İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

GÖRSELYÖRE TASARIMINDA DHTML ÜZERİNDEN GÖRÜNTÜ FONTLARI KULLANIMI : TEX2DHTML ÇEVİRİCİSİ

YÜKSEK LİSANS TEZİ

Mat. Müh. Rahşan TİKEN
(509970401011)

Tezin Enstitüye Verildiği Tarih : 5 Haziran 2000
Tezin Savunulduğu Tarih : 30 Haziran 2000

Tez Danışmanı :	Prof. Dr. Metin DEMİRALP
Diğer Jüri Üyeleri	Prof.Dr. Nüzhet DALFES
	Doç.Dr. Hasan DAĞ

HAZİRAN 2000

ÖNSÖZ

Bu çalışma ile internet ortamında kullanılacak yeni yazı tipleri üreten ve bu yazı tiplerini kulanarak TeX çıktılarını HTML dökümanına çeviren programlardan oluşmuş bir paket hazırlanmıştır. Yapılan çalışma farklılıkların kullanımı ve uygulama geliştirmeyi zorlaştırdığı internet ortamı için bir standart oluşturmaya çalışması ve TeX'i internete açması bakımından önemlidir.

Bu çalışmayı yönlendiren, geniş vizyonu ve hoşgörüsü bizlere daima örnek olmuş Sayın Hocam Prof. Dr. Metin Demiralp'e teşekkürü bir borç bilirim. Ayrıca desteklerinden ötürü başta annem olmak üzere tüm aileme ve yardımlarından ötürü Mat.Yük. Müh. Engin Kavas ile İnşaat Yük. Müh. Özgür Ay'a teşekkürlerimi sunarım.

Haziran, 2000 **Mat. Müh. Rahşan Tiken**

İÇİNDEKİLER

ŞEKİL LİSTESİ

GÖRSELYÖRE TASARIMINDA DHTML ÜZERİNDEN GÖRÜNTÜ FONTLARI KULLANIMI : TEX2DHTML ÇEVİRİCİSİ

ÖZET

Sanal bir ortam olan internet, özelikle son yıllarda çok farklı uygulamalar ile günlük hayatımızın vazgeçilmez bir parçası olmaya başlamıştır ve gelecekte öneminin giderek artacağı görülmektedir. Bunu bizlere düşündüren, teknolojideki hızlı değişimin internet ve benzeri ortamları, işlevleri ve yararlılığı giderek artan ortamlar olarak hayatımıza sokmasıdır. Ancak bu hızlı değişim beraberinde teknik olanakların artırılması ihtiyacını da taşımaktadır. Sözkonusu ihtiyaçlardan birisi de görselyöre tasarımlarının tüm farklı mimarilerden bakıldığında aynı görünüme sahip kılınmasıdır. Yazı tipleri, bu farklılaşma unsurularından birisidir. Yapılan çalışma ile özellikle akedemik ortamlarda kullanılmakta olan TeX kelime işlemcisinin çıktılarını internete taşıyabilmek ve -taşıma işlemi sonunda farklı tarayıcılardan farklı görüntülere ulaşılmaması için- TeX kelime işlemcisi tabanlı yeni, dinamik yazı tipleri oluşturmak hedeflenmiştir.

Gerek dinamik yazı tipleri gerekse TeX çıktılarının HTML'ye çevrilmesi konularında alternatif çalışmalar bulunmaktadır. Ancak her çalışmanın avantajları ve dezavantajları vardır. Bu çalışmada alternatiflerin getiri ve götürüleri gözönünde bulundurularak karşılaştırmalar yapılmakta ve diğer çalışmalara ait örneklemeler verilmektedir. Tüm bunlar ikinci bölümde ele alınan konulardır.

Bu çalışmada yazı tiplerinin üretilmesi için Bash kabuk programlama ve TeX programlama dillerinden; ghostscript, convert gibi grafik çıktıları üretmede kullanılacak uygulamalardan yaralanılmıştır. Kabuk programlama, her bir aşaması komut satırından gerçekleştirilecek olan işlemler topluluğunun belli bir sırada, bir önceki çıktının sonuçlarını kontrol edecek şekilde çalıştırılmasına olanak tanıdığından çalışma için çok uygun olmuştur. Yazı tiplerinin TeX çıktıları olarak üretilmesi ise TeX'in internet ortamında görüntülenmesi hedefi güdüldüğünden çalışmanın amacına çok uygundur.

TeX çıktılarının istenilen formata getirilmesi için ise Perl programlama dilinden yararlanılmıştır. Burada formatlama ile kasdedilen ele alınan dökümanın çıktıda görülmek istenilen satır boylarına göre yeniden yazdırılması, gerekli yerlerde Türkçe dilbilgisi kuralları gözönünde bulundurularak heceleme işlemi yapılması ve dinamik fontları kullanabilen bir yapı haline getirilmesidir. Perl, özellikle veriler üzerinde yapılacak işlemlerde esnek ve etkili bir kullanım sağladığından çalışma için uygun olmaktadır.

Oluşturulan dökümanın tarayıcı aracılığı ile internete açılmasında ise dinamik HTML ve javascript kullanılmıştır.

Uygulamaları geliştrime aşamasında kullanılan yöntemler ve geliştirme amaçları ayrıntıları ile üçüncü bölümün konusudur.

Sonuç bölümünde, belirlenen hedef doğrultusunda alınan sonuçlar ve belirlenen amaç için yapılanların ihtiyacı ne ölçüde karşıladığı ve ileride yapılması gerekenlerden bahsedilmektedir.

Bu çalışma ile, hedeflenen fonksiyonlarını tam olarak yerine getiren, İnternet şartlarına uygun bir sistemin ne şekilde hazırlanabileceğine ilişkin bir model geliştirilmeye çalışmış ve birtakım örnekler üzerinde uygulanarak hayata geçirilebileceği gözlemlenmiştir.

AT A WEBSITE DESIGN IMAGE FONT USAGE VIA DHTML : TEX2DHTML CONVERTER

SUMMARY

During recent years, the virtual media called Internet has become more and more important in our daily lives and it is expected to increase its importance in the future. The effect of the improvements in technology on Internet and similar environments helped us to make this forecast. It must be also kept in mind that these changes come together with some new techical requirements. One of the technical requirements is to be able to view the images as the same way when accessed from browsers running on different architectures. Fonts are one of the things that create difference. The aim of this study is to make the outputs from TeX, which is a very popular wordprocessor in academical studies, usable in internet environment and create new dynamic fonts for TeX.

It is a fact that there are other studies on both dynamic fonts and conversion of TeX outputs to HTML. Each study has cons and pros. In the second part of this study, comparisons have been made taking in to consideration of the positive and negative sides of different alternatives and some examples from other studies have been supplied.

In this study, "Bash" shell programming and TeX programming languages are used for programming purposes, where as Ghostscript and Convert are used for producing the visual outputs. It has been realised that shell programming method was very convenient for this study as it allows to run a group of transactions in an order from the command line, checking the results of the previous outputs. As the aim is to make TeX outputs visible from the Internet environment, creating the fonts as TeX outputs for this study is also very suitable.

Perl programming language has been used for formatting the TeX outputs. Here the phrase "formatting" is used for describing the required actions to create the printed document in desired font size, enabling spelling in required places according to the

rules present in Turkish language and creating a structure using dynamic fonts. Perl is chosen for the study as it can be used on data in a powerful and flexible way.

Dynamic HTML and Javascript are used for publishing the created document on the Internet.

The methods and development purposes are explained in detail in the third part of this study..

In conclusion part, the results and the things has to be done in future studies are explained.

Within this study, it has been tried to demonstrate how a suitable model for Internet can be developed that can perform all the desired actions. It is also shown by examples that it can be used in real life.

1. GİRİŞ

Bu çalışmada, Internet yayıncılığında kullanılmak üzere dinamik font üretmeye yönelik ve bu fontları Unix tabanlı görselyöre sunucuları üzerinden kullanabilme yeteneğine sahip program paketleri hazırlanmıştır.

Belirlenen amaca ulaşabilmek için font kapasitesi yüksek ve yeni font üretimi açısından son derece esnek olan TeX kelime işlem dili baz alınmakla birlikte, yaratılan fontlara Internet özelliklerinin katılmasında DHHTML ve buna bağlı olarak JavaScript dillerinden de faydalanılmıştır. Unix ve türevi işletim sistemleri için geliştirilmiş olan TeX dilinden, Internet yayınlarına font üretme yeteneklerinin gerçekleştirilmesinde Perl ve benzeri kabuk dilleri kullanılmıştır. Unix sistemine özgü bazı format dönüşümü yöntemleri de bu çalışmada kullanılmıştır.

Bu kaynakta, ikinci bölümde aynı konularda yapılmış diğer çalışmalar hakında bilgi verilmekte, üçüncü bölümde ise hazırlanan programlar ve kurulan yapı detaylı olarak incelenmektedir.

1.1 Çalışmanın Amacı

Farklı yazılım mimarisine sahip bilgisayarların bulunduğu Internet ortamında yayıncılık yapmanın en sıkıntılı yanlarından biri, görselyörenin tüm bu farklı mimarilerdeki tarayıcılardan bakıldığında da aynı görünüme sahip kılınmasıdır. Görünümdeki farklılaşmalar font kulanımlarında da ortaya çıkmaktadır. Ayrıca akademik özellikli dökümanlar gibi, font çeşitliliği isteyen yayınlar Internet'in standart yayıncılık dilleriyle sunulması zor olan yayınlardır. Bu çalışmada, bu iki konu üzerinde durulmuş ve farklı işletim sistemlerine sahip istemcilerdeki farklı tarayıcılar üzerinde, yukarıda bahsi geçen sınırların ve problemlerin aşılması yönünde yöntemler belirlenmiş ve bu doğrultuda program paketleri oluşturulmuştur.

İstemcilerde kullanılan Unix, MicroSoft Windows gibi işletim sistemlerindeki yapısal farklılıklar font yapılarında da mevcuttur. Internet yayıncılığında kullanılan HTML dili font yönetiminde oldukça zayıf bir yapıya sahiptir. Bir kelime işlem dilinin özelliklerine sahip olan HTML diliyle sunulan bir görselyörenin fontlarının

tarayıcıda görüntülenmesi, görselyöreye bağlanan istemcide bulunan işletim sisteminin sahip olduğu fontları kullanmaktan öteye gitmemektedir. Dolayısıyla, örneğin Microsoft Windows işletim sistemlerinde sıkça kullanılan fontlardan biri olan "Arial" fontuna göre HTML diliyle hazırlanan bir görselyöreye, Unix işletim sistemi kullanan bir istemci bağlandığında karşılaşılacak olan sayfa çok farklı görünecektir. Bu farklılık hem font biçimlerinde hem de fontların tarayıcı üzerindeki yerleşimlerinde hissedilebilecektir. Bunun gibi yaşanması olası problemler sadece farklı işletim sistemleri için geçerli olmayacak, ayrıca Internet Explorer ve Netscape gibi farklı tarayıcılar kullanan istemciler, hatta aynı tarayıcıyı kullanan ancak font ayarları farklı olan istemciler içinde benzeri sonuçlar doğuracaktır.

Yapılan çalışmanın amaçlarından biri de istemcinin işletim sisteminde bulunan fontlarla kısıtlı olan HTML dilinin font yapısının zenginleştirilmesi için yeni yöntemler belirlenmesidir. Örneğin, Türkçe hazırlanmış bir görselyörenin, sisteminde Türkçe fontlar bulunmayan bir istemci tarafından sağlıklı olarak görüntülenmesi mümkün değildir, bu konu daha geniş bir yelpazeden değerlendirildiğinde şu olgu ortaya çıkmaktadır, matematiksel bilgiler içeren bir kaynağın Internet'te yayınlanabilmesi için HTML'nin yetersiz kalacağı bir gerçektir. Bu aşamada, uygulanması gereken yöntemin Internet kıstasları düşünülerek oluşturulması çalışmada belirlenen hedefler arasındadır. Buna göre; oluşturulacak fontların işletim sisteminden bağımsız ve en yaygın tarayıcıların anlayabilecekleri bir formatta yaratılması amaçlanmıştır. Bu amaç doğrultusunda fontların büyüklüğü az olan resim dosyaları şeklinde yaratılması en uygun çözüm olacağı kanısına varılmıştır. Resim dosyaları olarak yaratılan fontların, istemci tarafına ilk sefer yüklendikten sonra, görselyöreye tekrar bağlanıldığında yeniden istemci tarafına çekilmeyip istemci cache'inden tekrar kullanılabilir yapıda olmaları da istenilen özelliklerden biridir. Sistem, TeX kelime işlem dili baz alınarak tasarlandığından, yaratılabilecek olan fontların sayısının sınırsız olacağı yadsınamaz bir gerçek olacaktır. Ayrıca, yaratılacak olan bu fontların rahat bir şekilde Internet dökümanlarında kullanılması için gerekli olan altyordamların hazırlanması ve bunun Internet programlamanın günümüzdeki doğal yapısı ve işleyişi dışına çıkılmadan tasarlanması da dikkat edilen hususlardan biridir.

Çalışmada, özellikle TeX dökümanlarının Internet üzerinden yayınlanabilmesi için yapılmış olan çeviriciler araştırılmıştır. Araştırılan çeviriciler arasında gerek LaTeXtoHtml gibi standart olarak işletim sistemiyle birlikte gelen çeviriciler olduğu gibi ticari amaçlı olarak yazılan çeviriciler de bulunmaktadır. Bu çeviricilerin

karşılaştırmalarının yapılması ve bu analiz sonrası çevirimi en etkin şekilde yapabilen ancak görselyörenin performansını olumsuz etkilemeyecek, görselyöre sunucuya en uyumlu yapıda çeviricilerin oluşturulması üzerinde de durulmuştur.

2. DİNAMİK YAZI TİPİ ÜRETİMİ VE TEXTOHTML MODELLERİ

Yapılan çalışma dinamik font üretimi ve TeX dökümanlarının HTML'ye çevrilmesi konusunda programları kapsamaktadır. Bu bölümde; her iki konuda da yapılmış olan diğer çalışmalardan bahsedilmektedir.

2.1 Dinamik Font Kullanımı

İnternet; kişilerin, hayal güçlerini sınır tanımaksızın diğer kişilere aktarabilecekleri bir ortamdır. Görsel yörlerde içerik kadar önemli bir diğer öğe de görselyörenin görünümüdür. Tasarımda kullanılan hareketli, hareketsiz, renkli, ses öğeleri olan nesneler yörenin ilgi uyandırıcı olmasınında etkin rol oynamaktadır. Tasarımdaki bu çeşitlilik kullanılan yazı stilleri ile de yakından ilgilidir. Tasarımcılar, özgün yazı tipleri kullanımı ile çalışmaya has birtakım değerleri, kullanıcılara daha rahat aktarabilmektedirler. Ayrıca belli bir yöreye hitabeden görselyörelerde, o yörenin karakteristikleri farklı yazı tipleri kullanımını gerektirmektedir. Örneğin; Japonca, Arapça, Rusça gibi Latin alfabesinin kullanılmadığı dillerde, bu dillerin kendi fontlarının oluşturulmasının gerekliliği tartışılmazdır. Aynı şekilde matematiksel, mühendislik ya da mimarlığa özel bazı karakterler için kullanıcının kendi yazı tiplerini tanımlaması gerekmektedir. Kişiye özgü yazı tiplerinin tanımlanabilmesi için dinamik yazı tipleri kullanılmaktadır.

Dinamik yazı tipi kullanımı HTML dizaynın son aşamalarından birisidir. Standart bir HTML sayfasında yazı tipleri kullanılmakta ancak bu yazı tiplerinin kullanıcı tarafında görülebilmesi için kullanıcının bilgisayarında tanımlı yazı tiplerinden birisi olması gerekmektedir. Dinamik yazı tipi kullanımı bu gerekliliği ortadan kaldırmaktadır. Dinamik yazı tipleri kullanılarak hazırlanılan görselyörelerdeki yazı tipleri, kullanıcı tarafında tanımlı olmasına gerek olmaksızın, dökümanla birlikte indirilmektedir. Belli bir sürüm numarasının üzerindeki tarayıcılarda bu indirilme işlemi otomatik olarak yapılmaktadır. Dinamik yazı tipleri ile oluşturulan görselyörelerde döküman üzerinde kopyalama yapmak ve kopyalanan parçaları farklı bir dökümana yapıştırabilmek mümkündür [1].

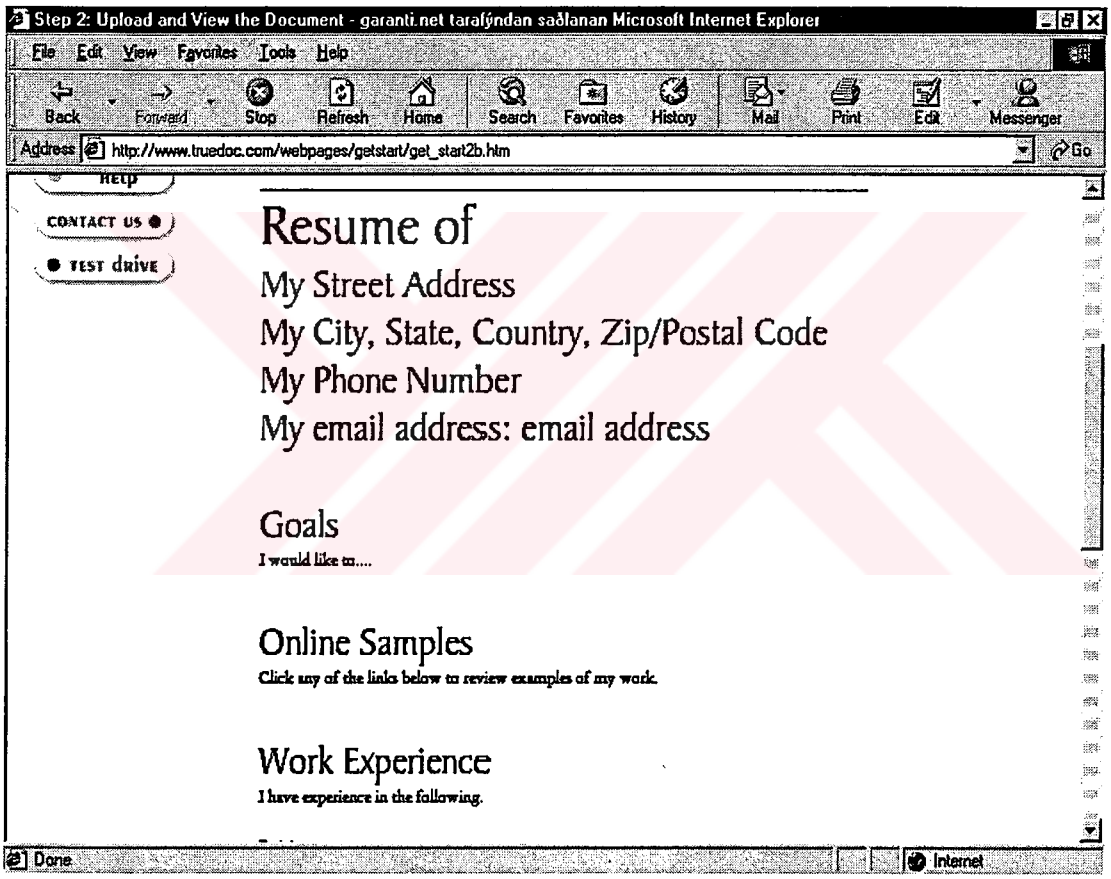
2.1.1 Dinamik Font Kullanımına Örnekler

"HTML'de Dinamik Font Kullanımı" başlığı altında incelendiğinde dinamik yazı tipi kullanımına yönelik farklı özelliklerde görselyöreler bulunmaktadır. Bunlara verilebilecek örneklerden ilki; "www.truedoc.com" adresinden ulaşılabilecek olan Bitstream tarafından oluşturulmuş görselyöredir. Bu görselyörede Bitstream, kendi Webfont teknolojisini kullanarak ürettiği yazı tiplerinin, kullanıcının hazırlayacağı HTML dökümanlarında kullanılmasına yönelik bir uygulama gerçekleştirmektedir. Burada, hazırlanacak olan HTML dosyasında kullanılan yazı tipleri için "www.truedoc.com" adresinin referans gösterilmesi yöntemi ile döküman yaratılmaktadır. Aşağıda Bitstream yazı karakterlerini kullanması istenen bir görselyöre için oluşturulan HTML dosyasının bir parçası yeralmaktadır.

```
<HTML>
<HEAD>
<TITLE>My Resume</TITLE>
<LINK REL="FONTDEF" SRC="http://www.truedoc.com/pfrs/AmeriGarmnd.pfr
">
<LINK REL="FONTDEF" SRC="http://www.truedoc.com/pfrs/BakerSignet.pfr">
<SCRIPT LANGUAGE="JavaScript"
SRC="http://www.truedoc.com/activex/tdserver.js"> </SCRIPT>
<link> <STYLE TYPE="text/css"> P { font-family: "AmeriGarmnd BT"; size: 3;
color: #000000; } </STYLE>
</HEAD>
<BODY>
<FONT FACE="BakerSignet BT" SIZE=7>Resume of </FONT> <BR>
<FONT FACE="BakerSignet BT" COLOR="#990000" SIZE=6> My Street Address
</FONT>
<BR> <FONT FACE="BakerSignet BT" COLOR="#990000" SIZE=6> My City,
State, Country, Zip/Postal Code </FONT>
<BR><FONT FACE="BakerSignet BT" COLOR="#990000" SIZE=6> My email
address: <A HREF="mailto:my_user_name@my_isp">email address </A>
</FONT>
</BODY>
```

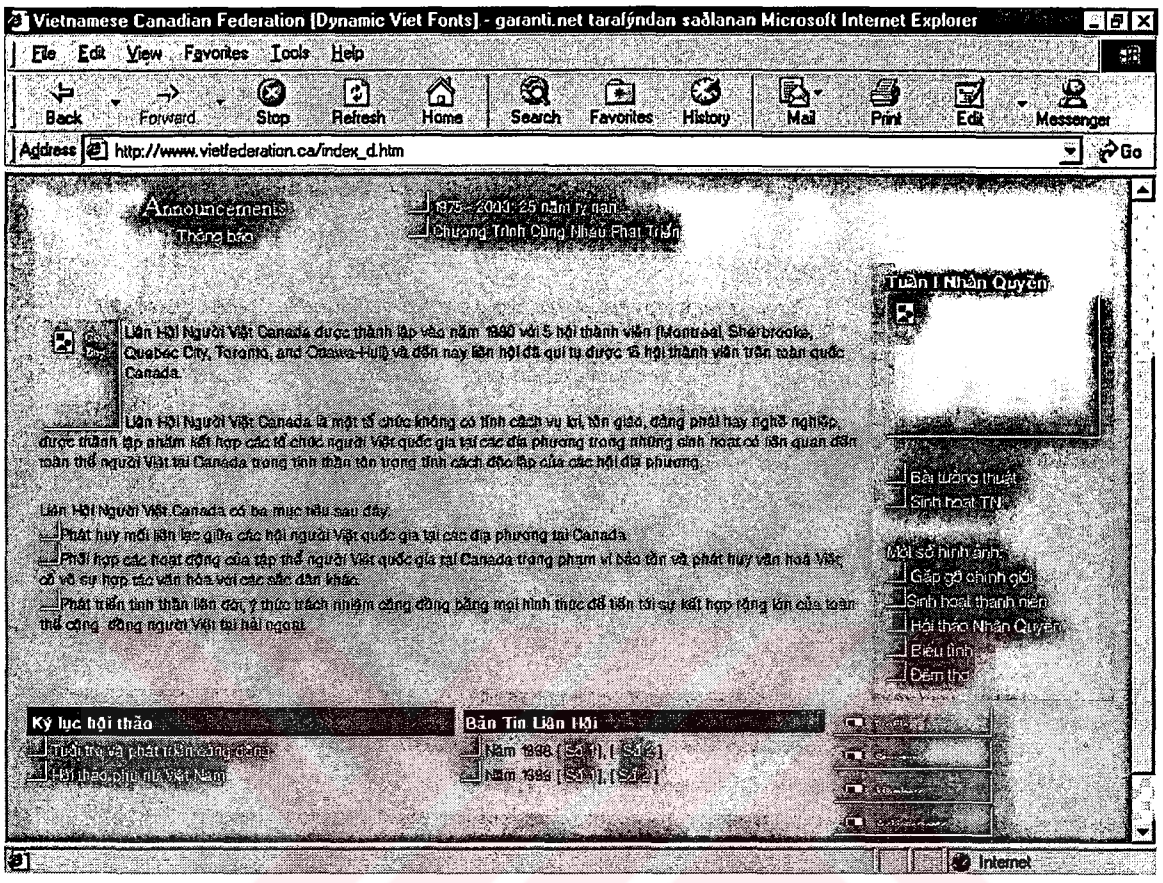
</HTML>

Bu döküman, script dili olarak JavaScript kullanmaktadır. HTML etiketlerinden biri olan Link etiketi ile yazı tiplerinin tanımlı olduğu PFR dosyalarına referans verilmektedir. PFR (Portable Font Resource) adı verilen dosyalar platformdan bağımsız olarak karakter görüntü tanımlarını barındıran ve bir kere yüklendiğinde cashe bellekten çağırılabilecek olan dosyalardır. Dosyalar için gerekli bağlantı Link etiketi ile tanımlandıktan sonra "Font" etiketi kullanılarak istenilen yerlerde istenilen yazı tiplerinin kullanımı sağlanmaktadır. Bu programın çıktısı Şekil 2.1'de görüldüğü gibi olacaktır:



Şekil 2.1 Dinamik Font Kullanımı Örneği

Yine dinamik yazı tipi kullanılarak Vietnam alfabesi ile oluşturulmuş bir görselyöre olan Vietnam Federation'a ait görselyöreye [3] "www.vietfederation.ca" adresinden ulaşılabilir. Görselyöreden örnek bir sayfa Şekil 2.2'de görülmektedir.



Şekil 2.2 Dinamik Yazı Tipleri - Vietnam Alfabesi İle Oluşturulmuş Görselyöre

2.1.2 Dinamik Yazı Tipleri Üreten Yazılımlar

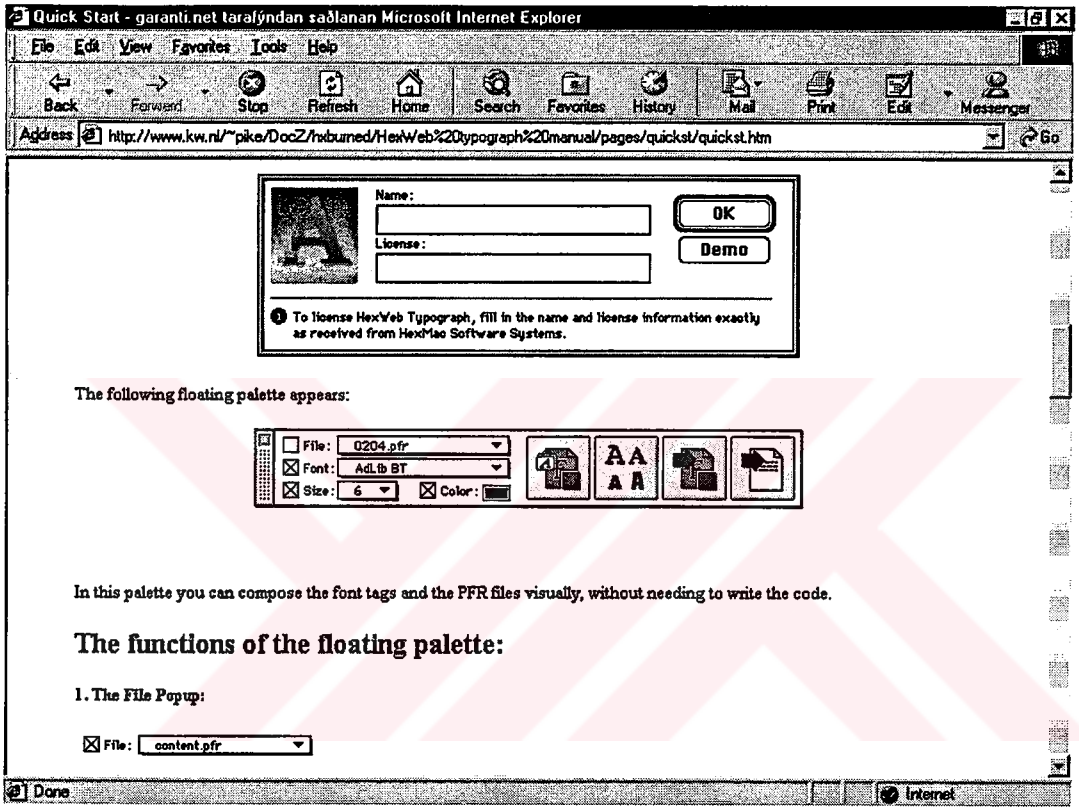
Dinamik yazı tiplerinin yaratılması işlemi, kullanıcının makinesinde yer alan True Type ya da Postscript yazı tipi kütüphanlerinden yararlanarak isteğe uygun yeni bir yazı tipi yaratılması ve bunun bir PFR (Portable Font Resource) dosyası olarak kaydedilmesinden ibarettir [1].

Kulanıcının dinamik yazı tiplerini yaratabilmesini sağlayan yazılımları "Dynamic Font Maker" başlığı altında toplamak olasıdır. Bu ürünlere örnek olarak HexMac firması tarafından yaratılan HexWeb Typograph ve Bitsream firması tarafından yaratılan WebFont Maker adlı yazılımlar ele alınabilir. İnternette yazı tipi üretimine yönelik ücretsiz tek yazılım HexWeb Typograph yazılımıdır.

HexWeb Typography; web yayıncılık yazılımları üreten bir yazılım firması olan HexMac Yazılım Firması ürünüdür. "www.hexmac.com" adresinden firmanın

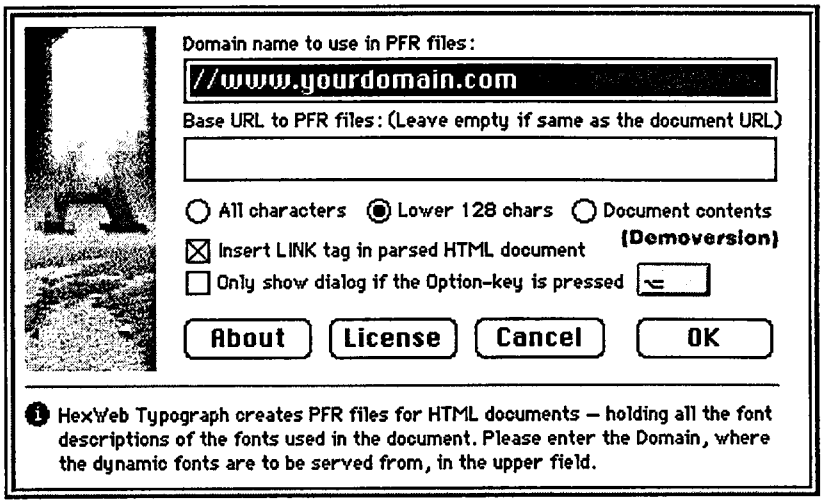
görselyöresine oradan da ürünler başlığı altında Typography ürünüyle ilgili bilgileri erişilebilmektedir [4].

Typographer yazılımı çalıştırıldığında yaratılıcak yazı tipine ilişkin tip, büyüklük, renk vb. bilgilerin seçileceği bir menüden istenilen özellikler seçilir. Şekil 2.3'de menü örneği görülmektedir.



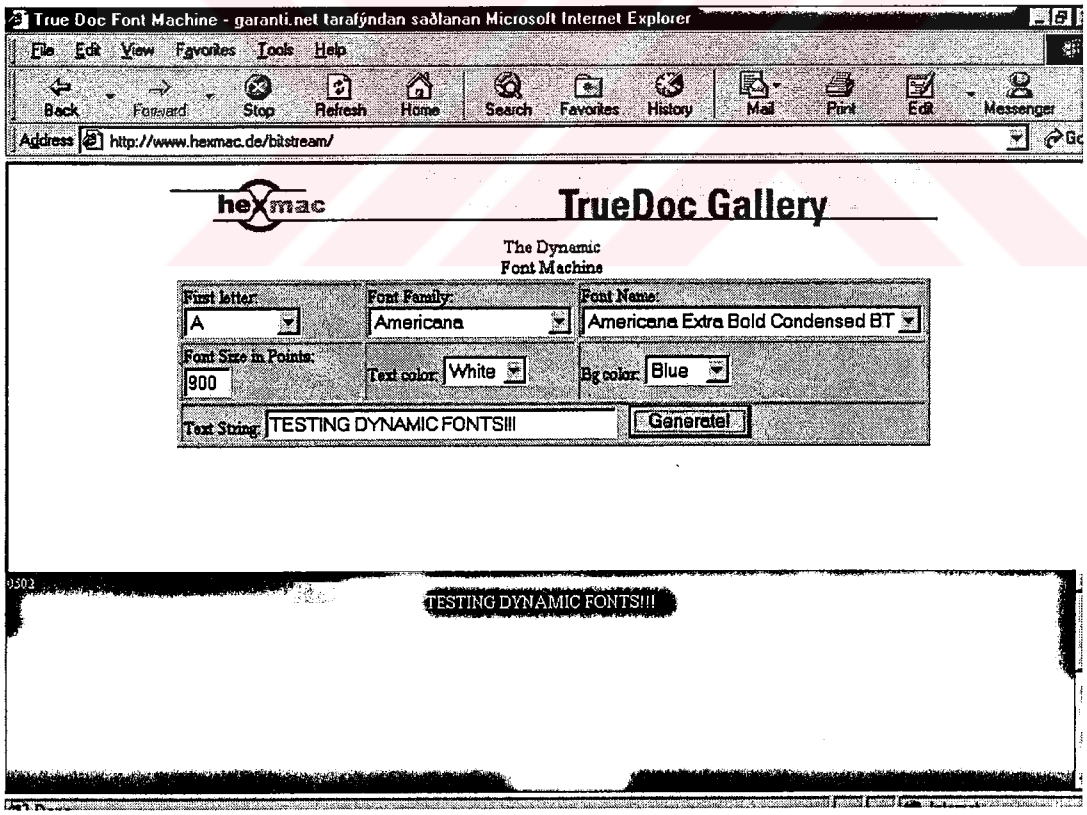
Şekil 2.3 Typehographer ekran çıktıları.

Burada "file", yaratılacak dinamik fonta verilecek olan dosya adını; "font", PFR dosyaları yaratılması için kullanıcının bilgisayarında bulunması gereken, dinamik fontun bazı olacak font tipini; "size", punto ölçü biriminden büyüklüğü; "color" ise renk seçimini sağlayacak öğelerdir. "Kaydet" tuşuna basıldıktan sonra kullanıcının karşısına yaratılacak yazı tipinin yeriyle ilgili birtakım bilgilerin doldurulacağı bir menü gelir. Şekil 2.4. Burada da gerekli bilgiler doldurularak "OK" tuşuna basıldığında yeni yazı tipi, kullanıma hazır hale gelmektedir [5].



Şekil 2.4 Typographer Ürünü Ekran Çıktısı Örneği

Typhographer kulanılarak üretilen yazı tipleri www.hexmac.de/bitstream adresinden online olarak denenebilmektedir [6]. Şekil 2.5'de bu görselyöre ve üretilmiş bir yazı örneği görülmektedir.



Şekil 2.5 Dinamik Yazı Tipleri Üretimi Örneği

2.1.3 Mevcut Modellerin Değerlendirilmesi

Dinamik yazı tipleri üretimiyle ilgili çalışmalar izlendiğinde, yapılanların görselyöre tasarımcıları için yeni bir dönem başlattığını farketmemek olanaksızdır. Gerek Hex Mac gerek Bitstream tarafından üretilmiş yazı tiplerinin kullanımı ile işletim sistemlerinin ve tarayıcıların yazı tipleri kısıtlarına bağlı kalmaksızın tasarımlar yapılabilmektedir.

Ancak tasarımlarda bu yazı tiplerinin kullanımı düşünüldüğünde işletim sistemleri ile yüklenen yazı tipleri kullanımından ayrılan tek nokta daha kalabalık bir yazı tipi galerisine sahip olmasıdır. Onun dışında başka bir üreticiye olan bağımlılık ortadan kalkmamaktadır.

Yapılan çalışmanın, internet üzerinden bulunabilecek yazı tipleri ile yer değiştiremeyecek nitelikte olmasının bir nedeni şudur: Bu çalışmada TeX kelime işlemcisi ile oluşturulan dökümanların internete taşınabilmesi için gerekli dinamik yazı tiplerinin oluşturulması hedeflenmiştir. Bu durumda kullanılacak yazı tiplerinin TeX'in ürettiği yazı tiplerinden az olmaması gerekliliği kaçınılmazdır. Bu nedenle yapı tiplerine kaynak olarak TeX kelime işlemcisi çıktıları alınmaktadır; böylece üretilecek yazı tipleri ile TeX çıktılarının eşleşebilecek olması garanti altına alınmış olmaktadır. Aynı şekilde Typographer ve benzeri yazılımlar ile oluşturulacak yazı tiplerinin TeX ile uyuşmama riski bulunmaktadır.

Hazır yazı tipleri kulanımında verilen referans ile kullanılacak yazı tipine ait dosyalar indirilmektedir. Bu çalışma sonucunda oluşturulan yazı tipleri ise çıktı dosyasında olacak ihtiyaca göre çağırılacak ya da çağırılmayacak ve bir kereden sonraki kullanımlarda cashe bellekte yer ettiğinden daha hızlı bir erişim sağlanacaktır. Bu ugulama ile kullanılmayacak yazı tiplerinin erişim performansı düşürmesi engellenmektedir.

Sonuç olarak; ürettiği yazı tipleri, İstanbul Teknik Üniversitesi bünyesinde sürdürülen Sanal Eğitim Projesi'nde kullanılacak dinamik yazı tipi üretim modeli mevcut yazı tipleri ile karşılanamayacak bir ihtiyacı gidermektedir.

2.2 TEX ve HTML

TeX, bir programlama dilinin esneklikleri ile yazım özelliklerini birleştiren, tüm bilimsel formül ve sembolleri içeren, sayfa üzerindeki alana mikron düzeyinde hakimiyeti sağlayan ve daha bir çok avantaja sahip olan bir yazılım sistemidir. Bu özelliklerinden dolayı, bir programlama dili olarak da algılanabilir.

HTML ise WWW sayfalarının temelini oluştururan metin biçimlendirme (Hypertext Markup Language) dilidir. HTML ile, metinler ve resim gibi diğer elementler kolayca bir sayfa üzerinde biçimlendirilebilir ve bu sayfalar bağlantılarla birbirlerine eklenir. Bu mantık ile Web sayfasındaki kelimeler veya sayfa üzerindeki resim, düğme gibi bileşenler kulanılarak, sayfalar arasında dolaşılabilir.

Son yıllarda DHTML (Dynamic HTML) adı verilen yeni sürümünün çıkmasıyla bir gelişim süreci yaşan HTML, nesne ve katman mantığını da içine alarak daha etkileşimli bir yapıya kavuşmuştur. DHTML'de JavaScript ön plana çıkmaktadır. DHTML'nin olumsuz yönü farklı tarayıcıların, bazı Javascript komutlarında ayrılması ve dolayısıyla standart bir yapıya henüz kavuşmuş olmamasıdır [7].

Her ne kadar işlevleri artırılsa da, HTML özellikle yazı tipleri yönetimi konusunda zayıf bir dildir. TeX gibi akademik ve mühendislik ortamlarda etkinliği kanıtlanmış bir dil, HTML ile kıyaslandığında çok daha avantajlı bir yapıdadır. HTML ile TeX'te hazırlanabilen nitelikte karmaşık formatlı dökümanlar oluşturmak olası değildir.

Bu durumda TeX'i HTML'ye çevirecek bir yazılımın çok yetenekli bir kelime işlemcinin internete açılmasını sağlaması açısından önemi büyüktür.

2.1.4 TexToHtml Modelleri

TeX'te hazırlanmış bir dökümanı HTML dökümanıma çeviren yazılımlar bulunmaktadır. Bunlardan bazıları formüllerini .gif formatı görüntü dosyaları olarak oluşturmakta, bazıları görüntü dosyaları kullanmaksızın HTML'ye çevirmektedir.

Adından en sık bahsedilen TeXToHtml çeviricisi Ian Hutchinson tarafından oluşturulmuş TeX2Html'dir. Aslında bu ürün, ilk olarak TTH adı ile oluşturulmuş çeviricinin ileriki bir versiyonudur. TTH, ticari olmayan kullanımlar için ücretsiz olarak kullanılabilen bir üründür. Tex2Html ise ürünün ticari versiyonudur. TTH

yazılımına "http://huthinson.belmont.ma.us/tth" adresli görselyöre üzerinden ulaşılabilir.

TTH, TeX'in ürettiği tüm özel karakterleri, hemen hemen tüm matematik formüllerini, girdi dosyalarını, makro ve tablo tanımlarını HTML dökümanına taşıyabilmektedir.

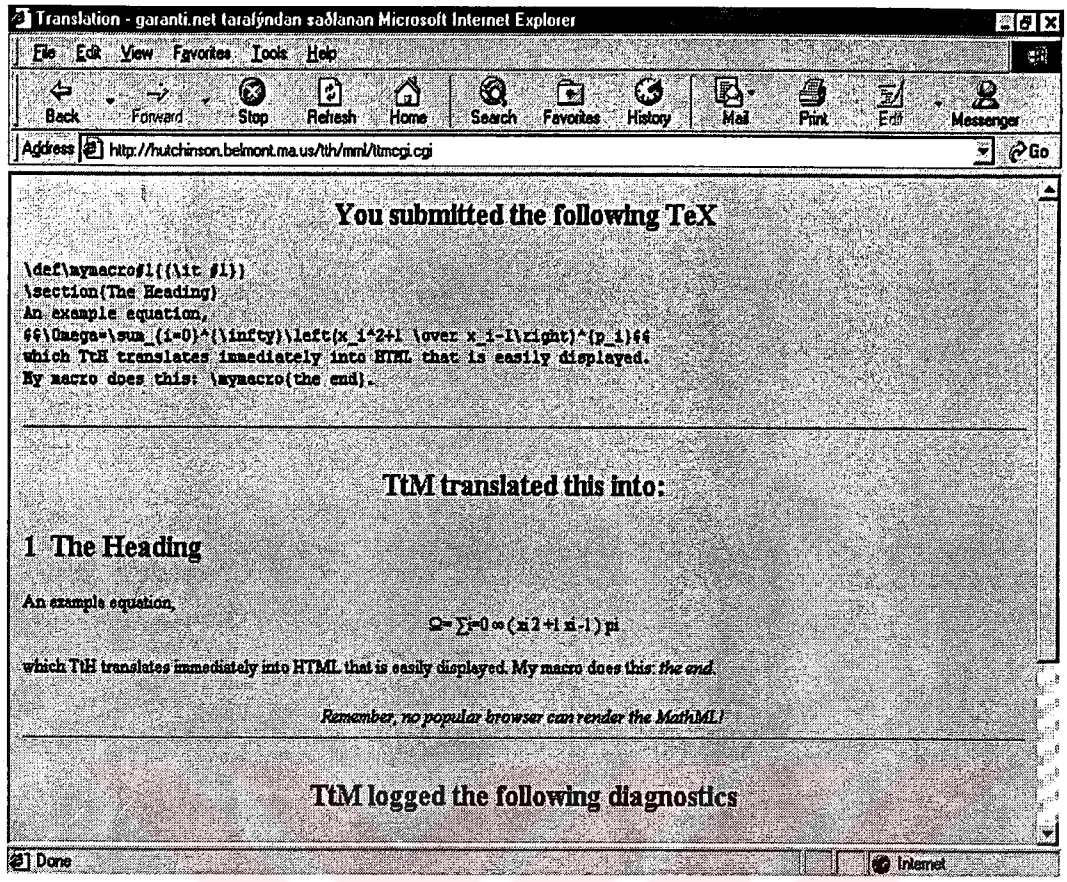
TTH'de formüller mümkün olduğu kadar, HTML'ye çevrilmektedir. Çevrilemeyen formüllerin görüntülenmesi için fractions kullanılmaktadır. Ayrıca bazı formüller <font face=``symbol"> etiketi kullanılarak semboller şeklinde dökümana alınmaktadır. Burada "http://hutchinson.belmont.ma.us/tth/ Xfonts.html" adresinden de izlenebilecek olan dinamik yazı tiplerine referans verilmektedir. TTH, çevrilmesi zor olan formülleri -özellikle türevli ifadeleri- görüntü dosyaları olarak HTML'ye taşımaksızın HTML'deki imkanları kullanarak semantik olarak doğru ifadeler üretmeye çalışmaktadır.

TTH, TeX'tekinin aksine formüllerde yer alan boşlukları kaldırmamaktadır. Bunun sebebi birçok tarayıcıda metematiksel yazı tipleri birbirine çok yakın olarak yazılması ve okunurluğu zorlaştırmasıdır. Bu nedenle TTH, boşlukları silmemektedir [8].

Şekil 2.6'da TTH kullanılarak web üzerinden yapılan bir TeX çevrimi örneği görülmektedir.

TeX'den çıktılarını HTML çeviren uygulamalara başka bir örnek Nikos Drakos tarafından yaratılmış olan LaTeX2HTML'dir. Bu yazılımda doğrudan HTML'de gösterilemeyen yapılar için görüntü dosyası oluşturulması methodu kullanılmıştır. Programlar Perl ile yazılmış, yazı takımı için Latex, görüntü dosyaları oluşumu için dvips, ghostscript ve netbm uygulamaları kullanılmıştır. Bu çeviriciye "http://cbl.leeds.ac.uk/nicos" adresili görselyöreden ulaşılabilir [12].

Eitan Gurari tarafından oluşturulmuş TeX4ht da LaTeX2HTML gibi HTML tarafından alıgılanamayan karakterler için .gif görüntü dosyaları oluşturmaktadır. Görüntü dosyalarının oluşturulması dvips ve Image Magic programları ile sağlanmaktadır. TeX4Ht'ye "http://www.cis.ohio-state.edu/~gurari/TeX4ht" adresli görselyöre üzerinden ulaşılabilir.



Şekil 2.6 TTH yazılımı kullanılarak TeX'den HTML'ye yapılan çevrim örneği.

2.1.5 Mevcut Modellerin Değerlendirilmesi

TeX çıktılarının HTML'ye çevrilmesiyle ilgili ele alınan ürünlerin birbirlerine kıyasla avantajlı ve dezavantajlı yanları vardır. LaTeX2HTML ve TeX4ht matematik için görüntü dosyaları oluşumunun zorluğunu barındırmaktadır. TTH ve TeX2HTML'nin karmaşık formülasyonlarda yazıları tam olarak temsil edememesi ise bu tür kompleks ifadelerin çevrimini başarısız kılmaktadır [9].

Bunun yanı sıra ürünler standart TeX dosyaları için başarılı denilebilecek çıktılar üretmektedir ancak bu çalışma ile oluşturulmasına başlanılan çevirici, yine TeX çıktılarından üretilen dinamik yazı tiplerinin kullanımına imkan tanıması yönünden, amaca daha uygun gözükmektedir. Ayrıca oluşturulmuş programlar kullanıcı tarafında herhangi bir incelemeye gerek kalmaksızın, Türkçe yazım kurallarını dikkate alarak, heceleme yapmakta; sayfa düzenini çıktıdaki satır uzunluklarını kontrol altına alarak istenilen formata getirilebilmektedir. Bu gibi özellikleri, TeX'in özellikleri ile birleştirildiğinde Türkçe için özel olarak tasarlanmış bir çevirici olarak kullanılabilecektir.

3. YAZI TİPLERİ OLUŞTURULMASI VE TEX2DHTML PROGRAMLARI YAPISI

Bu tezde yapılan çalışma; amacı ve işlevi düşünüldüğünde üç parça olarak ele alınabilir. İlk parça kullanılacak karakter takımlarının dinamik yazı tipleri olarak oluşturulması, ikinci parça TeX dökümanının yorumlanarak yeni yazı tiplerini kullanacak hale getirilmesi, üçüncü parça ise hazırlanan dökümanın internet üzerinde görüntülenebilmesi amacıyla DHTML dökümanına çevrilmesidir. Bu bölümde pakedin her bir parçası kullanılan teknikler ve çalışma temelleri açısından detaylı olarak incelenecektir.

3.2 TeX Yazı Takımının Oluşturulması

TeX yazı sistemi, bilgisayar için hem sıradan text dosyaları hem de her çeşit matematiksel yazılım için yüksek kalitede bir karakter takımına sahiptir. Esneklik, hecelenebilir olmak vb. özelliklerinden dolayı akedemik ve mühendislik ortamlarda en çok kulanılan yazı sistemidir. Bu nedenle üretilecek olan yeni yazı tipleri için TeX çıktılarının kullanılması düşünülmüştür.

3.1.1 TeX'de Karakterlerin Gösterilimi

TeX'te karakterler girdi ve çıktı karakterleri olmak üzere iki açıdan ele alınabilir. Dosyadan okunan her karakter sonuç dosyasında, kullanılan yazı takımındaki karaktere çevrilecek bir girdidir. Her bir karakterin ASCII kod tablosunda bir kod karşılığı vardır ve TeX; karakteri okuduğunda buna karşılık gelen kategori kodunu bularak işlemleri bu kod ile gerçekleştirir. Örneğin, "T" harfinin ASCII kodu karşılığı 84'dür. Dosyadan bir giriş karakteri alındığında TeX bunu; okunan karakterin kategori koduna karşılık gelen karakterin bulunduğu bir kutu üretecek komut olarak algılar. Çıktı dosyası bu kutucukların düzenlenerek biraraya getirilmesinden oluşmaktadır [10].

Karakteri klavyeden girmek yerine kategori kodunu kullanarak bastırmak için "\char" komutu kullanılabilir. Örneğin;

\char 92 komutunun çıktısı "\" karakteri olacaktır.

Özellikle bir takım kontrol değişkenleri yazdırılmak istendiğinde bu yöntem kullanılmaktadır. Ayrıca "\catcode" komutu ile kullanıcı kendi katogori kodu-karakter ikilisini tanımlayabilmektedir. Örneğin;

\catcode \@ = 11

komutu ile ad işaretinin katagori koduna 11 atanmaktadır.

Üçüncü bir karakter gösterilim şekli ise kontrol dizileri ile oluşturulan gösterimdir. Klavyedeki tuşlar, yazmak istediğimiz sınırsız sayıda işlev için yeterli olamadığından TeX, " \ " (ters-ayıraç) olarak ifade edilen bir yardımcı karakter geliştirmiştir. "\"dan hemen sonra yazılanlar, sıradan olmayan ve TeX'e önceden tanımlanmış özel işlevlerin yerine getirilmesini sağlarlar. Bunlara "kontrol dizisi" denir. Yardımcı karakterler ve arkasından da kontrol dizilerini kullanarak ifadeler oluşturulur. Türkçe karakterler bu kontrol dizileri ile oluşturulmuşlardır. Örneğin; "\i" "ı" harfi, "\"o" "ö" harfini oluşturmak için yaratılmış kontrol dizileridir. Matematik sembolleri için de benzer tanımlamalar yapılmıştır.

3.1.2 TeX'de Yazı Tipleri Kulanımı

Bir TeX yazı tipi aynı dizayn, aynı stil ve aynı büyüklük kulanılarak oluşturulan 256 çıktı karakteridir. Yazı tiplerinin kod tabloları, belirlenen karakterin o yazı tipindeki karşılığını ifade eder. Bu kod tablosu, karakterlerin ASCII kodu karşılıklarının tutulduğu kod tablosu ile uyumludur. Yalnız matematiksel ifadelerin yer aldığı bazı yazı tiplerinde karakterler yoktur, onun yerine \char komutu kullanılarak oluşturulan gösterilim vardır [10].

Bir fontun dökümanda kullanılabilmesi için bir TeX komutu ile çağırılması ve yüklenmesi gerekir. TeX fontlarını çağıran komut "\font" komutudur. Kullanım şekli:

\font*font_adı*=font metriği

Burada *\font_adı* o yazı tipini isimlendirmek için kullanılacak kontrol dizisini, metriği ise yazı tiplerinin yüklenmesi için kullanılacak .tfm dosyasının adını işaret eder. Öneğin;

\font\yeni_fontlar = cmbx12

\yeni_fontlar { deneme }

cbx12 metriğinde yazılmış "deneme" çıktısını verecektir.

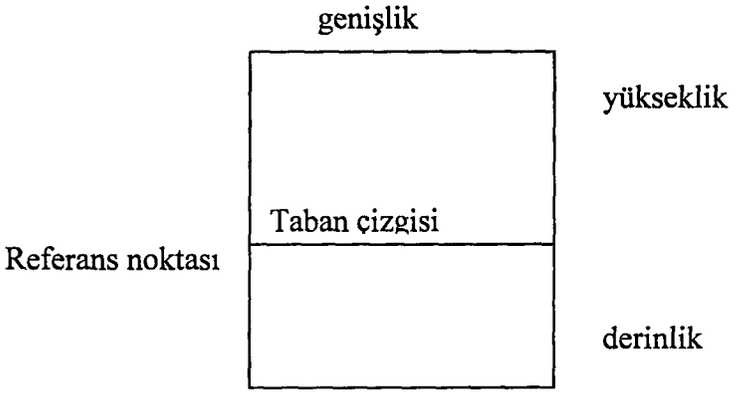
Farklı bir yazı tipi seçilinceye kadar, bir önceki seçim geçerli olacaktır. Font metrik dosyası her typeset karakteri için gerekli olan her karakterin büyüklüğü, boşluk vb. bilgilerin tutulduğu dosyalardır. Bu dosyalarda harflerin şekilleri hakkında hiçbir bilgi bulunmaz. Şekillerle ilgili bilgiler metrik dosyası ile aynı adı taşıyan ancak .pk (paketlenmiş format – paked format), .gf (generik format) uzantılı dosyalarda bulunur. Çıktı biriminizde daha önceden var olan bir yazı tipi için şekil dosyasına tekrar ihtiyaç duyulmaz.

TeX kendi içinde çok sayıda yazı tipi bulundurmaktadır (Computer Modern Fonts) ancak istenirse farklı kaynaklarının yazı tipleri de kullanılabilir.

3.1.3 TeX'de Karakterlerin Yazdırılması

Bölüm 2.1.1'de de bahsedildiği gibi TeX'de tüm sayfa bir kutu olarak ele alınır ve sayfa üzerinde yer alan her bir karakter kendi başına bir kutucuktur. Özetle bir TeX çıktısı kutu içinde yuvalanmış birçok kutudan oluşmaktadır.

Herbir kutunun eni, boyu ve derinliği vardır:



Şekil 3.1 TeX'de bir karakterin gösterilimi.

Taban çizgisi basılacak karakterler için bir alt sınır oluşturur. Örneğin "g" harfi taban çizgisinin altına taşarken "h" harfinde böyle bir taşma olmamaktadır. Yükseklik taban çizgisinden kutunun üst sınırına kadar olan, derinlik taban çizgisinden kutunun alt sınırına kadar olan uzaklıktır. Sol köşe ile taban çizgisinin kesişim noktasına referans noktası adı verilir [14].

Kullanıcı isterse \hbox ve \vbox kumutlarını kullanarak kendi kutucuklarını oluşturabilir:

\hbox{ *kelimeler* } komutu ile kelimeler'in uzunluğu ne olursa olsun bölünmeksizin tüm kelimeleri içine alacak bir kutu oluşturulur. Örneğin;

\hbox{ Burada yazılan kelimeler soldan sağa gruplanacaktır. }

Yukarıdaki örnekte referans noktası başlangıç kabul ederek soldan sağa doğru parantezler içinde yazılan kelimeleri içeren bir kutu oluşturulacaktır.

\vbox{ *kelimeler* } komutu ile kelimeler kaç satırdan oluşursa oluşsun yeni bir sayfaya geçilmeksizin tüm kelimeleri içine alacak bir kutu oluşturulur. Örneğin;

```
\vbox{ Burada
       yazılan
       kelimeler
      yukarıdan
      aşağıya
      gruplanacaktır. }
```

Yukarıdaki örnekte referans noktası başlangıç kabul ederek yukarıdan aşağıya doğru parantezler içinde yazılan kelimeleri gruplayarak, içeren bir kutu oluşturulacaktır.

3.1.4 Yazı Tipleri Yaratma Programı

Kullanılacak yazı tiplerinin görüntü dosyalarını oluşturan program paketi TeX, PostScript ve Bash, Gawk kabuk progamlama dilleri kullanılarak oluşturulmuş

birkaç programı içermektedir. Herbir programlama dili kullanılmaya en elverişli olduğu konular söz konusu olduğunda çalışmaya dahil edilmiştir.

Fontların yaratılmasıyla ilgili olarak temel yapıyı teşkil eden "Fontyarat" adlı pogram Bash kabuk programlama dili ile hazırlanmıştır. Bu programda kabuk programlama dili kullanımı sayesinde komut satırından sırayla yapılması gereken birçok işlem, ürettikleri çıktılar da gözönüne alınarak bir başlık alında toplanmaktadır.

Dosyanın başına, programın bir ön derleme aşaması olmaksızın komut satırından çalıştırılabilir olması istendiğinden bunun bir kabuk komutları dosyası olduğunu belirten;

#!/bin/bash

satırı yazılmıştır. Bu belirteç sayesinde dosya içinde yer alan her satırın çalıştırılabilir komutlara karşılık geldiği ve Bash kabuk programlama dili kurallarınca çalışması gerektiği belirtilmiş olmaktadır. Ayrıca dosya kaydedildikten "chmod" komutu kullanılarak çalıştırılabilir bir dosya haline getirilmiştir.

> chmod 777 fontyarat

Fontyarat programı parametre olarak üretilmek istenilen yazı tipini, punto ölçü birimi cinsinden büyüklüğünü ve "cmyk" renk sistemine göre renkli yazı tipleri oluşturmada kullanılacak renk arametrelerini almaktadır. Komut satırında;

> fontyarat cmr 25 0 0 0 1

komutu verildiğinde fontyarat programı cmr tipli yazı tiplerinden 25 punto büyüklükte, siyah renkli yazı tiplerine karşılık gelen gif dosyalarının yaratılmasına başlayacaktır.

Programda öncelikle oluşturulacak görüntü dosyalarının tutulacağı bir dizin yaratılmaktadır. Daha sonra üzerinde işlem yapılacak karakter tipi seçilmektedir. Karakterler, alfabetik karakterler ve rakamlar için düz TeX girdilerinden, noktalama işaretleri için \char komutu ile kullanılacak olan ASCII kodlardan ve \def ile tanımları yapılmış özel TeX karakterlerinden seçilmektedir. Her bir karakter "for döngüsü" ile teker teker işleme alınmaktadır. Seçilen üç farklı grup karakterler aşağıda yer almaktadır.

Alfabetik karakterler ve rakamlar:

```
for karakterno in a b c d e f g h I j k l m n o p r s t u v y z w x \
                  A B C D E F G H I J K L M N O P R S T U V Y Z W X \
                  0 1 2 3 4 5 6 7 8 9
do
```

\Char komutu ile yazdırılacak olan özel işaretler:

```
for degisken in 33 34 35 36 37 38 39    \
                 41 42 43 44 45 46 47    \
                 59                       \
                 61 64
do
```

Türkçe karakterler:

```
for harfler in  cT  CT sT ST oT OT uT UT gT GT iT IT 99
do
```

Bir sonraki aşama, üzerinde işlem yapılan karakterin TeX tarafından tanınır, anlaşılır bir karakter olup olmadığının kontrolü amacı ile yatay bir kutu için karakterin yazdırıldığı kutucuk oluşturulmaktadır. Bu kutucuk tek komutları ile oluşturulmakta ve derlenmek üzere "kontrol.tex" adı verilen bir ara dosyaya yazdırılmaktadır.

```
echo "\setbox200=\hbox{"$karakterno"}"                    >> kontrol.tex
echo "\ifdim\wd200>0pt \message{karaktervar}\fi\bye" >> kontrol.tex
```

Oluşturulan dosya TeX derleyicisi ile derlenmektedir. Derlenme işleminin hatasız olarak tamamlanması durumunda derleme sonuçları içinde "karakter var" mesajının yer alması beklenmektedir. Mesajın olup olmadığı Gawk'ın örüntü aratma tekniği kullanılarak kontrol edilmektedir. Karakter bulunursa sonucu yine bir ara kontrol dosyası olan "kontrol.tex" dosyasına yazılacaktır.

```
echo "\setbox200=\hbox{"$karakterno"}"                    >> kontrol.tex
echo "\ifdim\wd200>0pt \message{karaktervar}\fi\bye" >> kontrol.tex
tex kontrol.tex|gawk '/karaktervar/ {print $0}'          >  kontroldosyasi
```

Karakter mevcut ise görüntü dosyasının oluşturulması için yapılacak çalışma başlatılmaktadır.

```
if [ -s "kontroldosyasi" ]
 then
 echo  "\nopagenumbers                    >> aratexdosyasi
```

Burada yapılan işlemler .gif çıktısının elde edilmesinde ilk aşama olacak olan TeX dosyasını oluşturmayı hedeflemektedir. Böylece TeX dosyasının çıktısı olabilecek her karakterin bir yazı tipi olarak tanımlanabilmesi mümkün olmaktadır.

Oluşturulacak TeX dosyasının sonunda "fontdos.tex" olarak adlandırılacak bir dosyada toplanmaktadır. Yukarıda bahsdilen her karakter takımı için bu dosya oluşturulacaktır. Yalnız, Türkçe karakterlerin basılacağı üçüncü grupta bunun için gerekli gerekli makroların daha önceden oluşturulduğu "onparca" adlı dosya "fontdos.tex" dosyasının başına yapıştırılır. Böylece TeX, oluşturacağı karakteri, makro tanımı içinden alabilir.

Türkçe karakterler ile 1 punto'luk boşluğa karşılık gelecek 99 kodlu karakterin ele alındığı en son bölümde bu işlemden önce karakterler için gerekli makro tanımlarının yapıldığı "onparca" adlı dosya aşağıdaki tanımlamaları kapsamaktadır:

```
\def\cT{\c c}
\def\CT{\c C}
\def\sT{\c s}
\def\ST{\c S}
\def\gT{\u g}
\def\GT{\u G}
\def\oT{\"o}
\def\OT{\"O}
\def\uT{\"u}
\def\UT{\"U}
\def\iT{\i}
```

```
\def\IT{\.I}
\def\99{\hglue1pt}
```

Burada tanımlanan her makronun adının bu makro adının daha sonra gif dosyasının adı olacak olması açısından önemi vardır. Yapılan tanımlamalar sayesinde ana programda karakter kodu olarak makro adı verildiğinde TeX parantezler içinde yazılı olan komutu çalıştıracaktır. Örneğin; iT makrosu çağırıldığında "ı" karakterini yazdıracak olan TeX komutu çalışmaktadır, karakter adı 99 olduğunda ise TeX genişliği 1pt olan boşluktan ibaret bir çıktı üretecektir.

Oluşturulan dosyanın derlenmesi sonucu, seçilen karakterin basılı olduğu bir .dvi dosyası elde edilecektir. Basılan sayfada yalnız karakterin bulunduğu kısmı almak anlamlı olduğundan sayfa numarasının olmaması gerekmektedir. Bu nedenle TeX dosyası başına öncelikle sayfa numarası basılmaması gerektiği anlamına gelen "\nopagenumbers" satırı yazdırılmaktadır.

Daha sonraki komutlar;

```
echo "\font\yeni="$1"10 scaled "$200              >> aratexdosyasi
echo "\setbox200=\hbox{\yeni "$karakterno"}"  >> aratexdosyasi
echo "\special {ps: "$3" "$4" "$5" "$6" }"        >> aratexdosyasi
echo "\message{karakter= "$karakterno"}"      >> aratexdosyasi
```

komutlarıdır. Burada da sırası ile programa parametre olarak geçirilen font tipi ve büyüklüğünün, tip ve ölçü olarak tanımlandığı "font" komutu; karakterin içine yuvalanacağı kutunun yaratılmasını sağlayacak olan "setbox" komutu; yazı tiplerinin renklerinin belirendiği "special" komutu dosyaya yazdırılmaktadır. Son olarak dosya derlendiğinde hangi karakter üzerinde işlem yapıldığının anlaşılabilmesi amacı ile karakterin yazdırılacağı "\message" komutu eklenmiştir.

Bu komutların TeX dosyası içinde rolleri şöyle olacaktır:

\font\font_adı=scaled ölçü ; yazım şekli ile kullanılan font komutu, font adı ile belirtilen yazı tipinden ilgili .tfm dosyasının bulunarak belirlenen büyüklükte oluşturulmasını sağlar. Bu komutun program içindeki kullanımında $1 ile gösterilen bir yazı tipi olduğu görülmektedir. $1, programa parametre olarak geçirilen

değerlerden ilkinin alınması gerektiğini belirten bir kabuk değişkenidir. TeX'de "font" ile birlikte "scaled" belirteci kullanılırsa bu büyüklüğün 1000'e bölüneceği anlamına gelmektedir. O nedenle, oluşturulması istenilen yazı tipi büyüklüğü 1000 ile çarpılarak gerçek değerine getirilmektedir.

\special komutu "{}" işaretleri içinde verilen listenin .dvi çıktısı oluşturulurken direkt çıktıya yazılmasını sağlar. Buradaki kullanımı ile special, yazı tiplerinin cmyk renkleri ile basılmasını sağlayacaktır. CMYK, cyan-blue-yellow-black renklerinin birer harfleri alınarak oluşturulan kısaltmadır. Herbir renk için 0-1 arasında değerler girilmektedir. Girilen değer o rengin, çıktıdaki payını gösterecektir.

\message komutu ile "{}" işaretleri arasındaki text çıktıda mesaj olarak üretilir. Burada da üzerinde çalışılan karakter mesaj olarak ekranda gösterilmektedir.

Buraya kadar üzerinde işlem yapılan karakterin ve yazı tipinin belirilendiği bir tex dosyası oluşturulmuştur. Bundan sonra oluşturulan bu dosyanın sonuna -her karakter için aynı şekilde tanımlanmış olan- karakterin boş bir sayfanın sol alt köşesine yazılmasını sağlayacak olan komutların bulunduğu "texanaparca" dosyası eklenmektedir.

cat aratexdosyasi texanaparca > fontdos.tex

Bu işlem .dvi dosyasını diğer grafik dosyası formatına çevirmek için kullanılacak olan "ghostscript" derleyicisi okuma işlemine dosyanın sol alt köşesinden başladığı için yapılmaktadır. Böylece ghostscript'in okuyacağı ilk karakter font dosyasının oluşturulacağı karakter olacaktır. Texanaparca dosyası aşağıdaki komutlardan oluşmaktadır:

```
\setbox201=\hbox{\yeni j}
\setbox202=\hbox{\yeni \.I}
```

Bu parça "j" için 201 numaralı, "I" için 202 numaralı kutuların üretildiği parçadır. Bu kutular basılacak karakterin yüksekliğinin belirlenmesinde kontrol değeri olarak kullanılacaktır. "j" ve "I" harflerinin seçilmesindeki neden; "j"'nin derinliği en

büyük, "I"'nin yüksekliği en büyük olan karakter olmasıdır. Böylece basılacak diğer karakterlerin bu ikisi ile uyumlu olmaları sağlanmaya çalışılacaktır.

\newcount\yukseklik
\newcount\genislik
\newdimen\duseybosluk

"newcount" ve "newdimen" komutları seçilen yerel bir bölgede diğer ölçülerle karışıklık olmaması için istenilen tipi verilen isim ile register eder. Burada; yükseklik, genişlik ve duseyboluk adlarında yeni birimler oluşturulmuştur. Daha sonraki komutlarda bu birimlere yukarıda tanımlanan kutucukların yükseklik, genişlik değerleri atanacaktır.

\genislik=\wd200

Genişlik 200 numaralı kutunun genişliğidir. 200 numaralı kutu, karakterin içine yerleştirildiği kutudur. Böylece seçilen karakterin genişliği ne ise, "genişlik" o sayıya eşit olmaktadır. Aynı şekilde boy bilgisi;

\yukseklik=\ht202

komutu ile oluşturulur. 202, numaralı kutu "I" harfinin koyulduğu kutu idi. Yüksekliğe bu kutunun yüksekliği atanarak oluşturulacak tüm yazı tiplerinin boylarında uyumluluk sağlanılmaya çalışılmıştır.

"\global", tanımlanan özelliğin yeni bir tanımlama yapılıncaya ya da dökümanın sonuna kadar geçerli olması için kullanılır.

"\advance" ise adı verilen büyüklüğün belirtilen miktarda artırılmasını sağlar. Aşağıdaki komut; dökümanın sonuna dek geçerli olacak bir tanımlama yapmaktadır. Yükseklik değişkeninin değeri "j" harfinin derinliği kadar artırılacaktır. Bir önceki tanımlamalar hatırlanacak olursa karakter en derin harf ile en uzun harf arasındaki uzunlukta bir kutucuk içinde oluşturulacaktır. Genişliği de kendi genişliği olacaktır.

\global\advance\yukseklik by \dp201

Aşağıdaki komutlarla karakterin oluşturulmasında kullanılacak ölçü biriminin punto olması için gereken çevrim gerçekleştirilmektedir. Herhangi bir ölçübirimi belirtilmezse TeX'de "scaled point" birimi kullanılmaktadır. 65536 scaled point, 1 puntoya karşılık gelmektedir.

TeX'in anlayabildiği ölçü birimleri ve bunların TeX'deki gösterilimi aşağıdadır:

Pt	punto	(72.27 pr = 1 inc)
Pc	pika	(1 pika = 12 punto)
Bp	big point	(72 bp = 1 inc)
in	inch	
cm	santimetre	(2.54 cm = 1 inc)
mm	milimetre	(10 mm = 1 cm)
dd	didot point	(1157 dd = 1238 pt)
cc	cicero	(1 cicero = 12 didot point)
sp	scaled point	(65536 scaled point = 1 punto)

Burada genişlik ve yükseklik değerleri 65536'ya bölünerek scaled point'den punto'ya çevrilmektedir.

```
\global\divide\genislik by 65536
\global\divide\yukseklik by 65536
```

Bundan sonra verilen komutlar ilk başta sayfanın sol üst köşesinde yer alan karakterin, solda aşağılara taşınması için yapılacak işlemleri kapsamaktadır. Bunun yapılmasının sebebi "gs" kullanıldığında ilk olarak karakteri okuyup yukarıda belirlenen genişlik ve yükseklik ile oluşturulan bir kutu içine alabilmektir. "Gs" okuma işlemine sol alt köşeden başladığı için karakter oraya yerleştirilmeye çalışılmaktadır. Ancak burada yapılan işlemler işletim sistemi ve kullanılan "ghostscricpt" programının versiyonuna göre değişiklik gösterebilmektedir. Öyleki bir 3.1 Slackware sürümündeki "ghostscript", düşeyboşuk değerinin 40 pt. artırılması ile karakteri istenilen yere taşımaktadır. Ancak aynı işem farklı sürümlü slackware'deki "gs" ile artırım miktarı 80 pt olduğunda sağlanabilmektedir. Bu, yazı tipleri oluşturulurken dikkat edilmesi gereken bir konudur ve gerekirse programın derlendiği işletim sisteminin özelliklerine göre revize edilmesi gerekmektedir.

"\vsize" ile sayfa boyu belirlenir. \vsize sayfa başında kullanıldığında anlamlıdır. Sayfa belli bir büyüklükle hazırlanmış ise ikinci kez bu komut verilerek boyut değişirilemez. Sayfa boyunun belirlenmesinden sonra "düseybosluk" adında bir değişken tanımlanarak sayfaboyu büyüklüğüne getirilir. Bu büyüklük karakter basılmadan önce sayfanın üst tarafına yapıştırılacak boşluğun boyunu bulmak için kullanılacaktır.

```
\vsize=660pt
\duseybosluk=\vsize
\global\advance\duseybosluk by -\ht200
\global\advance\duseybosluk by -\dp201
\global\advance\duseybosluk by 40.0pt
```

Boşluğu yerleştirme işi de;

```
\vglue\duseybosluk
```

komutu ile yapılmaktadır.

Daha sonra bulunan genişlik ve yükseklik değerleri sonraki aşamalarda kullanılmak üzere ekrana yazdırılır. Böylece karakterin basılmasını sağlayacak TeX dosyası hazırlanmış olur.

```
\message{genislik= \the\genislik}
\message{yukseklik= \the\yukseklik}
\bye
```

Özetlenecek olursa bu aşamaya kadar üzerinde işlem yapılan karakterin TeX'de tanımlı olup olmadığı kontrol edilmiş ve karakterin görüntü dosyasını oluşturacak olan "fontdos.tex" dosyası oluşturulmuştur.

Artık sıra oluşturulan bu dosyanın derlenmesine gelmiştir. Bundan sonraki aşamalar için gerekli komutlar "tex2pcx" kabuk programı dosyasında biraya getirilmiştir. Tex2pcx programı ana program olan "fontyarat" dosyası içinden çağırılmakta ve parametre olarak font boyunu almaktadır, çünkü herbir font dosyasındaki yükseklik bilgisi font büyüklüğüne göre verilmektedir. DHTML dökümanında satır boşlukları için gerekli boyut da gözönüne alındığında her bir font dosyasının yüksekliğinin

talep edilen font yüksekliğinin iki katı olması tasarlanmıştır. Tex2pcx dosyası aşağıdaki şekilde oluşturulmuştur:

İlk olarak "fontyarat" dosyasında da olduğu gibi kabuk dosyası olduğunu gösterir komut;

```
#!/bin/bash
```

oluşturulur. Daha sonra "boy" adlı değişken tanımlanır. Boy değişkeni bu programa parametre olarak geçirilen karakter yüksekliğinin iki katı bir değere ulaşacaktır.

```
declare -i boy=2
boy=boy*$2
```

Bu da tamalandıktan sonra karakter için yaratılan TeX dosyası derlenecektir. Derleme sonucunda verilecek mesajlar yönlendirme yöntemi ile "hata" adlı dosaya yönlendirilecektir.

Kabuk da bir komutun yanına ">" işareti koyulduğununda bu o komut sonunda verilen çıktıların yönlendirme işaretinden sonraki dosyaya yazılacağı anlamındadır. Komut her çalıştırıldığında dosyadaki eski kayıtlar silinerek dosya yeniden oluşturulmaktadır. Eğer ">" yerine ">>" kullanılırsa dosyadaki eski kayıtlar saklanarak yenileri dosya sonuna eklenecektir.

```
tex $1.tex 1>hata
```

Derlenen TeX dosyaları çıktı olarak bir .log bir de .dvi dosyası üretirler. Log dosyası yapılan derleme işlemindeki sonuçların tutulduğu dosya, .dvi dosyası ise oluşturulan görüntü dosyasıdır.

Dvi çıktıları, "dvips" programı ile postscript dosyalarına çevrilebilmektedir. Postscript dosyaları "gs" gibi grafik dosyaları üzerinde çok farklı işlemler yapmayı sağlayan bir programın kullandığı dosya formatı olduğundan .dvi çıktıları postscript'e çevrilmektedir.

```
dvips -o $1.ps $1.dvi 2>/dev/null
```

Bundan sonra "gs" komutunun oluşturulacağı bir kabuk dosyası daha oluşturulmaktadır. Bu dosya oluşturulurken önce TeX çıktısında verilen mesaja bakılarak karakterin genişliği bulunmaktadır.

```
cat hata|gawk '/genislik=/ {print "gs -sDEVICE=pc9256  -dNOPAUSE \
                                    -dDEVICEWIDTH="$5"    \\" }'>arabuyruk
```

"Cat" yanındaki dosyanın içeriğinin listelenesini sağlar ancak yanında "|" operatörü olduğundan listeleme işlemi ekranda yapılmayacak, sonucu bir sonraki işleme girdi olarak gönderilecektir. Bir sonraki işlem "gawk" ile örüntü aratma işlemidir. Buna benzer bir işlem "fontyarat" dosyasının başında karakter olup olmadığının kontrolü için kullanılmıştı. Burada cat dosyasının sonucunda "genişlik" kelimesi aranmakta, var ise bir dosyaya "gs" ile genişlik bilgisinin kullanılacağı öğeler verilmektedir.

```
cat hata|gawk '/genislik=/ {print "gs -sDEVICE=pcx256  -dNOPAUSE \
                                    -dDEVICEWIDTH="$5"    \\" }'>arabuyruk
echo  -n  "-dDEVICEHEIGHT="$boy"      -sOUTPUTFILE=$1.pcx      $1.ps
1>/dev/null">>arabuyruk
chmod 755 arabuyruk
echo "quit" | ./arabuyruk
```

Oluşturulan aradosyanın çalıştırılması ile .pcx uzantılı, istenilen boyutta yeni bir görüntü dosyası oluşturulmuştur. Bu işleminde tamamlanması ile ana dosyaya dönülmekte ve dosyanın transparan bir gif dosyasına çevrilmesi sağlanmaktadır. Bu arada dosya adı da değiştirilerek ilgili yere taşınmış olur.

```
cat hata|gawk '/karakter=/ {print ""$2"0 " " "$4" " } '>> olculer
```

Olculer dosyası dökümanın HTML çıktısının hazırlanması aşamasında her bir yazı tipinin genişliğini okumak için kullanılacaktır.

```
./convert -transparency "#ffffff" fontdos.pcx "$karakterno"0.gif
mv "$karakterno"0.gif $1$2/.
```

Buraya kadar sözü geçen işlemler her bir karakter için yapılacak işlemlerdir. Tüm liste tamamlandığında TeX yazı tiplerine karşılık gelen yeni yazı tipi dosyaları oluşturulmuş ve ilgili dizinlere taşınmış olmaktadır.

3.2 TeX Dökümanın Yeni Yazı Tiplerini Kullanacak Formata Çevrilerek Yeniden Şekillendirilmesi – TEX2DHTML

Burada bahsedilen programlar, TeX komutları ile oluşturulmuş bir dökümanı, yeni yaratılan yazı tiplerini kullanır hale çevirmektedir. Ancak bu aşamada ele alınan Tex dökümanında yalnız karakterler ve matematik formüllerin olduğu varsayılarak işlem yapılmaktadır. Karakterler harf harf yeni yazı tipi formatına çevrilmekte, formüller ise bir bütün olarak TeX dosyasına taşınmakta, derlenmekte ve üretilen görüntü dosyaları son döküman içine taşınmaktadır.

Bu işlemlerin tümü veri okuma, yazma, karşılaştırma gibi işlemlere çok uygun olan Perl programlama dili ile gerçekleştirilmiştir.

3.2.1 Dökümanın Oluşturulması

Yapılan işlemlerin ilk aşaması; üzerinde çalışılacak dosyanın karakter karakter okunarak yeni fontlarda karşılığının olup olmadığının incelenmesi ve -varsa- matematik formüllerinin ayıklanılarak ayrı dosyalara yazdırılmasıdır. Bu amaçla "Cevrim" adı verilen program çalışmaktadır.

Program, bunun çalıştırılabilir bir perl dosyası olduğunu gösteren;

#!/usr/bin/perl

belirteci ile başlamaktadır.

Daha sonra bilgilerin okunacağı ve yeni karşılıkları ile yazılacağı dosyalar açılmaktadır. Bu amaçla "open" fonksiyonu kullanılmaktadır. İlk dosya okuma amacıyla açıldığından open fonksiyonuna parametre olarak yalnız dosyanın program içinde kullanılacağı adı ve dosyanın yerini gösterir ad verilmiştir. İkinci dosyaya bilgiler de yazılacağından dosyanın yer bilgisinden önce ">" işareti koyulmuştur. Dosya açma işleminde sorun olması durumunda kullanıcı hata mesajı ile uyarılacaktır. Hata kodu döndürülüp döndürülmeyeceği "unless" fonksiyonu ile

kontrol edilmektedir. Unless'den sonra parantezler içerisindeki komut "1 ya da true" sonucunu döndürür ise bir sonraki işleme devam edilecek, "0 ya da false" sonucunda "die" ile uyarı mesajı verilerek programın çalışması durdurulacaktır.

```
unless( open(IFILE,"dosya1") )
 { die ("Input dosyasi acilamiyor!\n");  }
unless( open(OFILE,">dosya2") )
 { die ("Ouput dosyasi acilamiyor!\n");  }
```

Bir sonraki adım formüllerin isimlendirilmesinde kullanılacak olan değişkenler için başlangıç değerleri verilmesidir:

```
$formul = 0;
$formul_say = 0;
```

Daha sonra girdi dosyasından bilgiler satır satır okunmakta ve okunan her satır için kontrollerin yapılması amacıyla while döngüsü oluşturulmaktadır. Döngüde kontrol edilen değer dosyadan okunmuş $satır değişkeninde bilgi olup olmadığıdır:

```
$satir = <IFILE>;
while ($satir ne "")  {
 chop($satir);
```

"Chop" ile satır sonundaki son karakter silinmektedir. Böylece yeni satıra geçiş karakterleri işleme katılmamış olur.

Satırda alınan bilgiler "split" komutu ile parçalanarak harfler dizisine atanır. Böylece dosyadan alınan her satırdaki bilgiler karakter karakter incelenmek üzere bir diziye atanmaktadır.

```
@tum_harfler = split(//, $satir);
```

Bu aşamadan sonra karakterin bir formüle ait olup olmadığı, değilse bölüm 3.1.1'de bahsedilen karakter gruplarından hangisine girdiği bulunmaktadır. Tüm bunlar if karşılaştırma komutundan yaralanarak yapılmaktadır. İlk olarak karakterin matematiksel ifadelerin yazdırılmasında kullanılan "$" karakteri olup olmadığı kontrol edilmektedir.

TeX'de matematiksel ifadeler "$ $" ya da "$$ $$" içinde yazdırılmaktadır. Aşağıda verilen program parçası bulunan matematiksel ifadenin bu iki gösterilimden hangisine uyduğuna bakmaktadır. Örneğin; $ ile başlayan bir formülizasyon için bulunan ilk $ işaretinden ikincisine kadar olan parça formülün parçaları olarak kabul edilmektedir. Bulunan her formül için bir sıra numarası verilmekte ve bu değer $formul_say adlı değişkende saklanmaktadır. Daha sonra çıktı dosyasında formül sırasına göre isimlendirilmiş çıktılar çağırılacaktır.

Görüntü dosyaları yazı stillerinde olduğu gibi hazırlanan TeX dosyasının derlenerek çıktısının .gif formatına çevrilmesi ile elde edilmektedir. Bu yüzden bulunan förmüller bir TeX dosyası formatında saklanmaktadır. Görüntü dosyasında sayfa numarası olmaması gerektiğinden TeX dosyası "\nopagenumbers" satırı ile başlatılır. Dosya sonuna da \bye komutu eklenir. Böylece kullanıcı etkileşimine gerek kalmaksızın derleme işlemi tamamlanabilecektir.

```
if ( ($harf eq "\$") || $formul ) {
    if ( $harf eq "\$")  { $formul_say++; }
    if ( ($harf eq "\$" ) && ($formul == 0) ) {
     open (FORMUL_YAZ, ">$formul_say.tex");
     print FORMUL_YAZ "\\nopagenumbers \n";
     if ( @tum_harfler[$harf_sira+1] eq "\$")  {
        print FORMUL_YAZ "\$";
        $harf_sira++; }
     print OFILE "\n $formul_say.gif \n";
    }
    if ( ($harf eq "\$" ) && ($formul == 1 ) )  {
     if ( @tum_harfler[$harf_sira+1] eq "\$")  {
        print FORMUL_YAZ " \$";
        $harf_sira++;                    }
     print FORMUL_YAZ $harf;
     print FORMUL_YAZ "\n\\end ";
     close (FORMUL_YAZ);
    }
    $formul = $formul_say % 2;
```

```
print FORMUL_YAZ $harf;
$harf_sira++;
```

Bulunan "$"karakteri değil ise özel karakter tanımlamak için kullanılan kontrol dizisi değişkeni "\" olup olmadığı kontrol edilir. Zira bu tip değişkenler için önceden tanımlan makro isimleri kullanılmalıdır. Aşğıda "ç" ve "ö" harfleri için yapılan kontroller örnek olarak verilmiştir.

```
} else {
 if ( $harf eq "\\" ) {
  if( (@tum_harfler[$harf_sira+1] eq "c")  &&
     (@tum_harfler[$harf_sira+2] eq " ")  &&
     (@tum_harfler[$harf_sira+3] eq "c") ) {
     $harf_yaz  = "cT";
     $harf_sira  = $harf_sira+4;
     print OFILE $harf_yaz;
} elsif( (@tum_harfler[$harf_sira+1] eq "\"") &&
          (@tum_harfler[$harf_sira+2] eq "o") ) {
            $harf_yaz  = "oT";
            $harf_sira  = $harf_sira+3;
            print OFILE $harf_yaz;
```

Eğer "\" ile başlayan ancak font üretme aşamasında tanımlanmayan bir karakter bulunur ise bir uyarı mesajı ile durum kullanıcıya bildirilmektedir. Tanımlanamayan karakter yerine 99 kodu ile takip edilen boşluk karakteri yerleştirilmektedir.

```
else {
  print $harf,@tum_harfler[$harf_sira+1],"Tanimlanamadi!\n";
  $harf = 99;
  print OFILE $harf;
  $harf_sira=$harf_sira+2;
 }
```

Son olarak $ ve \ dışındaki diğer karakterlerin çıktı dosyasına yazımı tamamlanmaktadır. Bu tür karakterler karakter adı yanına "0" koyularak

isimlendirildiğinden birleştirme operatörü olan "." Operatörü ile karakter adı sonuna "0" koyulmaktadır. Bu işlemden sonra ilk satırın okunması ile başlatılan while döngüsü kapatılmakta ve döngününde sonuçlanmasının ardından işlem yapılan dosyalar close komutu kullanılarak kapatılmaktadır.

```
else {
  if ( ($harf ne " ") && ($harf ne "\n") && ($harf ne "") )
    { $harf = $harf . "0"; }
    print OFILE $harf;
  }
 $satir = <IFILE>;
}
close(IFILE);
close(OFILE);
```

Böylece TeX kontrol değişkenleri ve formülleri ile oluşturulmuş bir dosya tarayıcı aracılığı ile üretilen yani yazı stillerini okuyacak formata çevrilmiştir. Bundan sonra her bir karakterin kullanılacak yazı tipinin genişlik parametreleri göz önünde tutularak ekranda gösteriliş biçiminin belirlenmesi aşaması gerçekleştirilecektir.

Bu işlemler "Sekil" adı verilen bir programda toplanmıştır. "Sekil" programı kendi içindeki kontrolleri kadar, referans ettiği kütüphanedeki altyordamları kullanmaktadır.

Altyordamlar; belli bir işlevi yapmak üzere hazırlanmış ayrı birer kod parçasıdır. Altyordamın her çağırılışında içinde yeralan komutlar çalıştırılır. Altyordamlar programların yazılmasını ve okunmasını kolaylaştıran küçük parçalara böldüğünden ve belli bir kod parçasının istenildiği kadar çalıştırılmasına olanak tanıdığından kullanışlıdır. Hazırlanan altyordamlar bir dosya içinde toplanarak farklı programlardan çağırılmasına olanak tanınır. Bu amaçla oluşturulan altyordam dosyalarına kütüphane adı verilir. Kütüphaneler .pl uzantılı dosyalardır ve kullanılacakları program içerisinden;

Require("kütüphane_adı.pl")

komutu ile referans edilirler [15].

"Sekil" programında Perl programı olduğunun gösteren ifadeden sonra "rlib.pl" kütüphanesini referans eden komut ile başlamaktadır.

```
#!/usr/bin/perl

require("rlib.pl");
```

Daha sonra bu kütüphanedeki fonksiyonlardan "datalar" altyordamı çağırılmaktadır.

```
&datalar;
```

"Datalar" altyordamı "cevrim" programı ile üretilmiş yeni fontları kullanan dosyadaki kelimelerin uzunluklarını tutmaktadır. Fakat buradaki uzunluk text ortamda her bir karakterin bir değer olarak sayıldığı uzunluk değil fontlar oluşturulurken TeX tarafından belirlenen uzunluklardır. Bilindiği gibi bu büyüklükler her bir karakterin oluşturulma aşamasında karakter adı ile bir dosyaya yazdırılmıştı.

"Datalar" altyordamının görevi yeni yazı tipleri ile oluşturulmuş dosyadan her bir kelimeyi alarak "uzunluk" adlı altyordamın yardımı ile kelimenin uzunluğunu bulmaktır. Uzunluğu bulunacak kelimeler .gif dosyalarını gösterir kelimler dışındaki kelimelerdir.

Perl'de bir altyordam içinden başka bir altyordam çağırılabilir. Aşağıda "datalar" altyordamından "uzunluk" altyordamının çağırıldığı parça görülmektedir.

```
while ($sayi_satir <= @kelimeler) {
    $kelime = @kelimeler[$sayi_satir];
    if ($kelime !~ /\.gif/) {
      &uzunluk;
      print DATAFILE $kelime_uz," ",$kelime,"\n";
    } else {
      print DATAFILE 0," " ,$kelime,"\n";
    }
}
```

Bir altyordamı çağırmak için altyordamına adının başına & işareti koyulmaktadır. Eğer altyordama geçirilecek bir parametre var ise bu da parantezler içinde belirtilir.

"Uzunluk" altyordamı bir parametre almadığından böyle bir özellik kullanılmaktadır. "Uzunluk" altyordamı aşağıdaki şekilde oluşturulmuştur:

```
sub uzunluk {
 unless(open(OLCULER,"olculer"))  {
   die ("Olculerin okunacagi dosya acilamiyor!\n"); }
 $harf_uz = $kelime_uz = 0;
   @harfler=split(//, @kelimeler[$sayi_satir]);
   $sayi_harf=0;
   while ($sayi_harf < @harfler) {
    $harf = @harfler[$sayi_harf];
    $harf = $harf . @harfler[$sayi_harf+1];
    open(OLCULER,"olculer");        #Olcu dosyasindan karsilastirma.
    $no_match = 1;
    $olcu_satir = <OLCULER>;
    while (($olcu_satir ne "") && ($no_match == 1) ) {
     @olculer = split(/\s+/, $olcu_satir);
     if ( $olculer[0] eq $harf )  {
      $harf_uz  = @olculer[1];
      $no_match = 0;
      $kelime_uz  = $kelime_uz + $harf_uz;
     }
     $olcu_satir=<OLCULER>;
    }
    if ($no_match) {
     $harf_uz  = 15;
     $kelime_uz  = $kelime_uz + $harf_uz;
    }
    close (OLCULER);
    $sayi_harf = $sayi_harf + 2;
   }
}
```

Bu altyordam adının kelimedeki her harfi "olculer" dosyasının ilk karakteri ile karşılaştırmaktadır. Eğer ölçüler dosyasında bulunan karakter aranan karakter ise "olculer" dosyasının ikincinci parametresi olan uzunluk parametresi alınmaktadır. Kelimedeki her harf için bulunan uzunluklar toplanarak kelimenin uzunluğu bulunmaktadır.

"Uzunluk" altyordamından uzunlukların alınmasının ardından ana programda görüntü dosyasının şekilendirme aşamasına gelinmektedir. Burada yapılan şekillendirme işlemi belirlenen satır uzunluğu dikkate alınarak kelimelerin çıktı dosyasına yazdırılmasıdır. Burada mantık, uzunlukları bulunan kelimelerin uzunluklarının toplanarak satır uzunluğunu geçip geçmediğini kontrol edilmesiyle oluşturulmaktır.

Eğer satır sonundaki kelime satırdan taşmaya neden oluyorsa hecelenip hecelenemeyeceği araştırılır. Ancak hecelemeye bakılmadan önce TeX'deki "tolarance" komutuna karşılık gelecek bir karşılaştırma yapılmaktadır.

Eğer satır uzunluğu, satırdaki kelimeler arasında 10'ar puntoluk boşluklar bırakıldığında tanımlanan uzunluğu geçiyor ise heceleme yapılıp yapılmaması gerektiğine karar verilir. Heceleme ancak satırdaki kelime sayısının birden fazla olması ve boşluk sayısının kelime sayısının üç katından fazla olması durumunda yapılmaktadır. Burada verilen kriterler isteğe bağlı olarak değişebilecektir.

```
if (($satir_uz + ($kelime_sira - 1) * 10) >= $def_line) {
   $satir_uz  = $satir_uz - @uzunluk[0];
   $bosluk    = $def_line - $satir_uz;
   if ( ($kelime_sira > 1 ) && ($bosluk > ($kelime_sira-1) * 3) ) {
```

Eğer kriterler sağlanıyor ise satır sonundaki kelimenin hecelenir olup olmadığı kontrol edilecektir. Bu kontrol yine hazırlanan bir altyordam ile yapılmaktadır. Altyordam kelimeye almakta ve eğer tüm karakterleri alfabetik karakterler ise Türkçe hece kurallarına göre bölünüp bölünemediğine bakılmaktadır.

Türkçe kelimelerde heceler bir, iki ya da üç harfli olabilmektedir ve her bir hece de sesli sessiz harf sıralanışı belirlidir. İki harfli hecelerde harfler sesli-sessiz ya da sessiz-sesli sıralanışında olabilir. Bu şu altyordam ile belirlenmektedir:

```
sub iki_harf {
 my($harf1,$harf2) = @_;
 if(($harf1 eq 1 && $harf2 eq 0 ) || ($harf1 eq 0 && $harf2 eq 1 ))
  { $retval_iki=1; }
 else
  { $retval_iki=0; }
}
```

Üç harfli heceler ise ancak sessiz-sesli-sessiz harflerden oluşmakatadır. Hecenin bu kurallara uyup uymadığını kontrol eden fonksiyon aşağıdadır.

```
sub uc_harf {
 my($harf1,$harf2,$harf3) = @_;
 if($harf1 eq 0 && $harf2 eq 1 && $harf3 eq 0)
  { $retval_uc = 1; }
 else
  { $retval_uc = 0; }
}
```

Her iki fonksiyonda da dönüş değeri olarak sonuç başarılı ise 1, başarısız ise 0 döndürülmektedir. Ayrıca harflerin sesli mi sessiz mi olduğu yine "rlib.pl" kütüphanesinde tanımlı olan fonksiyon ile bulunmaktadır.

Ana programdan hecelenir olup olmadığı bulunmak için "heceleme" altyordamına geçildiğinde sonuç olumlu ise her hece arasında "-" işareti olduğu halde kelime döndürülmektedir. Eğer kelime hecelenemez ise buna dair bir kod ile dönüş yapılmakta ve kalan boşluklar her kelime arasına eşit sayıda gelecek şekilde boşluk yazı tipi ile doldurulmaktadır.

Eğer kelime hecelenebilen bir kelime ise bu sefer alına ilk hece sonunda "-" işareti olacak şekilde üstteki satıra, kelimenin kalan kısmı ise alttaki satıra yapıştırılcaktır. Ayrıca her iki satıra da eklenen hecelerin uzunlukları, "-" uzunluğu gözönünde bulundurulcak şekilde, satır uzunluklarına eklenmekterir. Daha sonra varsa kalan boşluk hesaplanarak kelimeler arasına boşluklar serpiştirilir:

```
@satir_dizi = split(/\s/, $satir_yaz);
```

```
$count = 0;     $i = 0;
  while ($count <= $bosluk ) {
    if ($i > $kelime_sira-1)    { $i=0; }
    $satir_dizi =  @satir_dizi[$i];
    $satir_dizi =  $satir_dizi . "99" ;
    @satir_dizi[$i] =  $satir_dizi;
    $count++;
    $i++;
   }  #while sonu..
   print OFILE @satir_dizi,"\n";
   if ( $formul_var == 1 ) {
    print OFILE $formul_yaz,"\n";
    $formul_var=0;
   }
   $satir_yaz ="";
   $kelime_sira = 0;
   $satir_uz = 0;
   $satir_uz = $satir_uz + @uzunluk[0];
   $satir_yaz = $satir_yaz . @uzunluk[1] . " " ;
```

Tüm satırar bu şekilde oluşturulduğunda, internette çıktı olarak basılacak hali ile bir dosyaya yazılmış, yeni yazı tiplerini kullanan döküman oluşmuş olmaktadır.

3.3 Dinamik HTML Dökümanının Hazırlanması

TeX ile oluşturulmuş dosya yeni yazı tipleri belirteçlerini kullanır hale getirilip, formatı belirlendikten sonra son olarak dosyanın tarayıcı aracılığı ile görüntülenebilecek bir HTML dökümanı haline getirilmesi aşaması gerçekleştirilmektedir. Bu amaçla Perl ile yazılmış bir altyordam ve bir kabuk komutu çalıştırılmakta ve daha önceki adımlarda elde edilen text dosyası JavaScript kodlu bir dinamik HTML dosyasına çevirmektedir.

3.3.1 Dinamik HTML Nedir?

Dinamik HTML, görselyöre üzerinde dinamik nesneler tanımlamaya olanak tanıyan yeni bir teknolojidir. Eskiden, istemci tarafında yüklenilmiş bir sayfa, sunumcu tarafına gidip yenilenmeden değiştirilemezdi. Yani sayfa üzerindeki metin, yazı tipi, renk, büyüklük gibi kullanılan görsel öğeler sayfa bir kere yüklendiğinde tekrar yüklenmeden değiştirilemezdi. Dinamik HTML yüklenen bir sayfanın sunucu tarafına gitmeksizin değişmesine olanak tanıyan bir teknolojidir. DHTML kullanılarak sayfa üzerindeki bir grafiğin farklı bir yere taşınması ya da yazılmış bir yazının farklı renge dönüşmesi gibi özelliklerle sayfanın görünümün değişmesi mümkün olmaktadır. Ancak bu oldukça yeni bir uygulama alanı olduğunda tüm tarayıcılar DHTML dilleri algılayıp, bu programları çalıştırmamaktadır. DHTML'ye destek veren en sık kullanılan tarayıcılar; 4 ve üstü sürüme sahip Netscape ile Internet Explorer'dır.

DHTML script dili, DHTML özelliklerinin görselyörede uygulanırlığını sağlayan script dilidir. En çok kullanılan script dilleri JavaScript ve Visual Basic'dir. Bu çalışmada script dili olarak JavaScript kullanılmıştır.

3.3.2 JavaScript Nedir?

Javascript, HTML dosyaları oluşumunda kullanılan nesne tabanlı, yorumlanan bir script dilidir. Javascript kullanarak, sunumcu tarafındaki CGI'lara gerek kalmaksızın istemci tarafında kullanıcı, görselyöresi için kendi kontrol nesnelerini oluşturabilir.

Javascript, ilk olarak Netscape firması tarafından Livescript adı ile geliştirilmeye başlamıştır. Daha sonra 1985 yılında geliştiriciler takımına Sun çalışanlarının da katılımından sonra adı JavaSc7ript olarak değiştirilmiştir.

Javascript, Java ve C++ gibi dillerde olan nesne tabanlı özelliklere sahiptir. Kendi içinde nesneler, nesnelerin özellikler ve metodları hiyararşisine sahiptir. Ayrıca istenirse kullanıcı kendi nesnelerini ve metodlarını tanımlayabilmektedir. Javascript yorumlaman (interpreted) bir dildir. Bu yüzden çalıştırılması için derleyiciler, debugger'lar gibi birtakım yardımcı programların kullanılmasına gerek yoktur. Hazırlanan DHTML dosyası browser'da görüntülendiğinde javascript kodu da yorumlanarak çalıştırılır [11].

3.3.3 HTML Dökümanları İçinden JavaScript Komutları Çalıştırmak

JavaScript dili HTML dosyaları içinde herhangi br derleme işlemine gerek kalmaksızın çalıştırılabilen bir dildir.

Her JavaScript programında şu iki etiketin olması gerekmektedir:

<SCRIPT> ..... </SCRIPT>

İlk etiket JavaScript kodları bölümünü başlatmak, ikincisi ise sonlandırmak için kullanılır. JavaScript komutları bu iki etiket arasında yer alır.

<SCRIPT> etiketi LANGUAGE tanımı ile de kullanılabilir.

<SCRIPT LANGUAGE="Javascript">

Aşağıda basit bir JavaScript örnek dosyası yeralmaktadır:

```
<HTML>
<HEAD>
<TITLE>  JAVASCRIPT Örnek Dosyası </TITLE>
<SCRIPT LANGUAGE="Javascript">
<! –
document.write ("Henüz HTML kısmı başlamadı!");
// -- >
</SCRIPT>
</HEAD>
<BODY>
<H3> Bu bir Javascript örnek pogramıdır!! </H3>
</BODY>
</HTML>
```

Burada <SCRIPT> </SCRIPT> etiketleri dşında kalan kısım sıradan bir HTML dosyası oluşturulma kısmıdır. Bu etiketler arasında JavaScript kodu oluşturulmuştur. <! – // -- > belirteçleri, arada kalan kısmın JavaScript desteklemeyen tarayıcılarda normal bir text olarak algılanmasını sağlar.

Bu dosya ekranda bir satırlık bir çıktı üretecektir.

3.3.4 TeX'te Oluşturulmuş Dökümanın DHTML Çıktısının Hazırlanması

TeX dökümanının yeniden şekillendirilmesinin anlatıldığı ikinci bölümde kelimelerin görüntü dosyalarındaki uzunlukları dikkate alarak satırlara bölünmesi aşamasında "dosya2" adı ile bir ara dosya oluşturulmuş idi.

"Javaya_Hazırla" adı verilen bu altyordam "dosya2" dosyasından bilgileri satır satır okuyarak başına ve sonuna "satır_yaz" fonksiyonunun çalışabilmesi için gerekli argümanları eklemektedir. Ayrıca satırda okunan bilgi bir grafik dosyasına karşılık geliyor ise grafik dosyası için gerekli yazım şekli oluşturulmaktadır.

"Javaya_hazırla" altyordamı şu şekilde oluşturulmuştur:

```
sub javaya_hazirla{
 unless(open(SFILE,"dosya3"))  {
  die ("Olculerin okunacagi dosya acilamiyor!\n"); }
 unless(open(OFILE,">dosyaHTML"))  {
   die ("dosyaHTML dosyasi acilamiyor!\n"); }
   $yaz_satir = <SFILE>;
   while ($yaz_satir ne "")  {
    chop($yaz_satir);
    if ( $yaz_satir =~ /\.gif/) {
     print OFILE "\<img\>",$yaz_satir,"\<\\img\>\n";
    } else {
      print OFILE "satiryaz(\"",$yaz_satir,"\");\n";
    }
    $yaz_satir=<SFILE>;
   }
   close (SFILE);
   close (OFILE);
}
```

Bu bölümde de şu ana kadar kullanılan komutlardan farklı bir komut kullanılmaksızın Perl programı ile HTML dökümanı hazırlanmaktadır.

Ancak her HTML dosyasının başına ve sonunda bulunması gereken bazı etiketler vardır. Bu etiketleri de ilgili yerlere yapıştırmak için önceden hazıranmış iki dosya, yeni oluşturulan dosyanın başına ve sonuna eklenmektedir. Bu dosyalarda şu bilgiler bulunmakadır.

Dosyanın önüne eklenen HTML komutları:

```
<HTML>
<BODY BGCOLOR=ffffff>
<SCRIPT>
function satiryaz(a){
document.write("<NOBR>");
for(var i=0; i<=a.length-1; i=i+2){
var b=a.substring(i,i+2);
document.write("<IMG SRC=/rahsan/rfont/cmr20/"+b+".gif>")};
document.write("</NOBR>");
document.write("<BR>");
}
```

Burada önce HTML dosyası başlatılıyor, arka planın beyaz rente olaması istenilen BODY etiketi kullanılıyor ve JavaScript tanımı kısmına geçilerek, dinamik yazı tiplerini okuyarak dökümanı yazdıracak fonksiyonun tanımı yapılıyor.

Dosyanın sonuna eklenen HTML komutları:

```
</SCRIPT>
</BODY>
</HTML>
```

Bu parça ile dökümanın başında açılan ve sonunda kapatılması gereken bazı etiketler kapatılıyor.

3.4 Yapılan Çalışmaya Örnekler

Bu bölüme kadar anlatılan çalışmalar yazı tipleri oluşumundan başlamakta DHTML dökümanı oluşumuna dek devam etmektedir. Aşağıda yalnız TeX karakter gösterilimleri dikkate alınarak oluşturulmuş bir döküman bulunmaktadır:

\.ISTANBUL TEKN\.IK \"UN\.IVERS\.ITES\.I ve FEN B\.IL\.IMLER\.I ENST\.IT\"US\"U

Bu \c cal\i \c smada\, yeni yaz\i tipleri olu\c sturularak bunlar\i kullanan TeX d\"ok\"umanlar\i yaratmak hedeflenmi\c stir\! Tabii ki bu \c cal\i \c smada \"\u g\"\, \"\"O\" gibi karakterler ile rakamlar 0123456789 \, * \, \$\,\&\, \% bas\i labilmekte heceleme yap\i labilmektedir\.

Bu \"ornek 35 puntoluk yaz\i karakterleri kullan\i larak haz\i rlanm\i \c st\i r\!

Bu dosya üzerinde öncelikle -karakterlein yeni yazı tiplerinde karşılığını bulmak üzere- "Cevrim" programı çalışmaktadır. Daha sonra hecelemeler ve satır boylarınında ayarlanması için "Sekil" programı çalıştırılır. Şekil programının çıktıları şunlardır:

Her kelimenin uzunluğunun 35 puntoluk karakter uzunlukları dikkate alınarak oluşturulduğu "datalar" dosyası. Bu dosyada kelimeleri oluşturan her iki karakter o karakterin yeni yazı tiplerindeki adına karşılık gelen yeni adıdır:

179 ITS0T0A0N0B0U0L0

140 T0E0K0N0ITK0

248 UTN0ITV0E0R0S0ITT0E0S0IT

33 v0e0

71 F0E0N0

182 B0ITL0ITM0L0E0R0IT

244 E0N0S0T0ITT0UTS0UTB0u0

154 cTa0l0iTsTm0a0d0a044

61 y0e0n0i0

59 y0a0z0iT

87 t0i0p0l0e0r0i0

196 o0l0u0sTt0u0r0u0l0a0r0a0k0

105 b0u0n0l0a0r0iT

193 k0u0l0l0a0n0a0n0T0e0X0

186 d0oTk0uTm0a0n0l0a0r0iT

142 y0a0r0a0t0m0a0k0

216 h0e0d0e0f0l0e0n0m0i0sTt0i0r033

79 T0a0b0i0i0

27 k0i0

38 b0u0

205 cTa0l0iTsTm0a0d0a034gT3444

61 34OT34

54 g0i0b0i0

161 k0a0r0a0k0t0e0r0l0e0r0

33 i0l0e0

133 r0a0k0a0m0l0a0r0

170 00102030405060708090

9 44

17 42

9 44

62 36443844

29 37

211 b0a0s0iTl0a0b0i0l0m0e0k0t0e0

132 h0e0c0e0l0e0m0e0

309 y0a0p0iTl0a0b0i0l0m0e0k0t0e0d0i0r046B0u0

82 oTr0n0e0k0

34 3050

133 p0u0n0t0o0l0u0k0

59 y0a0z0iT

170 k0a0r0a0k0t0e0r0l0e0r0i0

174 k0u0l0l0a0n0iTl0a0r0a0k0

213 h0a0z0iTr0l0a0n0m0iTsTt0iTr033

"Sekil" programının ikinci bir çıktısı "Datalar"'ın satır uzunluğuna göre son halini aldığı dosyadır:

ITS0T0A0N0B0U0L0999999999999999999999999999999999999T0E0K0N0I
TK0999999999999999999999999999999999999999UTN0ITV0E0R0S0ITT0E0S0
IT99999999999999999999999999999999999999v0e0999999999999999999999
99999999999999999999999999999999999999999999999999999999
F0E0N09999B0ITL0ITM0L0E0R0IT99E0N0S0T0ITT0UTS0UTB0u099cTa0l0iTs
Tm0a0d0a04499y0e045
n0i0y0a0z0iTt0i0p0l0e0r0i0o0l0u0sTt0u0r0u0l0a0r0a0k0b0u0n0l0a0r0iTk0u0l0l0a
0n0a0n0T0e0X0d0oT45
k0uTm0a0n0l0a0r0iT99999999999999999999y0a0r0a0t0m0a0k0999999999999999
99999h0e0d0e0f0l0e0n0m0i0sTt0i0r033999999999999999999999T0a0b0i0i0999999
99999999999999k0i0999999999999999999b0u0
cTa0l0iTsTm0a0d0a034gT3444999999999999999999999934OT349999999999999
999999999g0i0b0i0999999999999999999999k0a0r0a0k0t0e0r0l0e0r09999999999
9999999999999i0l0e0999999999999999999999r0a0k0a0m0l0a0r0
0010203040506070809099999999999999999944999999999999999994299999999
99999999994499999999999999999936443844999999999999999993799999999999
99999999b0a0s0iTl0a0b0i0l0m0e0k0t0e0999999999999999h0e0c0e0l0e0m0e0
y0a0p0iTl0a0b0i0l0m0e0k0t0e0d0i0r046B0u09999999999999999oTr0n0e0k099999
99999999993050999999999999999p0u0n0t0o0l0u0k099999999999999y0a0z0iT99
999999999999k0a045
r0a0k0t0e0r0l0e0r0i0999999k0u0l0l0a0n0iTl0a0r0a0k0999999h0a0z0iTr0l0a0n0m0
iTsTt0iTr033999999999999

Bu dosya en son olarak bölüm 3.3 'de bahsedildiği gibi bir tarayıcı aracılığı ile görüntülecek HTML dosyası haline getirilmektedir:

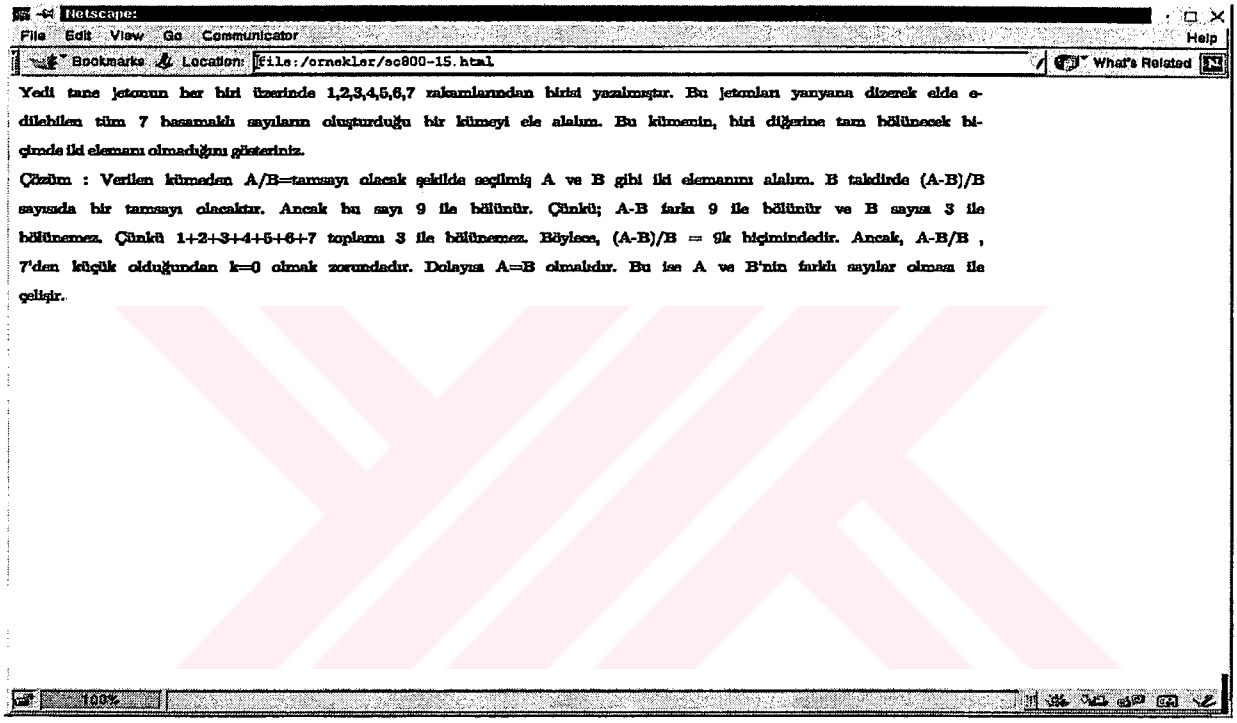
<HTML>
<BODY BGCOLOR=ffffff>
<SCRIPT>
function satiryaz(a){

```
document.write("<NOBR>");
for(var i=0; i<=a.length-1; i=i+2){
var b=a.substring(i,i+2);
document.write("<IMG SRC=/rahsan/rfont/cmr35/"+b+".gif>")};
document.write("</NOBR>");
document.write("<BR>");
}
satiryaz("ITS0T0A0N0B0U0L0999999999999999999999999999999999999999T0
E0K0N0ITK0999999999999999999999999999999999999UTN0ITV0E0R0S0I
TT0E0S0IT9999999999999999999999999999999999999v0e099999999999999
9999999999999999999999999999999999999999999999999999999999999999");
satiryaz("F0E0N09999B0ITL0ITM0L0E0R0IT99E0N0S0T0ITT0UTS0UTB0u099c
Ta0l0iTsTm0a0d0a04499y0e045");
satiryaz("n0i0y0a0z0iTt0i0p0l0e0r0i0o0l0u0sTt0u0r0u0l0a0r0a0k0b0u0n0l0a0r0iTk
0u0l0l0a0n0a0n0T0e0X0d0oT45");
satiryaz("k0uTm0a0n0l0a0r0iT9999999999999999999y0a0r0a0t0m0a0k09999999
9999999999999h0e0d0e0f0l0e0n0m0i0sTt0i0r033999999999999999999T0a0b0i0
i0999999999999999999k0i0999999999999999b0u0");
satiryaz("cTa0l0iTsTm0a0d0a034gT34449999999999999999999934OT34999999
9999999999999g0i0b0i0999999999999999999999k0a0r0a0k0t0e0r0l0e0r0999
99999999999999999i0l0e0999999999999999999r0a0k0a0m0l0a0r0");
satiryaz("00102030405060708090999999999999999994499999999999999994 2
99999999999999999449999999999999999936443844999999999999999993799
9999999999999999b0a0s0iTl0a0b0i0l0m0e0k0t0e09999999999999999h0e0c0e0l0e
0m0e0");
satiryaz("y0a0p0iTl0a0b0i0l0m0e0k0t0e0d0i0r046B0u09999999999999999oTr0n0e
0k0999999999999999930509999999999999999p0u0n0t0o0l0u0k099999999999999y0
a0z0iT99999999999999k0a045");
satiryaz("r0a0k0t0e0r0l0e0r0i0999999k0u0l0l0a0n0iTl0a0r0a0k0999999h0a0z0iTr0
l0a0n0m0iTsTt0iTr033999999999999");
</SCRIPT>
</BODY>
```

</HTML>

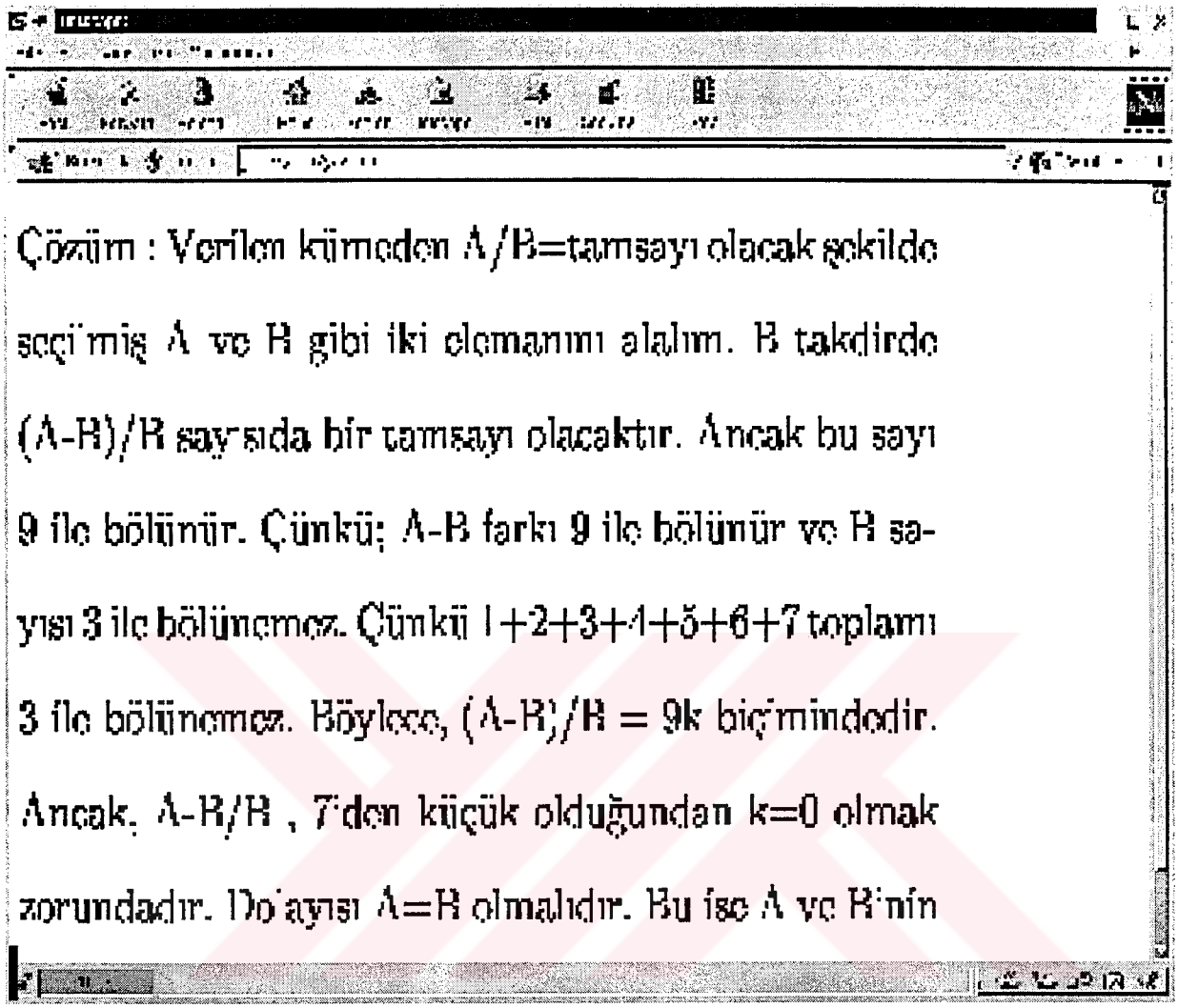
3.4.1 HTML Çıktılarına Örnekler

Şe,kil 3.2'de çıktıdaki satır uzunlukları 800 punto olarak tasarlanan bir dökümanın 15 punto büyüklüğündeki yazı takımı kullanılarak oluşturulmuş örneği görülmektedir.



Şekil 3.2 800 punto uzunluğundaki satıra, 15 punto'luk harfler ile yazılmış örnek.

Şekil 3.3'de ise çıktıdaki satır uzunlukları 800 punto olarak tasarlanan bir dökümanın 35 punto büyüklüğündeki yazı takımı kullanılarak oluşturulmuş örneği görülmektedir.



Şekil 3.3 800 punto uzunluğundaki satıra, 35 punto'luk harfler ile yazılmış örnek

SONUÇLAR ve TARTIŞMA

Bu çalışma ile, İTÜ'de yürütülmekte olan Sanal Ortamda Eğitim Projesi kapsamındaki kaynak dökümanı olarak hazırlanmış TeX dökümanlarının internete açılmasını sağlayamak amacıyla TeX tabanlı dinamik yazı tipi dosyaları oluşturulması ve TeX dökümanınan HTML çıktısı üretecek bir programlar oluşturulması hedeflenmiş idi. Çalışma sonunda belirlenen fonksiyonları yerine getiren bir yazılım paketi hazırlanmıştır.

İnternet; gerek teknolojinin çok hızlı gelişmesi ve değişmesi, gerek kullanım alanının genişliği nedeni ile çok farklı mimarilerin olması sebeplerinden standart ürünler üretilmesi zor bir ortamdır. Bu çalışmada temel amaçlardan biri özellikle yazı tipleri kullanımında bu satandartı sağlamaya çalışmak olmuştur ve üretilen görüntü dosyaları sayesinde bu sağlanmıştır.

Bununla beraber yazı tiplerinin oluşturulması aşamasında yaralanılan yazılımlardan GhostScript, farklı versiyonlarında farklı çıktılar üretebilen bir yazılımdır. Bu nedenle yazı tiplerinin üretimi aşmasında –kullanımı değil- farklı platformlarda kullanılan parametrelerden bazıları üzerinde değişiklik yapılması gerekebilmektedir. Bu çalışmanın devamında programa eklenebilecek yeteneklerden biri GhostScript'in versiyonlarını dikkate alarak parametre üretilmesi olabilir.

TeX çıktılarının DHtml'ye çevrilmesi amacıyla hazırlanmış programlar karakterler ve matematik formüllerinden oluşmuş bir dosya ile başarılı bir şekilde çalışmaktadır. Özellikle, çıktılarda üretilen dinamik yazı tiplerinin kullanması, kullanıcı etkileşimi gerekmesizin Türkçe heceleme yapabilmesi, çıktı dosyasının istenilen satır uzunluğu ile formatlanabilmesi çalışmayı mevcut ihtiyacı karşılayan bir konuma getirmektedir. Ancak TeX'in kelime işlem konusunda çok yetenekli bir dil olduğu ve pek çok komutu olduğu düşünülürse bu konuda daha yapılacaklar vardır.

Hazırlanılan dökümanın internet ortamında görüntülenme aşamasında istenilirse değişik yöntemler uygulanabilir. Mevcut yapıya ilk ilave farklı özellikteki yazı tiplerini bir dökümanda toplamak olabilir.

Yazılım geliştirme aşamasında ilk olarak Perl yerine C kullanılmıştır. Ancak yoğun olarak dosyalar üzerinde işlem yapıldığından, C'nin amaç için pratik bir çözüm olmadığı gözlenmiştir ve çalışmalara Perl ile devam edilmiştir. Perl'ün katarlar üzerinde işlem yapmada, örüntü aratmada etkin olduğu gözlemlenmiştir.

Perl ve kabuk programlama dilleri yaygın olarak kullanılan diller olduğundan ve uygulama geliştirme aşamasında esnek bir yapı kurulduğundan paket, diğer kulanıcılar tarafından da geliştirilmeye açıktır.

Bu çalışma, dinamik yazı tipi üretiminde yeni bir adım, TeX'in DHtml'e çevrilmesinde iyi bir başlangıç olarak değerlendirilebilir ve eklenecek yeni fonksiyonlar ile etkinliği artırılabilir.

KAYNAKLAR

[1] **www.seahorse.com/dynamic.html,** Seahorse

[2] **www.truedoc.com,** Bitstream

[3] **www.vietfederation.ca/index_d.html,** Viet Federation

[4] **www.hexmac.com**, HexMac

[5] **www.ke.nl**

[6] **www.hexmac.de/bitstream,** Hex Mac

[7] **Kavas, E.**, 2000. Dinamik ve kontrollü bir görsel yöre sunucusu tasarımı, *Yüksek Lisans Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[8] **http://hutchinson.belmont.ma.us/tth,** TTH

[9] **http://www.integretechpub.com/webmath,** Intertechpub

[10] **Abrahams, Paul W. with Berry K. and Hargreaves K.,** 1990. TeX for the Impatient, Addison Westley Publishing Company, USA.

[11] **Winsor, J. and Brayn F.,** 1997. Jumping JavaScript, Sun Microsystems Press, California

[12] **http://cbl.leeds.ac.uk/nicos,** LateXToHTML

[13] **Till, D.,** 1996. Teach Yourself Perl 5 in 21 Days, Sams Publishing, Indiana

[14] **Knuth, D. E.,** 1990. The TeXbook, Addison Westley Publishing Comp., USA.

[15] **http://perl.com,** Perl

Ek A

PERL Programlama Dili

Perl, text dosyalarını okuyan, text dosyalarından bilgi çeken çektiği bilgi üzerinden bir takım raporlamalar yapan programlama dilidir [1]. Unix işletim sistemi için Larry Wall tarafından oluşturulan dil C, awk, sed, sh ve buna benzer dillerin en iyi özelliklerini kendinde toplamaktadır.

Perl ismi "Practical Extraction and Report Language" kelimelerin ilk harfleri kullanılarak oluşturulmuştur. Dilbilgisi kuralları açısından da C dili ile benzerlikler göstermektedir [13].

PERL Programlama Dilinin Özellikleri

Değişkenler

Değişkeler bilgi depolayabilen değerlerdir. Depolanan bilgi bir nümerik ya da alfanümerik tek bir karakterden bir satırlık bilgiye kadar herşey olabilir. Perl'de bilgi saklama açısından bir limit yoktur, makinenin belleği yettiği süreci istenilen büyüklükte bilgi değişkenlerde saklanabilir.

Değişkenler tamsayı, gerçel sayı, karakter ya da katar olsun önüne $ imi koyularak adlandırılırlar. Adlandırılmada küçük büyük harf kullanımı etkilidir. Değişken isimleri istenilen uzunlukta olabilir ancak içinde "!,." bulunduramaz ve "_" ile başlayamaz.

Değişkenler için program içinde herhangi bir ön tanımlama yapmak gerekmez, "=" imi kulanılarak istenilen değer atanabilir.

```
$yıl = 2000;
$isim = "PERL Programlama Dili";
$uzaklık = 3.5;
```

Listeler ve dizi değişkenleri

Diziler her türlü değeri içerebilen bilgi topluluklarıdır. Aynı anda bir tamsayı, bir gerçel sayı ve bir değişken aynı dizinin elemanları olabilir. Dizi adlandırmı değişken adlandırımı ile aynı kurallara tabidir yalnız tanımlaması önüne "@" imi koyularak yapılır.

@ornek_dizi = (1, "kelime", $isim, "1 2 3")

Perl'de dizinin ilk elemanı sıfırıncı eleman olarak adlandırılır. Belli bir eleman seçilmek istenir ise sırası belirtilerek seçilebilir. Örneğin;

@ornek_dizi [1] = 1, @ornek_dizi [2] = "kelime", @ornek_dizi [3] = $isim

sonuçlarını verecektir.

Operatörler

Operatörler, yanlarındaki operandlar üzerinde belirli işlemleri yapmayı sağlarlar. En temel operatör "=" imi ile gösterilen atama operatörüdür. Belli başlı operatörler sıralanmak istenirse:

Aritmetik İşlem Operatörleri:

Toplama : $x + $y
Çıkarma : $x - $y
Çarpma : $x * $y
Bölme : $x / $y
Mod Alma : $x % $y

Dizin Karşılaştırma Operatörleri (Bunlara karşılık gelen nümerik operatörler "()" imleri içinde belirtilmiştir.)

Küçüktür : $x lt (<) $y
Büyüktür : $x gt (>) $y
Eşittir : $x eq (= =) $y
Küçük ve eşit : $x le (<=) $y
Büyük ve eşit : $x ge (>=) $y

Eşit değil : $x ne (!=) $y

Mantıksal Karşılaştırma Operatörleri

Ve : $x && $y

Ve ya : $x || $y

Değilini alma : ! $x

Dizin Birleştirme Operatörü

"." Operatörü iki dizini birleştirerek yeni bir dizin oluşturur.

$dizin = "kapı" . "kule";

Kontrol deyimleri

Perl programlama dilinde bir takım koşuların sağlanıp sağlanmadığının kontorlü amacı ile "if" deyimi kullanılır. Bu deyim ile istenilen koşul sağlandığ sürece belirtilen işlemlerin yapılması sağlanır. If deyimi iki bölümden oluşur; ilk bölüm "()" imleri arasına yazılan koşul, ikinci bölüm ise "{}" imleri arasına yazılan deyim bloğudur. Koşul bloğunda 1.1.3 bloğunda sözü geçen operatörler kullanılarak istenilen konroller yapılabilir.

```
if ( $kontrol_değeri > 5 ) {
  print "Koşul sağlanıyor";
} else { print "Koşul sağlanmadı."; }
```

Else ile belirlenen blokta koşul sağlanmadığında çalıştırılacak komutlar yer alır.

Döngüler

Döngü, bir deyim bloğunu bir çok kez yineleyerek işleme sokmaya yarar. Böylece birçek kere tekrarlanacak işlemler tek bir seferde gösterilmiş olur.

Perl'de en sık kullanılan iki döngü "while" ve "for" döngüleridir:

While döngüsü; while deyiminden sonra "()" imleri içinde belirtilen koşul sağlandığı sürece "{}" imleri arasına yazılan komut bloğunu çalıştırmak sureti ile döngüyü sağlar. Koşul sağlanmadığı takdirde "}" iminden sonraki komutlara geçilir.

```
$i = 0;
while ( $i < 10 ) {
 print $i;
 $i++;
}
```

Yukarıdaki örnekte 0-10 arasındaki sayılar ekrana yazırılmaktadır.

Aynı örnek for döngüsü kullanılarak şu şekilde oluşturulabilir:

```
for ($i =0; $i < 10;  $i++;) {  print $i; }
```

For döngüsünde "()" imleri arasında yer alan birinci deyim başlangıç deyimidir; yanız döngü çalıştılmadan önce bir kere çalıştırılır. İkinci deyim döngüyü sonandıracak olan deyimdir. Üçüncü deyim ise döngüdeki son deyim her çalıştığında bir kere çalıştırılır. "{}" imleri arasına yazılan komut bloğu ikinci deyim sağlandığı sürece çalıştırılacakları gösteren komut bloğudur.

For ve while dışında özellikle while ile çok benzer bir yapıda olan until döngüsü kullanılmaktadır. Until'de "()" imleri arasında yer alan koşul sağlanıncaya kadar "{}" imleri arasına yazılan komut bloğu arasında yazılan komutlar çalıştırılır.

```
$i = 0;
until ( $i > 9 ) {
 print $i;
 $i++;
}
```

Dosyalama işlemleri

Üzerinde işlem yapılan datalar bir dosyada saklanmadığı sürece yalnız programın çalışması sırasında kullanılabilen datalar olacaktır. Dataların program aracılığı ile saklanması amacı ile dosyalar kullanılır.

Bir dosya ile işlem yapabilmek için ilk olarak dosyayı açmak gerekecektir. "Open" komutu işletim sistemine "()" imleri içinde belirtilen dosyaya erişildiğini dolayısı ile üzerinde başka işlem yapılmaması gerektiğini söyler. Bu komutun iki argümanı vardır. Bunlardan ilki açılan dosyanın program içinde kullanıacak değişken adını ikinicisi dosyanın makinedeki yerini göstermektedir.

Open(INFILE, "/usr/local/deneme")

Dosyalar okuma, yazma ve append olmak üzere üç modda açılabilirler. Okuma modu için herhangi özel belirtece ihtiyaç yoktur, belirtilen dosyadan yalnız okuma amaçlı data çekilebileceğini belirtir. Dosya açılışı için benimsenmiş değer okuma modunda açılmasıdır. Yazma modu için dosya yerinden önce ">" operatörü kullanılır. Yazma modunda açılan dosya her seferinde yeniden oluşturulur. Append modda ise eski değerler saklanarak, yeni gelen değerler dosya sonuna eklenir. Append mod için dosya yerinden önce ">>" operatörü kullanılır.

Dosyadan değer okuma işlemi "<>" imleri arasına yazılan dosya adının bir değişkene atanması ise gerçekleştirilir.

$satır = <TAKIMLAR>;

komutu ile TAKIMLAR adlı dosyadaki ilk satır bir katar olarak $satır değişkenine atanır. Aynı işlem;

@satır = <TAKIMLAR>;

komutu ile yapılırsa dosyadaki her satır dizinin bir elemanı olacaktır.

Dosyalara değer yazmak için standart çıktıya değer yazdırmak için kullanılan print komutu kullanılır.

print TAKIMLAR "ankaragücü";

Dosyalar üzerinde yapılan çalışmalar tamamlandığında dosya "close" komutu ile kapatılmalıdır. Programın çalışması sonlandığında kapatılmamış dosyalar var ise Perl otomatik olarak kapatmaktadır.

Close (TAKIMLAR);

Fonksiyonlar

Fonksiyonlar daha önceden tanımları yapılmış ve isimleri birtakım parametreler de kulanılarak program içerisinde çağırılarak çalıştırılan altyordamlardır.

En sık kullanılan fonksiyonlara örnekler:

Chop($skaler_değişken) : Bu fonksiyon adı belirtilen skalar değişkenin son karakterini silmye yarar. Özellikle bir dosyadan satır satır bilgi çekme işleminde satır sonundaki basılabili olmayan yeni satır karakterlerini sildirmekte kullanılmaktadır. Örneğin;

```
open (IFILE, "deneme " )
$satır = <IFILE>
chop ($satır);
while ($satır ne "") {
  print $satır;
}
$satır = <IFILE>
```

programında satır "deneme" dosyasından okunan her kaydın son karakteri silinerek ekrana yazılacaktır. Deneme dosyasınında "1234567890" datasının olduğu varsayılırsa, çıktı "123456789" şeklinde olacaktır.

Length($karakter_değişken) : Bu fonkiyon verilen karaker değişkenin uzunluğunu vermektedir. Örneğin;

```
Print $length("merhaba");
```

komutu çıktısı 7 olacaktır.

Split(/ayıraç/, $değişken_adı) : Split fonkiyonu adı verilen değişken içinde ayıraç karakterlerini arayarak değişkeni parçalara böler. Örneğin;

```
$kelime = "merhaba";
@harfler = split (//, $kelime);
```

komutu

merhaba = ("m" "e", "r", "h", "a", "b", "a") dizisini üretecektir. Çünkü split komutu içinde ayıraç olarak belli bir karakter belirtilmeksizin her bir karakterin ayıraç olarak kullanılabileceği belirtilmiştir. Aynı komut aşağıdaki gibi uygulandığında;

```
@harfler = split (/a/, $kelime);
```

çıktı merhaba = ("merh", "a", "b", "a") şeklinde olacaktır.

Altyordamlar

Altyordamlar; belli bir işlevi yapmak üzere hazırlanmış ayrı birer kod parçasıdır. Altyordamın her çağırılışında içinde yeralan komutlar çalıştırılır. Altyordamlar programların yazılmasını ve okunmasını kolaylaştıran küçük parçalara böldüğünden ve belli bi kod parçasının istenildiği kadar çalıştırılmasına olanak tanıdığından kullanışlıdır.

Bir altyordam "sub *altyordam_adı* { }" belirteçleri ile tanımlanır. Program içinde "&" iminden sonra altyordam adı belirtilerek çalıştırılır.

```
#!/usr/local/bin/perl
$kelime = "";
$hece = <STDIN>;
 while ($hece ne "") {
   &hece_ekle;
  $hece = <STDIN>;
 }
print $kelime;

sub hece_ekle {
my($hece) = @_;
$kelime = $kelime . $hece;
}
```

Yukarıda verilen programda klavyeden bir değer girildiği sürece girilen değerler - hece_ekle altyordamı kullanılarak- kelime sonuna eklenecektir. Program içinde "&" iminin olduğu satırlarda altyordam çağırılmaktadır.

Altyordam içinde kullanılan değişkenlerin en son aldıkları değerler program içinde de kullanılmaktadır. Yani altyordamdan dönüş değeri altyordam içinde en son alınan değerdir.

Bir altyordama değer göndermek için "my($değişken_adı) = @_ " komutu kullanılır.

Örüntü Tanıma

Örüntü; belli karakterler içinde aranan karakter dizisidir. Perl programlama dili özellikle örüntüler üzerinde işlem yapması açısından çok kuvvetlidir.

Bir karakter topluluğu içinde belli bir karakteri aramak için;

$karakter_topluluğu =~ /$aranan_karakter/;

yapısı kullanılır. Burada aranılan karakter bulunur ise 1 ya da doğru; bulunamaz ise 0 ya da yanlış değeri döndürülür.

Aynı şekilde karakterlerin olmadığının kontrölü için "=~" imleri yerine "!~" imleri kullanılır.Aranlan karakter içinde yer alan karakterlerden ;

[aA] : [] arasında yer alan karakterlerden en az birini;

* : Ne olduğuna bakılmaksızın bir ya da birden fazla karakteri;

? : Ne olduğuna bakılmaksızın bir karakteri;

+ : Herhangi uzunluktaki bir karakteri temsil eder.

Perl dosyasının derlenmesi ve çalıştırılması

Perl programları içerisine yazılan;

#!/usr/local/bin/perl

satırı Perl'ün hem derleyici hem de yorumlayıcı olarak çalışmasını sağlar.
Program yazıldıktan sonra çalıştırılması için iki yol izelnebilir. Bunlar:

1. Komut satırından

Perl *program_adı*

Komutu verilerek programın derlenmesi sağlanır.

2. Hazırlanan Perl dosyasına çalıştırılma hakları verilerek, komut satırından çalıştırılır.

Chmod 777 *program_adı*

program_adı

Her iki yöntemde de programın çalışmasında bir hata var ise hata mesajı verilerek işlem kesilir.

Ek B

TeX Kelime İşlem Dili

TeX, Donald Ervin Knuth tarafından, 1970'li yılların sonlarında ilk sürümü piyasaya çıkarılan ve 1983 yılında son haline kavuşarak bugünlere kadar gelen, bütün bilimsel ve akademik kuruluşlar tarafından rağbet gören en büyük yazılım sistemidir. Kitap ve doküman yazımında kusursuzdur.

Piyasadaki diğer kelime işlemcilerden farklı bir yapıya sahip olan TeX, bir programlama dilinin esneklikleri ile yazım özelliklerini birleştiren, tüm bilimsel formül ve sembolleri içeren, sayfa üzerindeki alana mikron düzeyinde hakimiyeti sağlayan ve daha bir çok avantaja sahip olan bir yazılım sistemidir. Bu özelliklerinden dolayı, bir programlama dili olarak da algılanabilir.

TeX'in Derlenmesi ve Çalıştırılması

TeX, diğer kelime işlemcilerden farklı olarak "ne görürsen onu elde edersin" (WYSIWYG - What you see is what you get) grubunda olmayan bir yazılım sistemidir. Bu yüzden "yazım" ve "görüntü" olmak üzere iki ayrı aşamaya sahiptir. Yazım aşaması, komut satırından ya da bir editörde yazılan dosyalar ile gerçekleştirilir. Sistemden TeX 'i çalıştırarak, yani komut satırından hemen sonra, "tex" komutu yazılarak işe başlanır. Bu işlemden sonra şöyle bir mesajla karşılaşılır:

```
This is TeX , Version 3.1415N (C version 6.1)
**
```

Burada "**", TeX 'in çalıştıracağı dosyanın isminin yazılacağı yeri simgeler. Buraya, görüntülenmek istenilen dizin yazılıp ve bir sonraki asteriskden sonra da "\end" yazarak girişin tamamlandığı belirtilirse basit bir yazım gerçekleştirmiş olur. Örneğin:

```
This is TeX , Version 3.1415N (C version 6.1)
* merhaba !
```

* \end
[1]
Output written on texput.dvi (1 page, 224 bytes).
Transcript written on texput.log.

Burada, "texput" isimli bir sayfalık bir dosya hatasız olarak oluşturulmuştur. Aynı zamanda "texput.dvi" ve "texput.log" isimli iki dosya daha oluşturulmuştur. Bunlardan "dvi" uzantılı olanı görüntü dosyası ve "log" uzantılı olanı ise bu dosya hakkında mesaj ve varsa hataların yazıldığı bir dosyadır. Böylece bir TeX dosyası oluşturulmuş olur.

Görüntü için "xdvi" kullanılır. TeX 'de görüntüler, büyük beyaz bir sayfanın içinde gösterilirler. Bu ekran sadece gözlem için kullanılır, herhangi bir düzeltme yapılamaz.

Kısa dosyalar komut satırında oluşturulabildiği gibi daha uzun dosyalar bir editör ile oluşturulurlar. Örneğin, bir doya oluşturulup "deneme.tex" adı ile saklansın. Komut satırında "tex deneme.tex" yazılıp ikinci "*"dan sonra da "\end" veya "\bye" yazarak "deneme.tex" dosyasını "deneme.dvi" dosyasına çevirmiş oluruz. "\bye" komutu editörde dosyanın sonuna da yazılabilir, bu takdirde derledikten sonra tekrar yazılması gerekmez. Aynı zamanda "deneme.log" dosyası da oluşmuştur. Oluşan "dvi" uzantılı dosya, "xdvi deneme" komutu ile görüntülenir.

Yazı Tipleri

TeX'in oldukça zengin bir t kütüphanesi bulunmaktadır. Temel TeX'de mevcut olan yazı tipleri çeşitleri ise şunlardır:

"Roman"; normal karakterdir, kontrol dizisi "\rm"dir

"Slanted"; yana yatık karakterdir, kontrol dizisi "\sl"dir.

"Italic"; italik karakterdir, kontrol dizisi "\it"dir.

"Typewriter"; daktilo karakteridir, kontrol dizisi "\tt"dir.

"Bold"; koyu karakterdir, kontrol dizisi "\bf"dir.

TeX'de herhangi bir font tanımlanmadığı sürece herşey Roman olarak basılacaktır. Eğer bir font ve hemen arkasından gelen yazı özel parantezler arasında yazılırsa font lokal olarak tanımlanmış olur, yani sadece bu alan içerisinde etkinlik kazanır. Fontlar şekillerinde olduğu gibi boylarında da çeşitlilik gösterirler. Fontları çeşitli ebatlarda yazdırabilmek için "\font" komutu kullanılır. Bu komutun ardından tanımlanan değişkenin font tipi belirtilmiş olur. Örneğin;

```
\font\ninerm=cmr9
\font\eightrm=cmr8
```

"\ninerm" değişkeni, artık "cmr9"un özelliklerine sahip olmuştur. "cmr", Computer Modern Roman'ın kısaltmasıdır. Bunları şu şekilde kullanılabilir:

```
\ninerm daha-küçük
\eightrm ve daha-küçük
```

Benzer şekilde fontları ölçeklendirmek de mümkündür. Bunun için de "scaled\magstep0", "scaled\magstep1"... "scaled\magstep5" komutları kullanılır. "magstep" in yanına yazılan 0-5 arası rakamlar 1.2 sayısının üssü kadar fontu ölçeklerler. Mesela, "scaled\magstep2" fontu 1.2^2 = 1.2*1.2 kadar ölçekli büyültür. "scaled\magstep0" fontu 1.2^0 = 1 kadar büyültür, yani fontun kendisidir. Fontlar aşağıdaki şekilde:

```
\font\normal=cmr10 scaled\magstep0
\font\bir=cmr10 scaled\magstep1
\font\iki=cmr10 scaled\magstep2
\font\uc=cmr10 scaled\magstep3
\font\dort=cmr10 scaled\magstep4
\font\bes=cmr10 scaled\magstep5
```

isim vererek kullanabileceği gibi bütün dökümanın aynı ölçekle çalışması istenirse bu da "magnification" komutu ile uygulanabilir. Mesela, metnin başına :

```
\magnification=magstep2
```

yazılırsa, bütün metin 2 derece, yani 1.2*1.2 kadar, ölçeklenir. Burada yapılanlar bütün metinde geçerli, yani global olacaktır. Eğer, yazım esnasında farklı farklı ölçeklemeler kullanılacak ise, önceki yolda gösterildiği gibi her fontun tanımlanıp bunların gruplama yapılarak kullanılması gerekmektedir.

TeX'de kelimeler ve satırlar arası boşluklar sabittir. Ama bu mesafeler genişletilebilir veya az da olsa daraltılabilir. TeX, her paragrafı bir ünite olarak yorumlar. Bu nedenle bütün sayfa boy ve enini tutturmak için yazılanları satırlara yayar. Eğer bu mümkün değilse satır sonuna gelen kelimeleri İngilizce hece sistemine göre tireler koyarak böler. Eğer bu kelimeleri uygun şekilde bölemiyorsa "overfull" mesajı verir. Bu durumda uygun heceleme kullanıcı tarafından yapılmalıdır. Heceler kullanıcı tarafından "\-" ile bölünebilir.

Standart Komutlar

"\" ile başlayan ifadeler TeX'in kontrol dizileridir, bunlara komut da denilebilir. TeX, bünyesinde 900 adet komut içerir, bunların ise 300'ü pratik olarak kullanılmaktadır.

TeX 'de herbir komut belli bir işlem için tanımlanmıştır. TeX 'de bir metin yazılırken yazı düzenini belirleyen komutlar vardır. Bu komutlardan bazıları şunlardır:

"\vskip Boy",	"Boy" kadar düşey olarak boşluk bırakır.
"\vglue Boy",	"Boy" kadar sayfa başına boşluk bırakır.
"\hskip Boy",	"Boy" kadar yatay olarak boşluk bırakır.
"\baselineskip Boy",	"Boy" kadar satırların arasını açar.
"\rightskip Boy",	"Boy" kadar sayfanın sağında boşluk bırakır.
"\leftskip Boy",	"Boy" kadar sayfanın solunda boşluk bırakır.
"\parskip Boy",	"Boy" kadar paragrafların arasını açar.
"\parindent Boy",	"Boy" kadar paragrafı içeriden başlatır.
"\hsize Boy",	sayfanın genişliğini "Boy" kadar yapar.
"\vsize Boy",	sayfanın boyunu "Boy" kadar yapar.
"\phantom Boy",	ifadenin uzunluğu kadar boşluk bırakır.
"\quad Boy",	aynı satırdaki 2 formülün arasını 10pt açar.
"\qquad",	aynı satırdaki 2 formülün arasını 20pt açar.
"\null",	bir sayfa atlatır.
"\vfill",	sayfa atlatır.
"\end",	metnin bittiğini gösterir.

Kullanıcı bu komutları kullanarak sayfa düzeni oluşturmazsa TeX kendi standart sayfa düzenini kullanır.

Aşağıdaki komutlarla da metin sayfanın istenilen yerine yazdırılabilir:

"\par {yazı}",	Yazıyı paragraf olarak başlatır.
"\centerline {yazı},	Yazıyı satırın ortasına yazar.
"\leftline {yazı}",	Yazıyı satırın soluna dayalı yazar.
"\rightline {yazı}",	Yazıyı satırın sağına dayalı yazar.
"\headline {yazı}",	Yazıyı sayfadaki üst boşluğa yazar.
"\footline {yazı}",	Yazıyı sayfadaki alt boşluğa yazar.

TeX , her sayfanın başına sayfa numarasını birden başlatarak otomatik olarak yazar. Bütün bunları ayarlamak, kullanıcının elindedir. Sayfa numarası verilmek istenmiyorsa metnin başında "\nopagenumbers" yazılmalıdır. Eğer sayfa numarası belli bir numaradan itibaren yazdırılmak istenirse, metnin başında \pageno = ($numara$) yazılmalıdır. Sayfa numarası Roman rakamı şeklinde isteniyorsa, ($numara$)'ya negatif değer verilmelidir.

Matematiksel Yazılım

TeX 'in en önemli özelliklerinden birisi de en basitten en karmaşığa kadar her türlü matematiksel ifadenin, formüllerin ve özel simgelerin, istenilen tarzda yazılmasını mümkün kılmasıdır.

TeX de matematiksel ifadelerin hepsi "$", matematiksel mod, veya "$$", gösteriş modu, işaretlerinin içinde yazılır. Mesela yazılan formülün $(x + y)$ şeklinde başında ve sonunda birer tane "$" işareti bulunmalıdır. Aksi takdirde yazılan ifade matematiksel olmaz, sıradan bir yazılım olur. Eğer matematiksel ifadelerin içinde özel semboller varsa, bunlar $ içinde yazılmadığı takdirde, TeX hata verecektir. Kısaca bütün matematiksel ifadeler $...$ veya $$ ..$$ arasında belirtilmelidir. Dikkat edilmesi gereken bir husus da her açılan $ işaretine mutlaka bir $ işareti ile sonlandırılması gerekliliğidir.

Matematiksel ifadelerin içinde bırakılan boşluklar, TeX tarafından dikkate alınmaz. TeX, aynı zamanda kitap veya dergilerde satır ortasına yazılan özel matematiksel ifadeler için gösteriş modunu geliştirmiştir. Bu modda yazılan ifadeler $$...$$ arasına alınırlar. Örneğin;

$$ \alpha, \beta, \gamma, \delta $$

şeklindeki Latin alfabesinin ilk 4 karakteri,

$\alpha, \beta, \propto, \delta$

olarak görüntülenir. Böylece ifade hem matematiksel olarak yazılır, hem de satır ortasına getirilir ve satırın üst ve altından biraz boşluk bırakılarak güzel bir görünüm elde edilir.

TeX'de üslü ifadeler yaratabilmek için matematiksel modda iken üssü alınan değer, "^" işareti ve üs yazılır. Matematiksel ifadenin içindeki boşluklar dikkate alınmamaktadır. Bu nedenle boşluk bırakılmak isteniyor ise \ kullanılmalıdır.

Üslü ifadelerde gruplama yapmak ikiden fazla ifadeyi yazmak için şarttır, mesela;

"x\^{}y\^{}z" veya "x_y_z" ifadesi hatalıdır. Doğrusu "x\^{}{ y\^{}z}" veya "x_{ y_z}" dir.

Ayrıca formül veya sayıların üst ve altlarını çizdirmek de olasıdır.

```
$\underline 4$
$x^{\underline n}$
$\overline{x+y}$
```

Matematikte sıkça kullanılan karakterlere karşılık gelen komutlar TeX'in yapısında bulunmaktadır. Bunlardan bazıları: "\alpha", "\beta", "\gamma", "\delta", "\epsilon", "\zeta", "\eta", "\theta" "\kappa", "\lambda", "\mu", "\nu", "\xi", "\rho", "\varrho", "\sigma", "\varsigma", "\tau", "\upsilon", "\psi", "\omega" komutlardır.

Makrolar

Makrolar, TeX 'in mevcut tekniklerinin en önemlilerinden birisidir. Makrolar "\def " komutu ile yeni bir kontrol dizisi tanımlanarak oluşurlar. Bunun en basit biçimi:

```
\def\makroismi{ ...}
```

komut zinciridir. Bundan sonra "\makroismi", yukarıda küme işaterleri grubu içinde tanımlandığı şekilde çalışmaya hazır bir komut olmuştur. Burada dikkat edilmesi gereken nokta, verilen ismin nümerik veya Türkçe karakter içermemesi ve yapılan işi hatırlatmasıdır. Örneğin:

```
\def\ITU{ Istanbul Teknik Üniversitesi}
```

tanımlaması, isim olarak görevine uymaktadır. Çünkü kullanıcı,

\def\ITU{İstanbul Teknik üniversitesi} makrosunu tanımlayarak

"Ben \ITU'nde okumaktayım."

şeklinde yazdığında;

"Ben Istanbul Teknik Üniversitesi'nde okumaktayım."

cümlesini görüntüleyecektir.

Her defasında artık "\ITU" komutunu, uzun bir cümle yerine kullanmak, daha kolay ve esnek olacaktır.

Bütün kontrol dizileri, komutlar ve aynı zamanda makrolardan sonra bırakılan boşluklar ihmal edilir. Bu tür komutlardan sonra boşluk isteniyorsa, komutun ardından "\" kullanılmalıdır. Makrolarda da aynı mantık geçerlidir.

Makroların başka bir kullanım şekli de standart komutların isimlerini değiştirmek için kullanılmasıdır. Bunun için;

\let\yenikomut = \eskikomut

formuyla "\let" komutu kullanılır. Örneğin:

\let\ortala = \centerline
\let\solayaz = \leftline

olarak tanımlanan yeni makrolar;

\ortala{Istanbul Teknik üniversitesi}
\solayaz{Fen-Edebiyat Fakültesi}

şekliyle esnek ve rahat bir kullanım sağlar.

EK C

HTML

HTML programları herhangi bir metin editöründe yazılabilir, örneğin Linux'da joe,vi, Dos'ta Edit, Windows'da Notepad gibi. Bu editörlerde oluşturulan HTML programları daha sonra Netscape, Mosaic ve Internet Explorer gibi tarayıcılar aracılığı ile görüntülenebilir. HTML programlarını oluşturmak için özel editörler olsa da, bu editörler kapsamlı bir sayfa yapımı söz konusu olduğunda yetersiz kalabilirler.

HTML komutları etiket (tag) adı verilen, <...> karakterleri içinde kalan ifadeler şeklindedir. <komut> yapısındaki bir HTML komutu ardından gelen ifadelerden sonra, ki bu ifadeler komutun etkilediği ifadelerdir, </komut> şeklinde sonlandırılır. Her HTML dosyası prensip olarak <HTML> etiketi ile başlar ve </HTML> etiketi ile biter. Bu yapı içerisinde başlık ve gövde olmak üzere iki alt bölüm vardır. Başlık bölümü <head> etiketi ile başlar </head> etiketi ile biter, gövde bölümü <body> etiketi ile başlar </body> etiketi ile biter. Başlık bölümü, öncelikle hizmet birimlerine döküman hakkındaki bilgileri sağlamak için kullanılır, gövde bölümü ise Web sayfasını oluşturan diğer tüm komutların yerleştirildiği ana bölümdür.

Bu bilgilerin ışığında herhangi bir HTML programının yapısı şöyle olur:

```
<HTML>
  <head>
   ...
  </head>
  <body>
   ...
  </body>
</HTML>
```

Başlık Bölümü Komutları

Başlık bölümünde kullanılan en temel komutlar şunlardır.

<title>, tarayıcı penceresinin de başlığı olarak gözükecek olan, döküman başlığını belirler.
<base>, tüm bağlı Web sayfaları için temel adres tanımlasını yapar.
<isindex>, web sayfası üzeride etkileşimli bir arama mekanizması sağlar.
<meta>, hazırlanan sayfa hakkında genel bilgi verilmesini sağlar.

Yazı Tipleri

HTML'de kullanılan yazı tipleri fiziksel ve mantıksal olmak üzere ikiye ayırılır. Fiziksel tipler, tüm tarayıcılar tarafından gösterilen, bilinen karakterlerin tanımlanmasında kullanılır. Mantıksal tipler ise her tarayıcıda farklı algılanan, kesin bir biçimi olmayan, vurgulama ve metnin tarayıcı tarafından nasıl kullanılacağını belirleyen karakterler tanımlamada kullanılır.

Yazı büyüklükleri için kullanılan HTML komutu <Hn> komutudur. Burada n, 1'den 6'ya kadar olabilecek olan bir sayıdır. <H1> en büyük , <H6> en küçük yazı tipidir.

HTML'de farklı yazı tipleri elde etmek için kullanılan diğer bir komut ise komutudur. Bu komutla birlikte yazı büyüklüğünü belirleyen "size", yazı rengini belirleyen "color" ve sadece Microsoft işletim sistemlerinde gerçerli olan "face" parametreleri kullanılabilir. komutunun kullanım yapısı şöyledir:

 ifade... .

Burada sayı en küçük 1, en büyük 7 olabilir, renk ise red(kırmızı), blue(mavi), black(siyah) gibi İngilizce ifadeler ile belirlenebileceği gibi, #08AF03 gibi 16'lık sayı sistemine karşılık gelen rgb renk sistemindeki bir sayı ile de belirlenebilir. tip ise "Arial" veya "Helvetica" gibi bir yazı tipi olabilir.

Sayfaların Resimlendirilmesi

Web sayfasına resim eklemek için kullanılan komut <img> komutudur. Resim dosyasının ismi "src" parametresi ile komuta verilir. Dosya formatı jpeg veya gif resim formatında olabilir. Resmin genişliği "width", yüksekliği "height"

parametreleri ile ayarlanır. Örneğin "1.gif" isimli bir resmi şu şekilde sayfaya ekleyebiliriz:

<img src="1.gif" height=100 width=100>

<img> komutunun </img> ile sonlandırılmadığına dikkat edilmelidir.

Bağlantılar

HTML'nin en önemli özelliklerinden biri, bir sayfayı diğer bir sayfaya veya diğer adres protokollerine bağlayan yapıları oluşturabilmesidir. Bağlantılar <a href=...> komutu ile oluşturulabilir. Bu komutun kullanım yapısı şöyledir:

 Bağlantı ifadesi .

Burada adres başka bir HTML dosyası, mail veya ftp vb. adresler olabilir. Bağlantı ifadesi' nin rengi web sayfası üzerindeki diğer yazıların renginden farklı olacaktır, ayrıca fare ile üzerine gelindiğinda fare sembolünün değiştiği de görülecektir, böylece ziyaretçi bu ifadelerin üzerinde bir bağlantı olduğunu anlayabilmektedir. Bağlantı ifadesi bir resim olabilir. Bu işlem 3.1.4 bölümünde anlatılan <img> komutunun <a href=...> komutu içinde "Bağlantı ifadesi" olarak yerleştirilmesiyle yapılabilir.

ÖZGEÇMİŞ

Rahşan Tiken, 1975 yılında İstabul'da doğmuştur. Lise öğrenimini Kadıköy Anadolu Meslek Lisesi, Bilgisayar Programcılığı bölümünde tamamladıktan sonra 1993 yılında İstanbul Teknik Üniversitesi, Fen Edebiyat Fakültesi Matematik Mühendisliği bölümüne girmiş ve 1997 yaz döneminde mezun olmuştur. Yüksek lisans öğrenimini 1997 yılından bu yana İTÜ Fen Bilimleri Enstitüsü, Mühendislik Bilimleri Anabilim Dalı, Sistem Analizi programında sürdürmektedir. 1994 yılında İTU Vakfı Savunma Araştırmaları Merkezi'nde araştırmacı olarak çalışmıştır. 1998 yılından itibaren Garanti Bankası'nda Sistem Analisti olarak çalışmaya başlamıştır ve çalışma hayatına burada devam etmektedir.