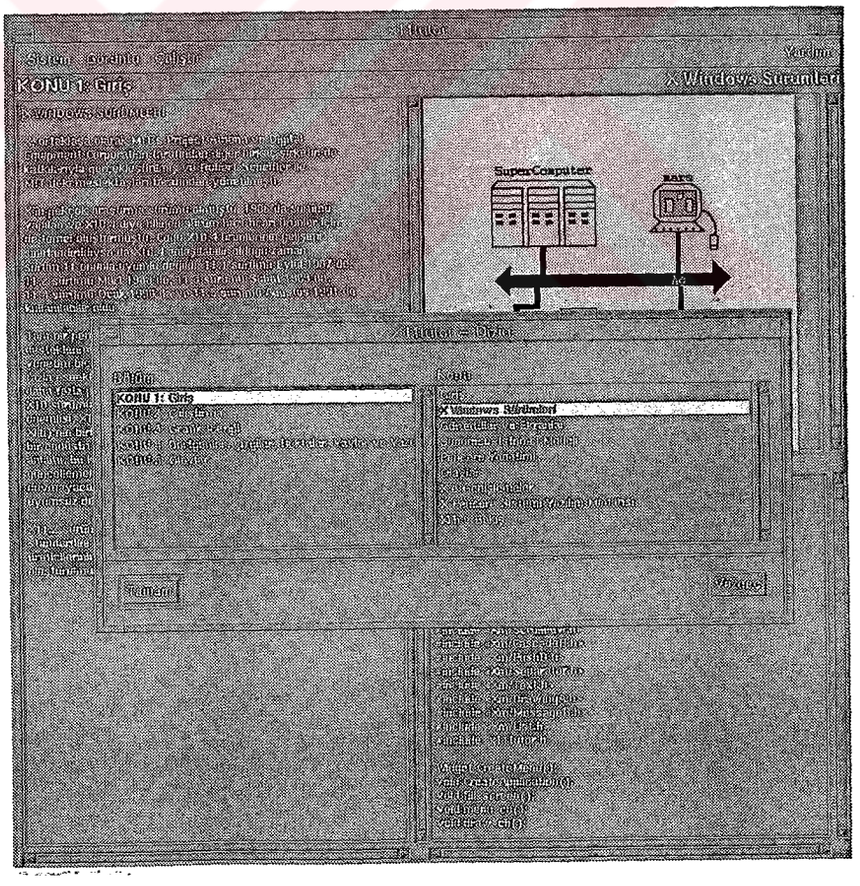
iii- Dizin

'Sonraki' menü maddesi seçildiğinde, aynı bölümdeki bir sonraki konu seçilerek görüntülenir. Eğer o anki konu, ait olduğu bölümün son konusu ise bir sonraki bölümün ilk konusu seçilerek görüntülenir. Fakat o anki bölüm tutorial'a ait son bölüm ve o anki konu o bölüme ait son konu ise 'Sonraki' menü maddesi pasif olacağından zaten seçilemeyecektir.

'Önceki' menü maddesi, 'Sonraki' menü maddesi gibi çalışmakta olup farklı olan yanı bir sonraki konunun değil bir önceki konunun seçilip görüntülen-

mesidir. Eğer o anki bölüm tutorial'a ait ilk bölüm ve o anki konu o bölüme ait ilk konu ise 'Önceki' menü maddesi pasif olacağından zaten seçilemeyecektir.

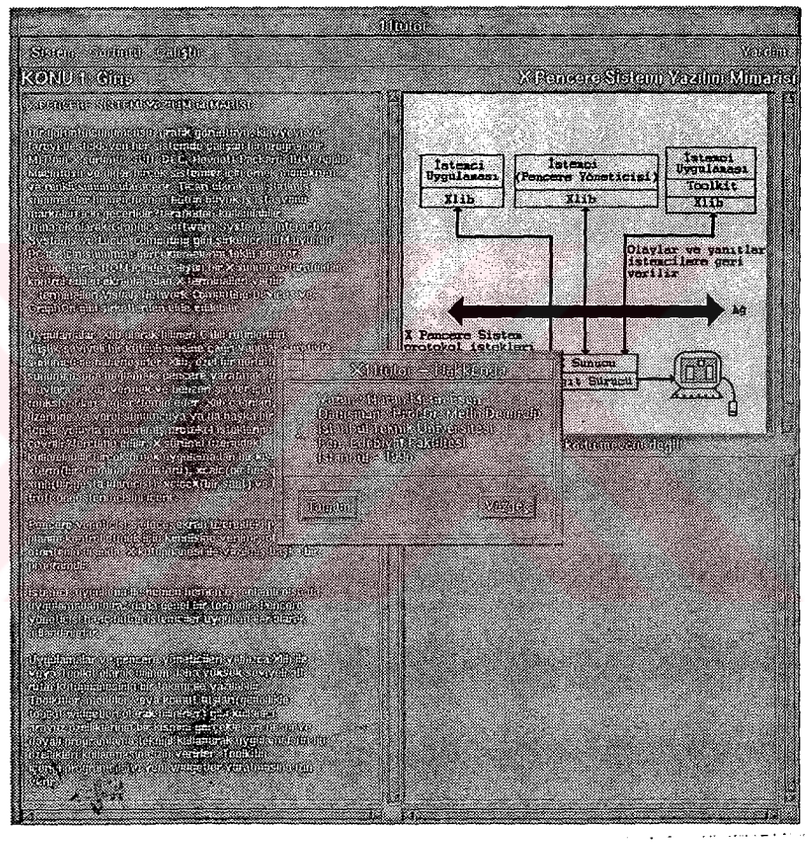
'Dizin' menü maddesi seçilmiş ise ekrana, içinde iki liste yöneticisi olan bir pencere çıkar (Bkz. Şekil 7.2). Dizin penceresi içinde bulunan ilk liste yöneticisinde bölüm başlıkları, ikinci liste yöneticisinde ise konu başlıkları bulunmaktadır. İlk liste yöneticisinde bir bölüm seçildiğinde, o bölüm ile ilgili olan konular ikinci liste yöneticisinde görüntülenir ve ilk konu otomatik olarak seçilir. İkinci liste yöneticisinde ise istenilen konu kullanıcı tarafından seçilebilir. Dizin penceresinin dibinde bulunan 'Tamam' tuşuna basıldığında, birinci ve ikinci liste yöneticisinden seçilmiş olan bölüm ve o bölüme ait konu ekranda görüntülenir. Dizin penceresinin dibinde bulunan bir diğer tuş da 'Vazgeç' tuşudur. Eğer 'Vazgeç' tuşuna basılmış ise dizinden bölüm ve/veya konu seçiminden vazgeçilmiş olur ve dizin penceresi kapatılır.




Şekil 7.2 x11tutor - Dizin penceresi

Çalıştır menüsü seçildiğinde o anki konu ile ilgili kaynak kodunun derlenmiş hali çalıştırılır. Çalıştır menüsünün seçilebilmesi için o anki konunun kaynak kodunun mevcut olması gerekmektedir. Aksi takdirde Çalıştır menüsü pasif halde olacaktır.

Yardım menü grubu içinde, 'Hakkında' isimli tek bir menü maddesi bulunmakta olup bu madde seçildiğinde program ve yazarı hakkında kısa bir bilgi veren bir pencere açılmaktadır (Bkz. Şekil 7.3).



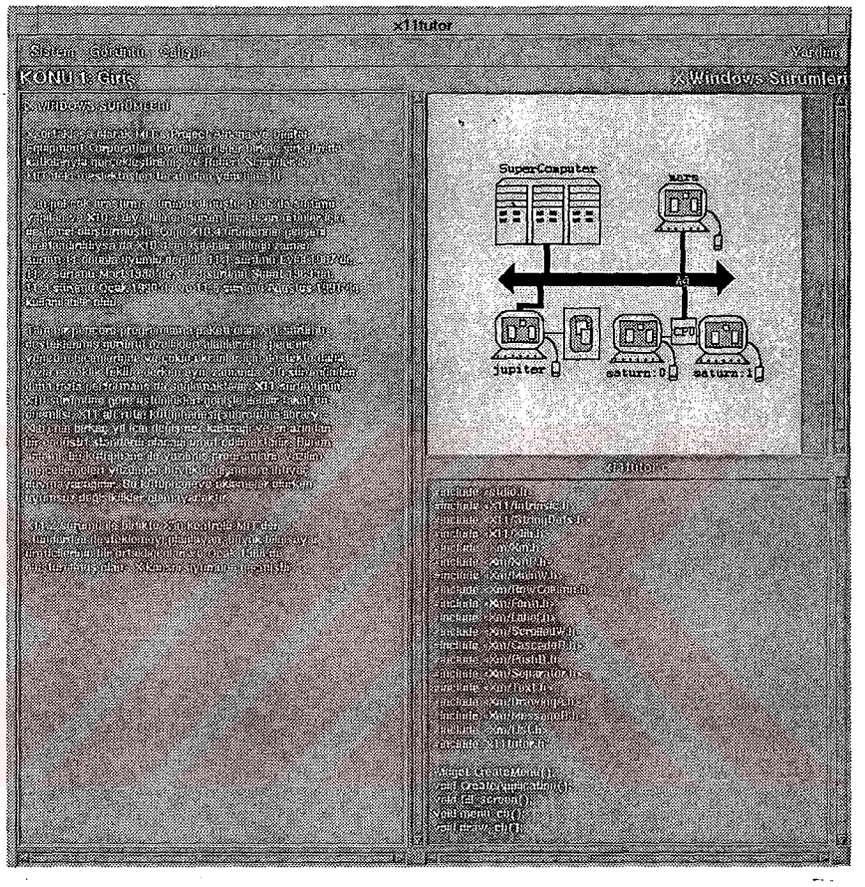

Şekil 7.3 x11tutor - Hakkında penceresi

### 7.2.1.2 Konu Görüntülenmesi

Bir bölüm ve konu seçildiğinde, ilgili bölümün başlığı menünün altında bir etiket olarak ve sola yanaşık durumda basılmaktadır. Konu başlığı ise bölüm başlığı ile dikey olarak aynı hizada fakat sağa yanaşık olarak basılmaktadır. Konu ile ilgili olarak dizinde belirtilen metin dosyası çekilir ve 'text widget'inin içine yüklenerek ekranın sol kısmında görüntülenir (Bkz. Şekil 7.4).

### 7.2.1.3 Grafiğin Görüntülenmesi

Seçilen bir konunun grafiği varsa ilgili grafik dosyası okunur ve sağ üst köşedeki yerinde görüntülenir. Grafikler PCX formatında olup siyah-beyaz, 16 renk veya 256 renk çözünürlülüğünde ekranda görüntülenebilmektedir.




Şekil 7.4 x11tutor programından bir görüntü

### 7.2.1.4 Kaynak Kodunun Görüntülenmesi

Seçilen bir konunun kaynak kodu mevcutsa, kaynak kodunun içeriği ilgili dosyadan okunarak pencerenin sağ alt kısmında bulunan 'text widget'ine yüklenir ve dosya ismi 'text widget'inin tepesindeki yerde ortalanmış olarak bir etiket şeklinde bastırılır.

### 7.2.1.5 X11 Tutorial Programının Özkaynağı

X11 ve Motif kullanıcı arayüzü programlarının bir özelliği de kullanıcı

arayüzü widgetlerinin kaynaklarını sadece kaynak kodunda değil bir özkaynak dosyası içinde de tutabilmesidir. Bu ise yazılan bir arayüz programında kullanılan widgetlerin özelliklerinin, tekrar kaynak kodu değiştirilmesine gerek kalmadan, sadece bir yazı editöründen değiştirilerek kullanılmasına olanak sağlar. X11 tutorial özkaynak dosyası Ek'te X11TUTOR ismi ile görülebilir.

### 7.2.2 X11 Tutorial Programının Kaynak Kodu

X11 tutorial programı bir başlık(header) ve iki kaynak kodundan oluşmaktadır:

i- `x11tutor.h`

ii- `x11tutor.c`

iii- `tutorio.c`

#### 7.2.2.1 `x11tutor.h` Başlık Kodu

`x11tutor.h` başlık kodu içinde `CHAPTER` ve `PCXHEADER` tipleri tanımlanmıştır.

`CHAPTER` tipinin özellikleri şunlardır:

`char Title[64];`

bölüm başlığı

`int Section;`

bölümdeki konu sayısı

`char SectionTitle[10][64];`

konu başlığı

`char SectionFileName[10][64];`

konunun bulunduğu dosya ismi

`char GraphicFileName[10][64];`

konu grafiğinin bulunduğu dosya ismi

`char SourceFileName[10][64];`

konu kaynak kodunun bulunduğu dosya ismi

Ayrıca `ChapterN` ile bölüm sayısı, `CurChapter` ile o anki bölüm numarası,

CurSection ile de o anki konu numarası tutulmaktadır. Chapter değişkeni ise CHAPTER tipinde bir diziyi gösterecek şekilde tanımlanmıştır.

PCXHEADER ise PCX dosyalarının yüklenmesi ile ilgili olup yapısı şu şekildedir:

```
unsigned char identifier, version, encoding, bits_per_pixel;
short int left, top, right, bottom, hor_res, ver_res;
unsigned char reserved, planes;
short int bytes_per_line, palette_infor;
unsigned char colormap_signatur;
```

Ayrıca PCX grafikleri için şu değişkenler tutulmaktadır:

```
PCXHEADER pcx_header;
short int pcx_columns, pcx_rows, pcx_colors;
char *pcx_colormap, *pcx_pixels;
unsigned short *image_pixels;
Boolean is_pcx_loaded;
```

#### 7.2.2.2 x11tutor.c Kaynak Kodu

x11tutor.c kaynak kodu, kullanıcı arayüzünün yaratıldığı ve bu arayüzde görüntülemenin yapıldığı, kullanıcının program ile etkileşimi sonucunda oluşan olayları değerlendirip yönetimin sağlandığı kaynak kodudur.

main() prosedüründe Motif içinde uygulamanın yaratılması ve ana döngünün başlatılması yer almaktadır.

CreateMenu() prosedüründe menü widgetleri yaratılmaktadır.

CreateApplication() prosedüründe ise bölüm ve konu başlık etiketleri ile konu içeriği, grafik ve kaynak kodunun görüntülendiği widgetler yaratılmaktadır. Ayrıca bütün bu widgetlerin pencere içinde nasıl dizileceği belirlenmektedir (Form widgeti kullanılarak dizilme işlemi sağlanır).

fill_screen() prosedürü ile CurChapter ve CurSection ile belirlenmiş olan ilgili bölümün ilgili konusuna Chapter dizininden erişilerek ilgili konu,

grafik ve kaynak kodunun yardımcı fonksiyonlar (`load_text`, `load_pcx` ve `show_pcx`) ile pencerede görüntülenmesi sağlanır.

`menu_cb()` prosedürü ile menü seçilmesi ile oluşan çağrılar değerlendirilir.

`draw_cb()` prosedürü ile grafik çizimi için oluşan çağrılar değerlendirilir.

`show_pcx()` prosedürü ile PCX dosya formatındaki grafik, `draw_pic` çizim widgetine yorumlanarak çizilir.

`call_index()` prosedürü ise görüntü menüsündeki dizin menü maddesi seçildiğinde ortaya çıkacak olan dizin penceresini oluşturur.

`print_current_chapter()` prosedürü, dizin penceresindeki `index_list_chp` liste yöneticisinin içini bölüm başlıkları ile doldurup, `CurChapter` ile belirlenmiş olan o anki bölüm başlığını seçer.

print_current_section() prosedürü, dizin penceresindeki `index_list_sec` liste yöneticisinin içini o anki bölümün konu başlıkları ile doldurup, `CurSection` ile belirlenmiş olan o anki konu başlığını seçer. Fakat seçilmiş bölüm ile o anki bölüm birbirini tutmuyor ise ilk konu başlığı seçilir.

`list_cb()` prosedürü içinde ise bölüm ve konu listesinden seçim yapıldığında ilgili işlemler yapılır.

`index_cb()` prosedürü ise dizin penceresinde 'Tamam' tuşuna basıldığında listelerde seçilmiş olan bölüm ve konuları `CurChapter` ve `CurSection` değişkenlerine atayarak `fill_screen` prosedürünü çağırır.

#### 7.2.2.3 `tutorio.c` Kaynak Kodu

x11tutor programının dosya işlemlerinin yapıldığı kaynak kodudur. İçinde şu prosedürler bulunur:

`load_text()` prosedürü, ismi verilmiş dosyayı okur ve kendisine gönderilmiş olan tampona doldurarak geri gönderir.

`read_index()` prosedürü, X11.tut dosyasını açarak içindeki yapıyı `Chapter` dizinine yerleştirir.

`load_pcx()` prosedürü, ismi verilmiş PCX grafik formatındaki dosyayı okuyarak ilgili değişkenleri doldurur.

### 7.3 X11 Tutorial Programının Dizini

X11 tutorial programı genel amaçlı yazıldığından herhangi bir tutorial'i çalıştırması için bu tutorial'in dizininin bir dizin dosyasında tutulmasına gereksinim vardır. Bu dizin dosyasının yapısı şöyledir:
CHAPTER NUMBER: bölüm_sayısı

CHAPTER TITLE: bölüm_başlığı
SECTION NUMBER: konu_sayısı

SECTION TITLE: konu başlığı
SECTION FILE NAME: konu_dosya_ismi
GRAPHIC FILE NAME: grafik_dosya_ismi
SOURCE FILE NAME: kaynak_kodu_dosya_ismi

Dizinde bölüm_sayısı kadar bölüm ve her bir bölümde konu_sayısı kadar konu olmalıdır. Sıralama ise her bölümün bütün konuları sıralı olarak verildikten sonra bir sonraki bölümün başlaması şeklindedir. Herhangi bir satırın başında bir veya daha fazla boşluk bulunması halinde ise satır değerlendirilmeden bir sonraki satıra geçilir. Ayrıca ilgili konunun metin dosyası, grafik dosyası ve/veya kaynak kodu dosyası yok ise 'NONE' yazılarak belirtilir (Bkz. EK X11.TUT).

### 7.4 Konu, Grafik ve Kaynak Kodunun İçerikleri

Konu, bir metin editöründe yazılmış, konunun içeriğini oluşturan herhangi bir metin olabilir.

Grafik ise PCX dosya formatında olup, 400x400 pixel sayısını geçmeyecek bir şekilde 2, 16 veya 256 renk ayırıcılığını içeren bir grafik dosyası olabilir.

Kaynak kodu ise konu ile ilgili herhangi bir kaynak kodunu içermekte olup ayrıca derlenmiş hali de 'Çalıştır' menü tuşu seçildiğinde çalışabilmesi için hazır bulundurulmalıdır [5], [6], [7], [8].

# SONUÇ

Bu projenin amacı olan X Windows tutorial programı, OSF-Motif arayüz programı kullanılarak gerçekleştirilmiş ve herhangi bir başka konuda hazırlanacak tutorial programına temel oluşturması düşünülerek genel olarak tasarlanmıştır. Bu açıdan düşünülürse, OSF-Motif pencere yöneticisi için de, tutorial programının temel yapısı bozulmadan fakat içeriği değiştirilerek (Motif ile), bir Motif tutorial programı hazırlanabilir veya X Windows tutorial programı geliştirilebilir. Bu projede bir X kullanıcısının, X'in yapısı ve programlanması hakkında gereksinim duyabileceği konular tez kapsamına alınmış ve incelenmiştir. Diğer taraftan, 'başka istemciler ve pencere yöneticileri ile işbirliği' ve 'istemciler arası haberleşme' konularının incelenmesi, eksikliği hissedilen bir boşluğu dolduracaktır. Ayrıca tutorial programındaki 'Yardım' seçeneğine bağlı olarak bir konu arama penceresinin eklenmesi de düşünülürse bu projenin daha kapsamlı olarak geliştirilmesi mümkün olacaktır.

## KAYNAKLAR


[1] NYE A., "Xlib Programming Manual", O'Reilly & Associates Inc., A.B.D., (1992)

[2] NYE A., "Xlib Reference Manual", O'Reilly & Associates Inc., A.B.D., (1992)

[3] MUI L., PEARCE E., "X Window System Administrator's Guide", O'Reilly & Associates Inc., A.B.D., (1993)

[4] LOVE T., "X Windows Version 11.4 A Concise Description", Cambridge, (1991)

[5] HALLE D., FERGUSON P. M., "Motif Programming Manual", O'Reilly & Associates Inc., A.B.D., (1993)

[6] FERGUSON P. M., "Motif Reference Manual", O'Reilly & Associates Inc., A.B.D., (1993)

[7] KOBARA S., "Visual Design with OSF/Motif", Addison Wesley, Massachusetts, (1991)

[8] SCHILDT H., "Teach Yourself C", Osborne McGraw-Hill, New York, (1990)

# EKLER.

## X TURORIAL PROGRAMININ KAYNAK KODU

X11TUTOR.H:

```
#ifndef _X11TUTOR_H_INCLUDED

typedef struct {
  char Title[64];
  int SectionN;
  char SectionTitle[10][64];
  char SectionFileName[10][64];
  char GraphicFileName[10][64];
  char SourceFileName[10][64];
} CHAPTER;

typedef struct {
  unsigned char identifier, version, encoding, bits_per_pixel;
  short int left, top, right, bottom,
            horizontal_resolution, vertical_resolution;
  unsigned char reserved, planes;
  short int bytes_per_line, palette_info;
  unsigned char colormap_signature;
} PCXHEADER;

extern Boolean load_text(char *, String *);
extern Boolean read_index();
extern Boolean load_pcx(char *);
extern void destroy_pcx();

extern CHAPTER Chapter[];
extern int ChapterN, CurChapter, CurSection;
extern PCXHEADER pcx_header;
extern short int pcx_columns, pcx_rows, pcx_colors;
extern char *pcx_colormap, *pcx_pixels;
extern unsigned short *image_pixels;
extern Boolean is_pcx_loaded;

#define _X11TUTOR_H_INCLUDED 1
#endif
```

X11TUTOR.C:

```
#include <stdio.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <X11/Xlib.h>
#include <Xm/Xm.h>
#include <Xm/XmP.h>
#include <Xm/MainW.h>
#include <Xm/RowColumn.h>
#include <Xm/Form.h>
#include <Xm/Label.h>
#include <Xm/ScrolledW.h>
#include <Xm/CascadeB.h>
#include <Xm/PushB.h>
#include <Xm/Separator.h>
#include <Xm/Text.h>
#include <Xm/DrawingA.h>
#include <Xm/MessageB.h>
#include <Xm/List.h>
#include "x11tutor.h"

Widget CreateMenu();
void CreateApplication();
void fill_screen();
void menu_cb();
void draw_cb();
void show_pcx();
void call_index();
void print_current_chapter();
void print_current_section(int);
void list_cb();
void index_cb();
void call_about_dialog();

Display *pcx_display;
Window pcx_window;
int pcx_screen;
Pixmap pcx_pixmap;
GC pcx_gc[256];
Boolean pixmap_first_time=True;
int selected_chapter, selected_section;

Widget toplevel, mainwindow, text_sec, draw_pic, text_src,
       pbutton_prev, pbutton_next, pbutton_idx,
       label_chpt, label_sec, label_src, index_list_chpt,
       index_list_sec;
XtAppContext app_con;
```

```
Arg args[10];
Cardinal nargs;

void main(argc, argv)
 int argc;
 char *argv[];
{
  toplevel = XtAppInitialize(&app_con, "X11tutor", NULL, 0,
                             &argc, argv, NULL, NULL, 0);
  CreateApplication();
  XtRealizeWidget(toplevel);
  XtAppMainLoop(app_con);
}

Widget CreateMenu()
{
  Widget menu_bar, cbutton_one, cbutton_two, cbutton_three,
         pdmenu_one, pdmenu_two, pbutton_open, pbutton_exit,
         separator, pdmenu_help, cbutton_help, pbutton_help;

  menu_bar = XmCreateMenuBar(mainwindow, "menu_bar", NULL, 0);
  XtManageChild(menu_bar);

  pdmenu_one = XmCreatePulldownMenu(menu_bar, "pdmenu_one",
                                    NULL, 0);
  pdmenu_two = XmCreatePulldownMenu(menu_bar, "pdmenu_two",
                                    NULL, 0);

  nargs = 0;
  XtSetArg (args[nargs], XmNsubMenuId, pdmenu_one);
  nargs++;
  cbutton_one = XmCreateCascadeButton(menu_bar, "cbutton_one",
                                      args, nargs);
  XtManageChild(cbutton_one);

  nargs = 0;
  XtSetArg (args[nargs], XmNsubMenuId, pdmenu_two);
  nargs++;
  cbutton_two = XmCreateCascadeButton(menu_bar, "cbutton_two",
                                      args, nargs);
  XtManageChild(cbutton_two);

  nargs = 0;
  cbutton_three=XmCreateCascadeButton(menu_bar, "cbutton_three",
                                      args, nargs);
  XtManageChild(cbutton_three);
  XtAddCallback(cbutton_three, XmNactivateCallback, menu_cb,
```

```
                (XtPointer) 10);

    pbutton_open = XmCreatePushButton(pdmenu_one, "pbutton_open",
                                      NULL, 0);
    XtManageChild(pbutton_open);
    separator = XmCreateSeparator(pdmenu_one, "separator",
                                  NULL, 0);
    XtManageChild(separator);
    pbutton_exit = XmCreatePushButton(pdmenu_one, "pbutton_exit",
                                      NULL, 0);
    XtManageChild(pbutton_exit);
    pbutton_prev = XmCreatePushButton(pdmenu_two, "pbutton_prev",
                                      NULL, 0);
    XtManageChild(pbutton_prev);
    pbutton_next = XmCreatePushButton(pdmenu_two, "pbutton_next",
                                      NULL, 0);
    XtManageChild(pbutton_next);
    separator = XmCreateSeparator(pdmenu_two, "separator",
                                  NULL, 0);
    XtManageChild(separator);
    pbutton_idx = XmCreatePushButton(pdmenu_two, "pbutton_idx",
                                     NULL, 0);
    XtManageChild(pbutton_idx);

    pdmenu_help = XmCreatePulldownMenu(menu_bar, "pdmenu_help",
                                       NULL, 0);

    nargs = 0;
    XtSetArg(args[nargs], XmNsubMenuId, pdmenu_help);
    nargs++;
    cbutton_help = XmCreateCascadeButton(menu_bar, "cbutton_help",
                                         args, nargs);
    XtManageChild(cbutton_help);

    nargs = 0;
    XtSetArg(args[nargs], XmNmenuHelpWidget,
             (XtArgVal) cbutton_help);
    nargs++;
    XtSetValues(menu_bar, args, nargs);

    XtAddCallback(pbutton_open, XmNactivateCallback, menu_cb,
                  (XtPointer) 1);
    XtAddCallback(pbutton_exit, XmNactivateCallback, menu_cb,
                  (XtPointer) 2);
    XtAddCallback(pbutton_prev, XmNactivateCallback, menu_cb,
                  (XtPointer) 5);
    XtAddCallback(pbutton_next, XmNactivateCallback, menu_cb,
```

```
                (XtPointer) 6);
  XtAddCallback(pbutton_idx, XmNactivateCallback, menu_cb,
                (XtPointer) 7);

  pbutton_help = XmCreatePushButton(pdmenu_help, "pbutton_help",
                                    NULL, 0);
  XtManageChild(pbutton_help);

  XtAddCallback(pbutton_help, XmNactivateCallback, menu_cb,
                (XtPointer) 20);

  return(menu_bar);
}

void CreateApplication()
{
  Widget form, form_sec, form_pic, form_src, scroll_win,
         menu_bar;

  mainwindow = XmCreateMainWindow(toplevel, "mainwindow",
                                  NULL, 0);
  XtManageChild(mainwindow);

  menu_bar = CreateMenu();

  form = XmCreateForm(mainwindow, "form", NULL, 0);
  XtManageChild(form);

  nargs = 0;
  XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_FORM);
  nargs++;
  XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
  nargs++;
  XtSetArg(args[nargs], XmNbottomAttachment, XmATTACH_FORM);
  nargs++;
  XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_POSITION);
  nargs++;
  XtSetArg(args[nargs], XmNrightPosition, 49);
  nargs++;
  form_sec = XmCreateForm(form, "form_sec", args, nargs);
  XtManageChild(form_sec);

  nargs = 0;
  XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_FORM);
  nargs++;
  XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
  nargs++;
```

```
XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_WIDGET);
nargs++;
XtSetArg(args[nargs], XmNleftWidget, form_sec);
nargs++;
XtSetArg(args[nargs], XmNbottomAttachment, XmATTACH_POSITION);
nargs++;
XtSetArg(args[nargs], XmNbottomPosition, 49);
nargs++;
form_pic = XmCreateForm(form, "form_pic", args, nargs);
XtManageChild(form_pic);

nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_WIDGET);
nargs++;
XtSetArg(args[nargs], XmNtopWidget, form_pic);
nargs++;
XtSetArg(args[nargs], XmNleftAttachment,
         XmATTACH_OPPOSITE_WIDGET);
nargs++;
XtSetArg(args[nargs], XmNleftWidget, form_pic);
nargs++;
XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNbottomAttachment, XmATTACH_FORM);
nargs++;
form_src = XmCreateForm(form, "form_src", args, nargs);
XtManageChild(form_src);

nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
nargs++;
label_chpt = XmCreateLabel(form_sec, "label_chpt",
                           args, nargs);
XtManageChild(label_chpt);

nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
nargs++;
label_sec = XmCreateLabel(form_pic, "label_sec", args, nargs);
```

```
XtManageChild(label_sec);

nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
nargs++;
label_src = XmCreateLabel(form_src, "label_src", args, nargs);
XtManageChild(label_src);

nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_WIDGET);
nargs++;
XtSetArg(args[nargs], XmNtopWidget, label_chpt);
nargs++;
XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNbottomAttachment, XmATTACH_FORM);
nargs++;
text_sec = XmCreateScrolledText(form_sec, "text_sec",
                                args, nargs);
XtManageChild(text_sec);

nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_WIDGET);
nargs++;
XtSetArg(args[nargs], XmNtopWidget, label_src);
nargs++;
XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNbottomAttachment, XmATTACH_FORM);
nargs++;
text_src = XmCreateScrolledText(form_src, "text_src",
                                args, nargs);
XtManageChild(text_src);

nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_WIDGET);
nargs++;
XtSetArg(args[nargs], XmNtopWidget, label_sec);
nargs++;
```

```
  XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
  nargs++;
  XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
  nargs++;
  XtSetArg(args[nargs], XmNbottomAttachment, XmATTACH_FORM);
  nargs++;
  scroll_win = XmCreateScrolledWindow(form_pic, "scroll_win",
                                      args, nargs);
  XtManageChild(scroll_win);

  draw_pic = XmCreateDrawingArea(scroll_win, "draw_pic",
                                 NULL, 0);
  XtManageChild(draw_pic);
  XtAddCallback(draw_pic, XmNexposeCallback, draw_cb,
                (XtPointer) 1);

  XmMainWindowSetAreas(mainwindow, menu_bar, NULL, NULL,
                       NULL, form);
}

void fill_screen()
{
  char *resourcebuffer = NULL, strlabel[64];
  XmString xmstr;

/* Bolum penceresinin icinin doldurulmasi */
  if(load_text(Chapter[CurChapter].SectionFileName[CurSection],
     &resourcebuffer) == False) return;
  XmTextSetString(text_sec, resourcebuffer);
  XtFree(resourcebuffer);
/*  Bolum etiketinin degistirilmesi */
  xmstr = XmStringCreateLocalized(Chapter[CurChapter].Title);
  nargs = 0;
  XtSetArg(args[nargs], XmNlabelString, (XtArgVal) xmstr);
  nargs++;
  XtSetValues(label_chpt, args, nargs);
  XmStringFree(xmstr);
/* Konu etiketinin deitirilmesi */
  xmstr = XmStringCreateLocalized(
                  Chapter[CurChapter].SectionTitle[CurSection]);
  nargs = 0;
  XtSetArg(args[nargs], XmNlabelString, (XtArgVal) xmstr);
  nargs++;
  XtSetValues(label_sec, args, nargs);
  XmStringFree(xmstr);

/* Kaynak penceresinin icinin doldurulmasi */
```

```
  if(strcmp(Chapter[CurChapter].SourceFileName[CurSection],
            "NONE") == 0) {
    resourcebuffer = (char *) malloc(1);
    resourcebuffer[0] = 0;
    strcpy(strlabel, "Kaynak kodu mevcut degil!");
  } else {
    if(load_text(Chapter[CurChapter].SourceFileName[CurSection],
       &resourcebuffer) == False) return;
    strcpy(strlabel,
           Chapter[CurChapter].SourceFileName[CurSection]);
  }
  XmTextSetString(text_src, resourcebuffer);
  XtFree(resourcebuffer);
/* Kaynak etiketinin degistirilmesi */
  xmstr = XmStringCreateLocalized(strlabel);
  nargs = 0;
  XtSetArg(args[nargs], XmNlabelString, (XtArgVal) xmstr);
  nargs++;
  XtSetValues(label_src, args, nargs);
  XmStringFree(xmstr);

/* PCX Grafik penceresinin icinin doldurulmasi */
  if (is_pcx_loaded) destroy_pcx();
  if(strcmp(Chapter[CurChapter].GraphicFileName[CurSection],
            "NONE") == 0) {
    XClearWindow(pcx_display, pcx_window);
    return;
  }
  if(load_pcx(Chapter[CurChapter].GraphicFileName[CurSection])
      == False) {
    XClearWindow(pcx_display, pcx_window);
    return;
  }
  show_pcx(True);
}

void menu_cb(w, client_data, call_data)
 Widget w;
 XtPointer client_data, call_data;
{
  switch ((int) client_data) {
    case  1:
      if(read_index() == True) {
/* Bir Widget'in argumanlarini degistirme */
        nargs = 0;
        XtSetArg(args[nargs], XmNsensitive, (XtArgVal) True);
        nargs++;
```

```
      XtSetValues(pbutton_next, args, nargs);
      nargs = 0;
      XtSetArg(args[nargs], XmNsensitive, (XtArgVal) True);
      nargs++;
      XtSetValues(pbutton_idx, args, nargs);
      fill_screen();
    }
    break;
  case  2: exit(0);
  case  5:
    if(CurChapter == 0 && CurSection == 0) {
      nargs = 0;
      XtSetArg(args[nargs], XmNsensitive, (XtArgVal) False);
      nargs++;
      XtSetValues(pbutton_prev, args, nargs);
      break;
    }
    if(CurSection == 0) {
      CurChapter--;
      CurSection = Chapter[CurChapter].SectionN-1;
    } else CurSection--;
    nargs = 0;
    XtSetArg(args[nargs], XmNsensitive, (XtArgVal) True);
    nargs++;
    XtSetValues(pbutton_next, args, nargs);
    fill_screen();
    break;
  case  6:
    if(CurChapter == ChapterN-1 &&
       CurSection == Chapter[CurChapter].SectionN-1) {
      nargs = 0;
      XtSetArg(args[nargs], XmNsensitive, (XtArgVal) False);
      nargs++;
      XtSetValues(pbutton_next, args, nargs);
      break;
    }
    if(CurSection == Chapter[CurChapter].SectionN-1) {
      CurChapter++;
      CurSection = 0;
    } else CurSection++;
    nargs = 0;
    XtSetArg(args[nargs], XmNsensitive, (XtArgVal) True);
    nargs++;
    XtSetValues(pbutton_prev, args, nargs);
    fill_screen();
    break;
  case  7: call_index(); break;
```

```
    case 10: puts("run"); break;
    case 20: call_about_dialog(); break;
  }
}

void draw_cb(w, client_data, call_data)
 Widget w;
 XtPointer client_data, call_data;
{
  XmDrawingAreaCallbackStruct *dacs =
    (XmDrawingAreaCallbackStruct *) call_data;

  switch(dacs->reason) {
    case XmCR_EXPOSE: show_pcx(False); break;
  }
}

void show_pcx(pcx_first_time)
 Boolean pcx_first_time;
{
  XColor gc_color, dummy;
  XGCValues gc_values;
  register int x, y;
  register unsigned short *q;

  if (pixmap_first_time) {
    int i;
    short int pcx_width, pcx_height;

    pixmap_first_time = False;
    pcx_display = XtDisplay(draw_pic);
    pcx_screen = DefaultScreen(pcx_display);
    pcx_window = XtWindow(draw_pic);
    nargs = 0;
    XtSetArg(args[nargs], XmNwidth, &pcx_width); nargs++;
    XtSetArg(args[nargs], XmNheight, &pcx_height); nargs++;
    XtGetValues(draw_pic, args, nargs);
    pcx_pixmap = XCreatePixmap(pcx_display, pcx_window,
                               pcx_width, pcx_height,
         XDefaultDepth(pcx_display, pcx_screen));
    XAllocNamedColor(pcx_display, DefaultColormap(pcx_display,
                     pcx_screen), "black", &gc_color, &dummy);
    gc_values.foreground=gc_values.background=gc_color.pixel;
    for (i=0; i<256; i++)
      pcx_gc[i] = XCreateGC(pcx_display, pcx_window,
                  GCForeground+GCBackground, &gc_values);
  }
```

```
  if (pcx_first_time) {
    char color_name[8], str[8];
    int i;

    for (i=0; i<pcx_colors; i++) {
      color_name[0] = '#';
      sprintf(str, "%04hx", pcx_colormap[i*3]);
      color_name[1] = str[2];
      color_name[2] = str[3];
      sprintf(str, "%04hx", pcx_colormap[i*3+1]);
      color_name[3] = str[2];
      color_name[4] = str[3];
      sprintf(str, "%04hx", pcx_colormap[i*3+2]);
      color_name[5] = str[2];
      color_name[6] = str[3];
      color_name[7] = 0;
      if (!XParseColor(pcx_display, DefaultColormap(pcx_display,
                       pcx_screen),
         color_name, &gc_color))
          printf("XParseColor Error: %s\n", color_name);
      if (!XAllocColor(pcx_display, DefaultColormap(pcx_display,
                       pcx_screen),
         &gc_color)) printf("XAllocColorError: %d - %s\n", i,
                            color_name);
      gc_values.foreground = gc_color.pixel;
      XChangeGC(pcx_display, pcx_gc[i], GCForeground,
                &gc_values);
      XClearWindow(pcx_display, pcx_window);
    }
  }
  if (is_pcx_loaded) {
    q = image_pixels;
    for (y=0; y<pcx_rows; y++)
     for (x=0; x<pcx_columns; x++)
      XDrawPoint(pcx_display, pcx_pixmap, pcx_gc[*(q++)], x, y);
    XCopyArea(pcx_display, pcx_pixmap, pcx_window, pcx_gc[0],
              0, 0, pcx_columns, pcx_rows, 0, 0);
  }
}

void call_index()
{
  Widget index_dialog, index_button, index_mainwindow,
         index_form, index_form_chapter, index_form_section,
         index_label_chpt, index_label_sec;

  index_dialog = XmCreateMessageDialog(mainwindow,
```

```
                                        "index_dialog", NULL, 0);

    index_button = XmMessageBoxGetChild(index_dialog,
                                        XmDIALOG_HELP_BUTTON);
    XtUnmanageChild(index_button);

    XtAddCallback(index_dialog, XmNokCallback, index_cb,
                  (XtPointer) 1);

    index_mainwindow = XmCreateMainWindow(index_dialog,
                                    "index_mainwindow", NULL, 0);
    XtManageChild(index_mainwindow);

    index_form = XmCreateForm(index_mainwindow, "index_form",
                              NULL, 0);
    XtManageChild(index_form);

    nargs = 0;
    XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_FORM);
    nargs++;
    XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
    nargs++;
    XtSetArg(args[nargs], XmNbottomAttachment, XmATTACH_FORM);
    nargs++;
    XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_POSITION);
    nargs++;
    XtSetArg(args[nargs], XmNrightPosition, 49);
    nargs++;
    index_form_chapter = XmCreateForm(index_form,
                                "index_form_chapter", args, nargs);
    XtManageChild(index_form_chapter);

    nargs = 0;
    XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_FORM);
    nargs++;
    XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
    nargs++;
    XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_WIDGET);
    nargs++;
    XtSetArg(args[nargs], XmNleftWidget, index_form_chapter);
    nargs++;
    XtSetArg(args[nargs], XmNbottomAttachment, XmATTACH_FORM);
    nargs++;
    index_form_section = XmCreateForm(index_form,
                                "index_form_section", args, nargs);
    XtManageChild(index_form_section);
```

```
nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
nargs++;
index_label_chpt = XmCreateLabel(index_form_chapter,
                                 "index_label_chpt", args, nargs);
XtManageChild(index_label_chpt);

nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
nargs++;
index_label_sec = XmCreateLabel(index_form_section,
                                "index_label_sec", args, nargs);
XtManageChild(index_label_sec);

nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_WIDGET);
nargs++;
XtSetArg(args[nargs], XmNtopWidget, index_label_chpt);
nargs++;
XtSetArg(args[nargs], XmNbottomAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
nargs++;
index_list_chpt = XmCreateScrolledList(index_form_chapter,
                                 "index_list_chpt", args, nargs);
XtManageChild(index_list_chpt);

nargs = 0;
XtSetArg(args[nargs], XmNtopAttachment, XmATTACH_WIDGET);
nargs++;
XtSetArg(args[nargs], XmNtopWidget, index_label_sec);
nargs++;
XtSetArg(args[nargs], XmNbottomAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNleftAttachment, XmATTACH_FORM);
nargs++;
XtSetArg(args[nargs], XmNrightAttachment, XmATTACH_FORM);
```

```
  nargs++;
  index_list_sec = XmCreateScrolledList(index_form_section,
                                  "index_list_sec", args, nargs);
  XtManageChild(index_list_sec);

  XtAddCallback(index_list_chpt, XmNbrowseSelectionCallback,
                list_cb, (XtPointer)1);
  XtAddCallback(index_list_sec, XmNbrowseSelectionCallback,
                list_cb, (XtPointer)2);

  XmMainWindowSetAreas(index_mainwindow, NULL, NULL, NULL,
                       NULL, index_form);

  print_current_chapter();
  selected_chapter = CurChapter;
  selected_section = CurSection;
  XtManageChild(index_dialog);
}

void print_current_chapter()
{
  XmString *xmstr;
  int i;

  xmstr = (XmString *) malloc(ChapterN*sizeof(XmString));
  for(i=0; i<ChapterN; i++)
    xmstr[i] = XmStringCreateLocalized(Chapter[i].Title);
  XmListAddItems(index_list_chpt, xmstr, ChapterN, 0);
  for(i=0; i<ChapterN; i++)
    XmStringFree(xmstr[i]);
  free(xmstr);
  XmListSelectPos(index_list_chpt, CurChapter+1, False);
  print_current_section(CurChapter);
}

void print_current_section(parm_chpt_no)
 int parm_chpt_no;
{
  XmString *xmstr;
  int i;

  XmListDeleteAllItems(index_list_sec);
  xmstr = (XmString *) malloc(Chapter[parm_chpt_no].SectionN*
                              sizeof(XmString));
  for(i=0; i<Chapter[parm_chpt_no].SectionN; i++)
   xmstr[i] = XmStringCreateLocalized(
                        Chapter[parm_chpt_no].SectionTitle[i]);
```

```
  XmListAddItems(index_list_sec, xmstr,
                 Chapter[parm_chpt_no].SectionN, 0);
  for(i=0; i<Chapter[parm_chpt_no].SectionN; i++)
    XmStringFree(xmstr[i]);
  free(xmstr);
  XmListSelectPos(index_list_sec,
       (CurChapter == parm_chpt_no) ? CurSection+1 : 1, False);
}

void list_cb(w, client_data, call_data)
 Widget w;
 XtPointer client_data, call_data;
{
  XmListCallbackStruct *LMData =
    (XmListCallbackStruct *) call_data;

  switch((int)client_data) {
    case 1:
      selected_chapter = LMData->item_position-1;
      selected_section = (selected_chapter == CurChapter) ?
                          CurSection : 0;
      print_current_section(selected_chapter);
      break;
    case 2:
      selected_section = LMData->item_position - 1;
      break;
  }
}

void index_cb(w, client_data, call_data)
 Widget w;
 XtPointer client_data, call_data;
{
  switch((int)client_data) {
    case 1:
      CurChapter = selected_chapter;
      CurSection = selected_section;
      if(CurChapter == 0 && CurSection == 0) {
        nargs = 0;
        XtSetArg(args[nargs], XmNsensitive, (XtArgVal) False);
        nargs++;
        XtSetValues(pbutton_prev, args, nargs);
      } else {
        nargs = 0;
        XtSetArg(args[nargs], XmNsensitive, (XtArgVal) True);
        nargs++;
        XtSetValues(pbutton_prev, args, nargs);
```

```
        }
        if(CurChapter == ChapterN-1 &&
           CurSection == Chapter[CurChapter].SectionN-1) {
          nargs = 0;
          XtSetArg(args[nargs], XmNsensitive, (XtArgVal) False);
          nargs++;
          XtSetValues(pbutton_next, args, nargs);
        } else {
          nargs = 0;
          XtSetArg(args[nargs], XmNsensitive, (XtArgVal) True);
          nargs++;
          XtSetValues(pbutton_next, args, nargs);
        }
        fill_screen();
        break;
    }
}

void call_about_dialog()
{
  Widget about_dialog, about_button;

  about_dialog = XmCreateInformationDialog(mainwindow,
                                           "about_dialog", NULL, 0);

  about_button = XmMessageBoxGetChild(about_dialog,
                                      XmDIALOG_HELP_BUTTON);
  XtUnmanageChild(about_button);

  about_button = XmMessageBoxGetChild(about_dialog,
                                      XmDIALOG_HELP_BUTTON);
  XtUnmanageChild(about_button);

  XtManageChild(about_dialog);
}
```

TUTORIO.C:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <Xm/Xm.h>
```

```
#include "x11tutor.h"

FILE *finput;
CHAPTER Chapter[10];
int ChapterN=0, CurChapter=-1, CurSection=-1;

PCXHEADER pcx_header;
short int pcx_columns, pcx_rows, pcx_colors;
char *pcx_colormap, *pcx_pixels;
unsigned short *image_pixels;
Boolean is_pcx_loaded=False;


Boolean load_text(filename, filebuffer)
 char *filename;
 String *filebuffer;
{
  struct stat statbuffer;
  Cardinal filelength;

  if((finput=fopen(filename, "r"))==NULL) {
    puts("Dosya Alamad!");
    return False;
  }
  stat(filename, &statbuffer);
  filelength = statbuffer.st_size;
  *filebuffer = (String) XtMalloc(sizeof(char)*(filelength+1));
  fread(*filebuffer, sizeof(char), filelength, finput);
  *(*filebuffer+filelength) = '\0';
  fclose(finput);
  return True;
}

Boolean read_index()
{
  char str[256];
  int i, j, k, l;

  if((finput=fopen("X11.tut", "r"))==NULL) {
    puts("Index Dosyas Alamad!");
    ChapterN = 0;
    return False;
  }
  while(1) {
    fgets(str, 255, finput);
    if(strlen(str)!=1 && str[0]!=' ') break;
  }
```

```
sscanf(str, "CHAPTER NUMBER: %d", &ChapterN);

for(i=0; i<ChapterN; i++) {
  while(1) {
    fgets(str, 255, finput);
    if(strlen(str)!=1 && str[0]!=' ') break;
  }
  for(k=0; str[k]!='"'; k++); k++;
  for(l=0; str[k]!='"'; k++, l++)
    Chapter[i].Title[l] = str[k];
  Chapter[i].Title[l] = 0;
  while(1) {
    fgets(str, 255, finput);
    if(strlen(str)!=1 && str[0]!=' ') break;
  }
  sscanf(str, "SECTION NUMBER: %d", &Chapter[i].SectionN);
  for(j=0; j<Chapter[i].SectionN; j++) {
    while(1) {
      fgets(str, 255, finput);
      if(strlen(str)!=1 && str[0]!=' ') break;
    }
    for(k=0; str[k]!='"'; k++); k++;
    for(l=0; str[k]!='"'; k++, l++)
      Chapter[i].SectionTitle[j][l] = str[k];
    Chapter[i].SectionTitle[j][l] = 0;
    while(1) {
      fgets(str, 255, finput);
      if(strlen(str)!=1 && str[0]!=' ') break;
    }
    sscanf(str, "SECTION FILE NAME: %s",
           Chapter[i].SectionFileName[j]);
    while(1) {
      fgets(str, 255, finput);
      if(strlen(str)!=1 && str[0]!=' ') break;
    }
    sscanf(str, "GRAPHIC FILE NAME: %s",
           Chapter[i].GraphicFileName[j]);
    while(1) {
      fgets(str, 255, finput);
      if(strlen(str)!=1 && str[0]!=' ') break;
    }
    sscanf(str, "SOURCE FILE NAME: %s",
           Chapter[i].SourceFileName[j]);
  }
}
fclose(finput);
CurChapter = CurSection = 0;
```

```
/*  for(i=0; i<ChapterN; i++) {
    printf("CHAPTER TITLE: %s\n", Chapter[i].Title);
    printf("Section NUMBER: %d\n", Chapter[i].SectionN);
    for(j=0; j<Chapter[i].SectionN; j++) {
      printf("SECTION TITLE: %s\n", Chapter[i].SectionTitle[j]);
      printf("SECTION FILE NAME: %s\n",
             Chapter[i].SectionFileName[j]);
      printf("GRAPHIC FILE NAME: %s\n",
             Chapter[i].GraphicFileName[j]);
      printf("SOURCE FILE NAME: %s\n",
             Chapter[i].SourceFileName[j]);
    }
  } */
  return True;
}

Boolean load_pcx(pcx_name)
 char *pcx_name;
{
  register int i, x, y;
  register unsigned char *p;
  register unsigned short *q;
  int packets, count, image_packets;
  unsigned char packet;

  finput = fopen(pcx_name, "rb");
  fread(&pcx_header.identifier, 1, 1, finput);
  if (pcx_header.identifier != 0x0a) {
    puts("Not a PCX image file");
    return False;
  }
  fread(&pcx_header.version, 1, 1, finput);
  fread(&pcx_header.encoding, 1, 1, finput);
  fread(&pcx_header.bits_per_pixel, 1, 1, finput);
  fread(&pcx_header.left, 2, 1, finput);
  fread(&pcx_header.top, 2, 1, finput);
  fread(&pcx_header.right, 2, 1, finput);
  fread(&pcx_header.bottom, 2, 1, finput);
  fread(&pcx_header.horizontal_resolution, 2, 1, finput);
  fread(&pcx_header.vertical_resolution, 2, 1, finput);
  pcx_columns = (pcx_header.right-pcx_header.left)+1;
  pcx_rows = (pcx_header.bottom-pcx_header.top)+1;
  pcx_colors = 16;
  pcx_colormap = (unsigned char *) malloc(3*256*
                                          sizeof(unsigned char));
  fread((char *) pcx_colormap, 3, (int) pcx_colors, finput);
  p = pcx_colormap;
```

```
fread(&pcx_header.reserved, 1, 1, finput);
fread(&pcx_header.planes, 1, 1, finput);
fread(&pcx_header.bytes_per_line, 2, 1, finput);
fread(&pcx_header.palette_info, 2, 1, finput);
for (i=0; i<58; i++) fgetc(finput);
packets=pcx_rows*pcx_header.bytes_per_line*pcx_header.planes;
pcx_pixels = (unsigned char *) malloc(packets*
                                      sizeof(unsigned char));
p = pcx_pixels;
while (packets > 0) {
  packet = fgetc(finput);
  if ((packet & 0xc0) != 0xc0) {
    *p++ = packet;
    packets--;
    continue;
  }
  count = packet & 0x3f;
  packet = fgetc(finput);
  packets -= count;
  while (--count >= 0) *p++ = packet;
}
pcx_colors = 1<<(pcx_header.bits_per_pixel*pcx_header.planes);
if (pcx_colors > 16) {
  fread(&pcx_header.colormap_signature, 1, 1, finput);
  fread(pcx_colormap, 3, pcx_colors, finput);
} else if (pcx_colors == 2) {
  pcx_colormap[0] = 0;
  pcx_colormap[1] = 0;
  pcx_colormap[2] = 0;
  pcx_colormap[3] = 255;
  pcx_colormap[4] = 255;
  pcx_colormap[5] = 255;
}
image_packets = pcx_columns * pcx_rows;
image_pixels = (unsigned short *) malloc(image_packets*
                                      sizeof(unsigned short));
q = image_pixels;
if (pcx_header.planes > 1) {
  register int bits, mask;

  for (i=0; i<image_packets; i++) *(q++) = 0;
  for (y=0; y < pcx_rows; y++) {
    p = pcx_pixels + (y*pcx_header.bytes_per_line*
        pcx_header.planes);
    for (i=0; i<pcx_header.planes; i++) {
      q = image_pixels + y*pcx_columns;
      for (x=0; x < pcx_header.bytes_per_line; x++) {
```

```
          bits = (*p++);
          for (mask=0x80; mask != 0; mask>>=1) {
            if (bits & mask) *q |= 1 << i;
            q++;
          }
        }
      }
    } else for (y=0; y<pcx_rows; y++) {
      p = pcx_pixels + y*pcx_header.bytes_per_line;
      switch(pcx_header.bits_per_pixel) {
        case 1: {
          register int bit;

          for (x=0; x < (pcx_columns-7); x+=8) {
            for (bit=7; bit >= 0; bit--)
              *(q++) = ((*p) & (0x01 << bit) ? 0x01 : 0x00);
            p++;
          }
          if((pcx_columns % 8) != 0) {
            for (bit=7; bit >= (8-(pcx_columns % 8)); bit--)
              *(q++) = ((*p) & (0x01 << bit) ? 0x01 : 0x00);
            p++;
          }
          break;
        }
        case 2: {
          for (x=0; x < (pcx_columns-3); x+=4) {
            *(q++) = (*p >> 6) & 0x3;
            *(q++) = (*p >> 4) & 0x3;
            *(q++) = (*p >> 2) & 0x3;
            *(q++) = (*p) & 0x3;
            p++;
          }
          if ((pcx_columns % 4) != 0) {
            for (i=3; i >= (4-(pcx_columns % 4)); i--)
              *(q++) = (*p >> (i*2)) & 0x03;
            p++;
          }
          break;
        }
        case 4: {
          for (x=0; x < (pcx_columns-1); x+=2) {
            *(q++) = (*p >> 4) & 0xf;
            *(q++) = (*p) & 0xf;
            p++;
          }
```

```
        if ((pcx_columns % 2) != 0) {
          *(q++) = (*p >> 4) & 0xf;
          p++;
        }
        break;
      }
      case 8: {
        for (x=0; x < pcx_columns; x++) {
          *(q++) = (*p);
          p++;
        }
        break;
      }
      default: break;
    }
  }
  free(pcx_pixels);
  fclose(finput);
  is_pcx_loaded = True;
  return True;
}

void destroy_pcx()
{
  free(image_pixels);
  free(pcx_colormap);
  is_pcx_loaded = False;
}
```

X11TUTOR:

```
X11tutor.geometry: 800x600+1+1
*fontList: -*-helvetica-bold-r-normal-*-14-100-*-*-*-*-iso8859-9

*label_chpt.labelString: Bölüm Etiketi!
*label_chpt.alignment:   XmALIGNMENT_BEGINNING
*label_chpt*fontList:    -*-helvetica-*-r-*--18-*
!*label_chpt.foreground: NAVYBLUE

*label_sec.labelString: Konu Etiketi!
*label_sec.alignment:   XmALIGNMENT_END
*label_sec*fontList:    -*-helvetica-*-r-*--18-*
!*label_sec.foreground: NAVYBLUE

*label_src.labelString: Kaynak Etiketi!
*label_src.alignment:   XmALIGNMENT_CENTER
```

```
*label_src*fontList:     -*-helvetica-*-r-*--14-*
!*label_src.foreground: BLUE

*text_sec.editMode:          XmMULTI_LINE_EDIT
*text_sec.editable:          False
*text_sec.scrollHorizontal: True
*text_sec.scrollVertical:    True
*text_sec*fontList:          -*-helvetica-*-r-*--12-*

*text_src.editMode:          XmMULTI_LINE_EDIT
*text_src.editable:          False
*text_src.scrollHorizontal: True
*text_src.scrollVertical:    True
*text_src*fontList:          -*-helvetica-*-r-*--12-*

*draw_pic.height:        400
*draw_pic.width:         400
*draw_pic.resizePolicy: XmRESIZE_NONE

*scroll_win.scrollingPolicy: XmAUTOMATIC

*cbutton_one.labelString: Sistem
*cbutton_one.mnemonic:    S

*cbutton_two.labelString: Görüntü
*cbutton_two.mnemonic:    G

*cbutton_three.labelString: Çalıştır
*cbutton_three.mnemonic:    a

*cbutton_help.labelString: Yardım
*cbutton_help.mnemonic:    Y

*pbutton_open.labelString:      Aç
*pbutton_open.mnemonic:         A
*pbutton_open.acceleratorText: F3
*pbutton_open.accelerator:      <Key>F3:

*pbutton_exit.labelString:      Çıkış
*pbutton_exit.mnemonic:         k
*pbutton_exit.acceleratorText: Alt-x
*pbutton_exit.accelerator:      Alt<Key>x:

*pbutton_next.labelString:      Sonraki
*pbutton_next.sensitive:        False
*pbutton_next.mnemonic:         o
*pbutton_next.acceleratorText: F6
```

```
*pbutton_next.accelerator:      <Key>F6:

*pbutton_prev.labelString:      Önceki
*pbutton_prev.sensitive:        False
*pbutton_prev.mnemonic:         n
*pbutton_prev.acceleratorText:  F5
*pbutton_prev.accelerator:      <Key>F5:

*pbutton_idx.labelString:       Dizin
*pbutton_idx.sensitive:         False
*pbutton_idx.mnemonic:          D
*pbutton_idx.acceleratorText:   F7
*pbutton_idx.accelerator:       <Key>F7:

*pbutton_help.labelString:      Yardım
*pbutton_help.mnemonic:         a
*pbutton_help.acceleratorText:  F1
*pbutton_help.accelerator:      <Key>F1:

!*pdmenu_two*TearOffModel:          TEAR_OFF_DISABLED
!*TearOffControl.height:            10
!*TearOffControl.shadowThickness:   3

*index_dialog.dialogTitle:          X11tutor - Dizin
*index_dialog.dialogStyle:          XmDIALOG_APPLICATION_MODAL
*index_dialog.okLabelString:        Tamam
*index_dialog.cancelLabelString:    Vazgeç

*index_label_chpt.labelString:      Bölüm
*index_label_chpt.alignment:        XmALIGNMENT_BEGINNING
*index_label_chpt*fontList:         -*-helvetica-*-r-*--14-*
*index_label_chpt.foreground:       NAVYBLUE

*index_label_sec.labelString:       Konu
*index_label_sec.alignment:         XmALIGNMENT_BEGINNING
*index_label_sec*fontList:          -*-helvetica-*-r-*--14-*
*index_label_sec.foreground:        NAVYBLUE

*index_list_chpt.scrollBarDisplayPolicy: XmSTATIC
*index_list_chpt*fontList:              -*-helvetica-*-r-*--12-*
*index_list_chpt.visibleItemCount:       10
*index_list_chpt.selectionPolicy:        XmBROWSE_SELECT
*index_list_chpt.width:                  150

*index_list_sec.scrollBarDisplayPolicy: XmSTATIC
*index_list_sec*fontList:               *-helvetica-*-r-*--12-*
*index_list_sec.visibleItemCount:       10
```

```
*index_list_sec.selectionPolicy:        XmBROWSE_SELECT
*index_list_sec.width:                  150

*about_dialog.dialogTitle:       X11tutor - Hakkında
*about_dialog.dialogStyle:       XmDIALOG_APPLICATION_MODAL
*about_dialog.okLabelString:     Tamam
*about_dialog.cancelLabelString: Vazgeç
*about_dialog.messageString: \
      Yazan : Nuran Metin Esen\n\
      Danışman : Prof.Dr. Metin Demiralp\n\
      Istanbul Teknik Üniversitesi\n\
      Fen-Edebiyat Fakültesi\n\
      Istanbul - 1996
```

X11.TUT:

```
CHAPTER NUMBER: 6

CHAPTER TITLE: "KONU 1: Giriş"
SECTION NUMBER: 9

SECTION TITLE: "Giriş"
SECTION FILE NAME: chp1sec0.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

SECTION TITLE: "X Windows Sürümleri"
SECTION FILE NAME: chp1sec1.txt
GRAPHIC FILE NAME: chp1sec1.pcx
SOURCE FILE NAME: NONE

SECTION TITLE: "Görüntüler ve Ekranlar"
SECTION FILE NAME: chp1sec2.txt
GRAPHIC FILE NAME: chp1sec2.pcx
SOURCE FILE NAME: NONE

SECTION TITLE: "Sunumcu-Istemci Modeli"
SECTION FILE NAME: chp1sec3.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

SECTION TITLE: "Pencere Yönetimi"
SECTION FILE NAME: chp1sec4.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE
```

```
SECTION TITLE: "Olaylar"
SECTION FILE NAME: chp1sec5.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

SECTION TITLE: "X'e Genişlemeler"
SECTION FILE NAME: chp1sec6.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

SECTION TITLE: "X Pencere Sistemi, Yazılım Mimarisi"
SECTION FILE NAME: chp1sec7.txt
GRAPHIC FILE NAME: chp1sec7.pcx
SOURCE FILE NAME: NONE

SECTION TITLE: "Xlib'e Bakış"
SECTION FILE NAME: chp1sec8.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

CHAPTER TITLE: "KONU 2: Çalıştırma"
SECTION NUMBER: 5

SECTION TITLE: "Bir Görüntü Sunumcusu Açma"
SECTION FILE NAME: chp2sec1.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

SECTION TITLE: "Görüntü Hakkında Bilgi Elde Etme"
SECTION FILE NAME: chp2sec2.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: tutorio.c

SECTION TITLE: "Pencere Yaratma ve Yok Etme"
SECTION FILE NAME: chp2sec3.txt
GRAPHIC FILE NAME: chp2sec3.pcx
SOURCE FILE NAME: NONE

SECTION TITLE: "Pencerelerin Tasviri, Hareketi ve \
               Ortaya Çıkarılması"
SECTION FILE NAME: chp2sec4.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

SECTION TITLE: "Bitmapler ve Pixmaplerin Tanımlanması"
SECTION FILE NAME: chp2sec5.txt
GRAPHIC FILE NAME: NONE
```

```
SOURCE FILE NAME: NONE

CHAPTER TITLE: "KONU 3: Grafik Içeriği"
SECTION NUMBER: 1

SECTION TITLE: "Grafik Içeriği"
SECTION FILE NAME: chp3sec1.txt
GRAPHIC FILE NAME: chp3sec1.pcx
SOURCE FILE NAME: NONE

CHAPTER TITLE: "KONU 4: Dörtgenler, Çizgiler, Noktalar, \
                Yaylar ve Yazı"
SECTION NUMBER: 5

SECTION TITLE: "Dörtgenler, Çizgiler, Noktalar, Yaylar ve Yazı"
SECTION FILE NAME: chp4sec0.txt
GRAPHIC FILE NAME: chp4sec0.pcx
SOURCE FILE NAME: NONE

SECTION TITLE: "Pencerede Yazı Çizimi"
SECTION FILE NAME: chp4sec1.txt
GRAPHIC FILE NAME: chp4sec1.pcx
SOURCE FILE NAME: NONE

SECTION TITLE: "Karakterlerin Eninin Bulunması"
SECTION FILE NAME: chp4sec2.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

SECTION TITLE: "Ekranda Karakterlerin Basımı"
SECTION FILE NAME: chp4sec3.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

SECTION TITLE: "Imleç Yaratma"
SECTION FILE NAME: chp4sec4.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

CHAPTER TITLE: "KONU 5: Olaylar"
SECTION NUMBER: 2

SECTION TITLE: "Olaylar"
SECTION FILE NAME: chp5sec0.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE
```

```
SECTION TITLE: "Olay Tipleri"
SECTION FILE NAME: chp5sec1.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

CHAPTER TITLE: "KONU 6: Renk Kullanımı"
SECTION NUMBER: 2

SECTION TITLE: "Renk Kullanımı"
SECTION FILE NAME: chp6sec0.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE

SECTION TITLE: "Sadece Okunan Renk Hücreleri"
SECTION FILE NAME: chp6sec1.txt
GRAPHIC FILE NAME: NONE
SOURCE FILE NAME: NONE
```

# ÖZGEÇMİŞ

NURAN METİN ESEN 1971 yılında İstanbul'da doğmuş, lise öğrenimini Bahçelievler lisesinde tamamlamıştır. 1988 yılında girdiği İstanbul Teknik Üniversitesi, Fen-Edebiyat Fakültesi, Matematik Mühendisliği bölümünü 1992 yılında bitirmiş, aynı yıl İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Sistem Analizi programında yüksek lisans öğrenimine başlamıştır. 1995 yılında PAKOM'da programcı/sistem analist olarak iş hayatına atılmıştır. 1996 yılında TurkishBank'a sistem analist olarak geçmiş olup, halen bu görevini sürdürmektedir.