İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

# BİR KONFEKSİYON OTOMASYON YAZILIMI

YÜKSEK LİSANS TEZİ

Müh. Ferhat TORGALÖZ



Tezin Enstitüye Verildiği Tarih : 15 Mayıs 1997

Tezin Savunulduğu Tarih : 22 Mayıs 1997

Tez Danışmanı : Doç. Dr. Tevfik Akgün

Diğer Jüri Üyeleri : Prof. Dr. Eşref Adalı

Doç. Dr. Nadia Erdoğan

MAYIS 1997

# ÖNSÖZ

Otomasyon uygulamasının gerçekleştirilmesinde ve koordinasyonunda sürekli destek olan yüksek lisans danışmanım öğretim üyesi Sayın Doç. Dr. Tevfik Akgün'e, bilgisayar destekli çalışmalar konusundaki bilgilerimin konfeksiyon sektörünün ihtiyaçlarına cevap verecek şekilde biçimlenmesinde desteği olan Tekstil Mühendisi Sayın Dr. Belgin Bozkurt'a ve özellikle tüm öğrencilik hayatım boyunca bana destek olan, büyük bir özverili ile hep yanımda olan anneme, ablama, merhum babama teşekkür ederim.

Böyle bir uygulamanın hayata geçirilmesi için bize çalışma ortamını ve çeşitli sistem gereksinimlerini sağlayan İ.T.Ü. KOSGEB Teknoloji Geliştirme Merkezine ve tüm çalışanlarına ve projenin gerçekleştirilmesinde beraber çalıştığımız arkadaşım Bilgisayar Müh. Levent Çuhacı'ya teşekkür ederim.

Tez çalışması esnasında ortaya konulan otomasyon yazılımının bu konuda Türkiye'de gerçekleştirilen çalışmalara yardımcı olacağını ümit eder, bilgisayar destekli çalışmaların piyasada faaliyet gösteren gerek kamu gerekse de özel teşebbüsler tarafından daha çok destek görmesini temenni ederim.

Ferhat Torgalöz
Mayıs 1997

# İÇİNDEKİLER

# ŞEKİL LİSTESİ

Sayfa No

# TABLO LİSTESİ

Sayfa No

# ÖZET

Günümüzde bir çok üretim ve hizmet sektöründe üretim sürecinin tamamlanması için sipariş değerlendirme, kavramsal çalışma, tasarım, öncül örnek, üretim aşamaları ve gereksinimleri, üretim ve kalite kontrol gibi bir dizi çalışmanın yapılması gerekir. Bu çalışmalar genellikle uzun zaman alan ve pahalı olan emeğe dayalı işlemlerdir. Daha hızlı, daha ucuz ve daha güvenilir sonuçlar elde edebilmek için üretimin çeşitli aşamalarında çeşitli şekillerde bilgisayar sistemlerinin, lojik tasarımların ve ölçme sistemlerinin kullanılması o sistemin otomasyonu anlamına gelir. Bu otomasyon sistemi bazen Bilgisayar Destekli Tasarım, bazen Bilgisayar Destekli Üretim veya bazen de her iki yöntemi içeren bir çözüm olabilir.

Konfeksiyon sektöründe ise Üretim İş Planması, Üretim Takibi, Stok Takibi gibi bir çok üretim aşaması mevcuttur. Bu tezin konusunu oluşturan çalışma konfeksiyon sektöründe kullanılmakta olan model tasarım, serilendirme ve pastal planı hazırlama tasarım aşamalarıdır. Hem emek yoğun ve hem de insan beğenisine dayalı bir çalışma olması nedeniyle istenen sonuçlar elde edilene kadar defalarca tekrarlanması gerekmektedir. Her bir deneme ek bir masraf ve zaman demektir. Bu değerler de üretim maliyetlerini arttıran faktörlerdir. Çoğu zaman bu ek masraflar altında maliyetler artmakta veya istenen sonuçlar sağlanana kadar deneme yapılmamakta ve ucuz fakat sevimsiz sonuçlar elde edilebilmektedir. Bu şartlar altında böyle bir üretim aşamasını bilgisayar desteği ile gerçeklemek konfeksiyon sektöründe önemli bir atılım olmaktadır. Bilgisayar desteği ile sağlanan sistem çözümleri yukarı da değindiğimiz gibi çok daha ucuz, daha hızlı ve defalarca tekrarlanabilirlik özelliklerine sahiptir. Bu özellikler ise maliyeti doğrudan etkileyen faktörlerdir.

Tez kapsamındaki ürün sayesinde konfeksiyon firması ihtiyaca uygun tasarımı model tasarım modülünde gerçekler. Pervaz, pile, çıt, dikiş payı gibi konfeksiyon sektörüne özgü özellikleri modeli oluşturan kalıp parçaları üzerinde tanımlayabilir. Tasarımcı isterse noktalar ve bunların arasındaki düz ve eğri çizgilerden oluşan kalıplar üzerinde nokta ekleme, silme, döndürme, aynalama, simetri, kesme, birleştirme gibi bir takım geometrik işlemleri gerçekleyebilir. Tasarımı biten model temel beden olarak tanımlanır ve diğer türetilmek istenen beden setleri ve bu setler arası geçişler tanımlanmak sureti ile serilendirme işlemi gerçeklenebilir. Tasarımı ve serilendirmesi biten modeller arasında aynı kumas malzemesi ile üretilecek olan modeller bir pastal siparişi şeklinde düzenlenir. Bu işlemin ardından eni ve boyu belli kumaş malzemesi üzerinde modellerin pastal yerleşim planı hazırlanır. Pastal yerleşim planı tasarlanırken önemli sorun modeli oluşturan kalıpların minimum kayıp ile kumaş malzemesi üzerine yerleşimlerinin düzenlenmesidir. Kalıpların yerleşim planı ya fare yardımı ile tek tek stoktan alıp kumaş malzemesine taşıma şeklinde ya da otomatik pastal tasarım algoritmaları sayesinde gerçeklenir.

Tez kapsamında ise bu otomasyon yazılımının oluşturulmasında kullanılan Microsoft Windows işletim sistemi, Nesneye Yönelik Programlama ( Object-Oriented Programming ) ve Microsoft Visual C++ programlama ortamı gibi tekniklerin ve yaklaşımların kısaca tanımı ve bu Otomasyon Sisteminin tanıtımı yapılmaktadır.

## SUMMARY


## A TEXTILE AUTOMATION SOFTWARE

The Experiences and knowledges, that advanced increasingly and were transmited parent to child, have exposed the present technologies in the course of time. Semi conductor tranzistor technology were developed at the beginning of 1970s. As a result of this, the science supplied that developments not possible to be realized or not economical are accomplished. The level attained finally at the technological development is computer era.

The using of computers at the works supplies to be obtained more rapidly the more reliable results. The making operations to the usual way needs the certain experiments that are achieved after a long time. But the making the same operations by computer needs the more little experiments. The cause of this is that the application gives users information in the common of working and presents users explanatory information in the various level of working. In addition, the computers can store information in the requested form at the appropriate time. The users can access later and use again this information. In this way, the users can obtain the different working instance. Such a structure supplies a flexible working way.

The computer aided design and computer aided management that is such advantages is used at the textile sector the same way to other management sectors. The first researches at this subject started on America at 1960s. They converted an extensive trade, because the France and Espanol firm went the market. Also at our country, these researches weren't developed. Therefore firms buyed the applications that produced at the other country and used these applications.

Our textile sector needed to develop and tended the research works. As a result of this, our porject was emerged. This project is developed in the Microsoft Visual C++ Integrated Development Environment . Its data is stored in the Microsoft Access Database Management System. Visual C++ has fully Windows Programming and Object-Oriented Programming features. Because of this, first I introduce what the Windows Programming and Object-Oriented Programming are and what their features are. And then I introduce the Textile Automation Software that is subject to my Master thesis.

**Windows Programming**

Microsoft Windows has emerged as the most popular graphical user interface environment for MS-DOS.

For the user, Windows provides a multitasking graphical-based windowing environment that runs programs espically designed for Windows. Programs written for Windows have a consistent appearance and command structure, and are thus often easeir to learn and use than conventional MS-DOS programs. Users can easily switch among different Windows programs and exchange data between them. Windows also provides an easy-to-use icon-based Program Manager for running programs as well as File Manager and Print Manager for file maintenance and printer-queue management.

Altough Windows exists primarily to run applications especially written for the environment, Windows can also run many programs written for MS-DOS. Of course, these programs cannot take advantage of many Windows feature, but in some cases they can be windowed and multitasked alongside Windows programs.

For the program developer, Windows provides a wealth of built-in routines that allow the use of menus, dialog boxes, scroll bars, and other components of a friendly user interface. Windows also contains an extensive graphics programming lanuage that includes the use of formatted text in a variety of fonts. Programmers can treat the keyboard, mouse, video display, printer, system timer, and RS-232 communication ports in a device-independent manner. Windows programs run the same on a variety of hardware configurations.

Windows provides considerable advantages to both users and programmers over the conventional MS-Dos environment. The benefits to users and the benefits to program developer are really quite similar, because the jobs of a program developers is to give users what they need and want. Windows makes this possible.

Windows is a graphical user interface (GUI), sometimes also called a "visual interface" or "graphical windowing environment". All graphical user interface make use of graphics on a bitmapped video display. In a graphical user interface, the video display itself becomes a source of user input. The video display shows various grahical objects in the form of icons and input devices such as buttons and scroll bars. Using keyboard (or, more directly, a pointing device such as a mouse), the user can directly manipulate these objects on the screen. Graphics objects can be dragged, buttons can be pushed, and scroll bars can be scrolled.

Under Windows, every program in effect becomes a RAM-resident popup. Several Windows programs can be displayed and running at the same time. Each program occupies a rectangular window on th screen. The user can move the windows aorund on th screen, change their size, switch between different programs, and transfer data from one program to another.

An operating system cannot implement multitasking without doing something about memory management. As new programs are started up and old ones terminate,

memory can become fragmented. The system must be able to consolidate free memory space. This requires the system to move blocks of code and data in memory.

Windows is a graphical interface, and Windows programs can make full use of graphics and formatted text on both the video display and printer. Programs written for Windows do not directly access the hardware of graphics display devices such as the screen and printer. Instead, Windows includes a graphics programming language (called the Graphics Device Interface, or GDI) that allows the easy display of graphics and formatted text. Windows virtualizes display hardware. A program written for Windows will run with any video board or any printer for which a Windows device driver is available. The program does not need to determine what type of device is attached to the system.

**Object-Oriented Programming**

Object,Oriented programming – sometimes fondly refered to as OOP – is a hot topic int the computer industry. It is a language that tries to solve problems in a new and better way by dividing programs into objects that have meaningful relationships with each other rather than into largely unrelated aggregations of functions and data. When you write an object-oriented program, your first task is to design a data form that corresponds to the essential features of your problem. Then you must create a set of procedures, or methods, that manipulate that data to solve the problem.

If you're writing program designed to solve more than one problem, you can create sets of data structures, and methods that work together, and then you can arrange those structures and behaviours into relationships and hierarchies that also work well together. Simply stated, you can break a problem down into objects that have built-in data and built-in behaviours. Then you can group those building blocks into a sensible arrangement that seem to be suited to solving your problem.

If your fist attmpt to solve the problem doesn't work as well as expected, you can rearrange the buildin blocks that make up an object-oriented program into different arrangement. Object-oriented langauges let you do that easily and conveniently because the objects that make up an object-oriented program are more independent than the procedures that make up a procedural program, and can therefore be rearranged and moved around more freely.

Furthermore, when you have designed an object-oriented program that work, you can reuse the program's code in other applications much more easily and convenently than you can when you're working with procedural programs such as C and Pascal. Thus, when you write programs into an object-oreinted language, you don't have to start from scratch every time you get a new program to write or a new problem to solve.

For almost 50 years now, software designers have used *procedural* and *structural* paradigms to create most kinds of computer programs. Procedural and structural

paradigms take a top-down, or structured, approach to program design. A task is partitioned into subtasks; those subtasks are further partitioned into still smaller subtasks, and so on until the main task has been divided into subtasks that are simple enough to be programmed in a conventional procedural language.

A object-oreinted program is made up of reusable objects that are relatively independent of other objects, yet have characteristics that are inherited from other objects and also have characteristics that can be passed on to other objects. This paradigm is said to have more in common with the standard paradigms used in other engineering disciplines than does the procedural paradigm that is used for program construction by conventional procedural languages.

It's important to note that OOP does not repudiate the procedural paradigm, but enhances it. Object-oriented programs makes use of procedure, but those procedures are encapsulated within objects, along with data that the procedures manipulate. Thus, object-oriented technology has fundamentally changed - but has not destroyed – the old procedural programming paradigm.

Other terms often encountered in OOP circles include these :

*Objects* : Ann object is a user-defined data type that resembles a C-style struct, but contains both data and functions that can manipulate that data. The data fields contained in an object are usually called *member variables*, but are sometimes refered to as *data member*. The functions contained in an object are usually called *member functions*, but are sometimes called *methods*. In an object-oriented langauge, the data and procedures that are grouped together to form objects can be used together to create more objects. In C++, *inheritance* lets you create objects that inherit properties of other objects. And objects can access data and call functions that are members of other objects.

*Classes* : In C++, every object is an instance of a user-created data type called *class*. A class is a framework that sets up the architecture of an object. An object is an instance of a class. Therefore, the act of creating an object is called *instantiating* the object.

*Data encapsulation* : In C++, The member functions declared within the definition of a class can access all member variables that belong to the same class. However, an object can safeguard its member variables from being accessed ar altered from elsewhere in a program. Tha ability that an object has to conceal its data from other parts of a program is called *encapsulation*.

*Data abstraction* : When you design a class in C++, you can conceal the details of how data is represented and modified inside the objects that belong to a class. Data abstraction is what lets the calling function ignore the details of how data types are represented inside objects and lets it concentrate instead on the job that it wants done.

*Inheritance* : In an object-oreinted language, you can create a new class from any existing class by modifying the properties of existing class. The usual purpose for deriving one class from another is to create a more specialized class. The derivation

of one class from another class is called *inheritance*. A class from which other classes are derived is called a *base class*, and classes that are derived from (or inherit from) a base class are called *derived classes*. The immediate base classs of a derived classs is called a *parent class*.

*Multiple Inheritance* : Some object-oriented languages - including C++ - support *multiple inheritance*. In a language that supports multiple inheritance, an object can inherit properties from more than one class.

*Polymorphism* : In biology, the occurrence of different properties in individual organisms, or in organisms in the same species, is called *polymorphism*. In a C++ program, polymorphism lets you create a family of classes with behaviours that vary, according to what is appropriate for each class. If you call a member function of a particular object, your call may have one result, and if you issue the same call to a a member function of another object, the result may be completely different.

**Pattern Design, Grading and Markermaking**

The first applications that the computer aided pattern design researches are based on is started by Howard Hughes, a famous and well-known American. He started first the development program that contains the computer aided hardware related to 2 D application. The basic idea of his research is that any drawing and cutting operation can be accomplished on the two dimensional surface.

This research is beginned on the Hughes Research Company at 1960 year. At the same year, a laser circle is introduced. At 1968 after eight years, Hughes that work together Genesco successfully developed a computer aided cloth cutting. It worked to the laser ray and had more speed and truth than the conventional methods.

It is awared of that this developed technological evoluation can allow us to make easier and faster the producting of ready-made suits. Hughes designed a system easily accomplishing the "grading" and "markermaking" ,that is most difficult production process, with the support of Autographics which had experiment a long time. This system had the very powerful main computer produced Hewlect Packard. It which is the first computer aided management (CAM) system is called AM1.

Hughes Apparel System developed continously and sold the AM1 throughout the ten years. Gerber a American firm bought it at 1978-9 years. Gerber had the experiment about the computer aided machine cutting the bulk cloth. It also developed the system which it had the patent and the copy-right. The Hughes's AM1 a Computer Aided Design and Computer Aided Management (CAD / CAM) for the markermaking and pattern design was used in order to cut the bulk cloth by Gerber. Gerber developed the AM1 and released the AM5 more modern than the AM1. Both systems worked on the main computer of Hewlect Packard and also can worked on the computer of IBM.

At themoment, the systems of American Camsco and France Lectra are built. Lectra started the researches at 1975 and released their first systems at 1978. They had the experiments about the Lazer preparatory and sold the lazer systems which can cutted the carton patterns, or the one layer cloth or the leather. Also both firm producted the Computer Aided Management systems (grading and markermaking) and tried to develop the Computer Aided Design system (graphics and cloth pattern design software) that works with together CAM. Gerber bought Camsco firm in order to terminate the competition and monopolize in the market which was weak at the beginning of 1980s.

At this case, two big firm Gerber and Lectra had the small trade. Their trade was rapidly increased after the two firm done the hard and serious sale program. After his moment, the trade was extensive and other computer firms participated and started the research about this subjects. Espanol Investronica participated powerfully this market. The other firms followed them. In the final, the researches and the CAD/CAM systems about this pattern design, grading and markermaking have the situation at present. They are similar together ,because they based on the basic idea which Hughes introduced at 1960s.

In the course of time, Our textile sector has developed and started to be the world-wide trade. Because of this, they have needed to do the researches and suppurted the working at this subject for not the back of other firms and countries. They have tried to produce and developed the native computer automation systems. Our project is emerged at the result of such a working.

# 1. BÖLÜM

## GİRİŞ

Günümüzde hızla gelişen talep piyasasındaki ihtiyaçlara cevap verebilmek için arz sektörleri arasında bir rekabet ortamı oluşmuştur. Böylesi bir rekabet ortamı kısa bir zaman diliminde pek çok işin gerçekleştirilmesini zorunlu kılmaktadır. İhtiyaçlara alışıldık tarzda yöntemlerle yani emek yoğun çalışmalarla sonuçlar üretmeye gayret edildiği takdirde piyasa talebinin gerisinde kalınacağı açıktır. Bu aşamada teknolojinin son ürünü olan bilgisayar destekli yaklaşımlar devreye girmektedir.

Bilgisayar destekli işlemler emek yoğun bir şekilde gerçekleştirilen işlemlere göre daha süratli, güvenilir ve esnek olması sebebiyle günümüzde tüm sektörlerde yaygın hale gelmektedir. Bilgisayarın bize sağladığı, bilgiyi çeşitli çalışma kademelerinde istenilen formatta saklama ve gerektiğinde yeniden elde edebilme özelliği sayesinde sistemler oldukça esnek bir hal almıştır.

Konfeksiyon sektöründe bir modelin üretimine başlamadan önce çeşitli safhalardan geçilmesi gerekir. Bu aşamalar ne kadar çabuk aşılırsa piyasa talebine ve diğer firmalarla rekabete o derece uyum sağlanır. Alışıldık tarzda üretim aşamalarından biri de model kalıplarının tasarım aşaması ve bu kalıpların kumaş üzerine serildiği, yerleşiminin planlandığı pastal yerleşimini hazırlama aşamasıdır. Bu aşamalar aynı zamanda piyasaya yeni bir modelin çıkarılabilmesi için ilk yapılacak çalışmalardır.

Tasarımı gerçekleştirilmiş kalıplar hakkında bilgiler saklanır. Daha sonraki zaman dilimlerinde tekrar üzerinde başka değişiklikler yapılarak veya kopyalar alınarak yeni modellerin oluşturulmasında kullanılabilir. Böylesi yaklaşımlar içeren bir aşamayı bilgisayar desteği gerçeklemek işlemleri oldukça çabuklaştırmaktadır.

Bu çalışmada temel gayemiz bilgisayar hakkında az bilgiye sahip kişilerinde kullanabileceği kadar kolay, sağladığı açıklayıcı bilgilerle kullanıcı dostu, kullanıcı ile etkileşimli çalışan bir otomasyon yazılımı oluşturabilmektir. Çalışmamızda konfeksiyon sektörünün ihtiyacı olan otomasyon sistemi üzerinde çeşitli uygulamalar yapılmış ve mümküm mertebe yeni gelişmeler kullanılmaya gayret edilmiştir. Bu maksatla gelecekte de yaygın bir şekilde kullanılacağı tahmin edilen Windows çalışma ortamında, en modern programlama tekniği olan Nesne Yönelik Programlama ( Object-Oriented Programming ) yöntemi üzerine kurulu Microsoft Visual C++ programlama ortamı kullanılmıştır. Veri tabanı olarak yaygın bir şekilde kullanılmakta olan Access veri tabanı ortamı tercih edilmiştir.

# 2. BÖLÜM

# WINDOWS PROGRAMLAMA

Tezin konusunu oluşturan konfeksiyon otomasyon yazılımının Microsoft Visual C++ proje geliştirme ortamında gerçekleştirildiği daha önce bahsedilmişti. Bu bölümde Microsoft Visual C++'ın bileşenlerinden biri olan Windows Programlama (Windows Programming) yönteminin takdimi yer almaktadır.

## 2.1 Windows ve Windows Programlama :

Microsoft Windows MS-DOS için en popular grafik kullanıcı arabirim ortamıdır. Windows özel olarak tasarlanmış programların çalıştığı çok görevli grafik temelli pencerelerden oluşan bir ortam sağlar. Windows için yazılmış programlar düzenli bir görünüm ve komut yapısına sahiptir ve sıradan MS-DOS programlara göre öğrenilmesi ve kullanılması daha kolaydır. Kullanıcı farklı Windows programları arasında kolayca anahtarlayabilir ve onlar arasında verileri değiş tokuş edebilir.

Windows program geliştiriciler için ise menülerin, dialog kutularının, scroll barların ve diğer kullanıcı dostu bileşenlerin kullanımına izin veren hazır rutinleri sağlar. Ayrıca çeşitli fontlarda formatlı textlerin kullanımını sağlayan yoğun bir grafik programlama dili de içerir. Programcılar klavye, mouse, video display, sistem timer ve RS-232 haberleşme iskeleleri ile aygıt-bağımsız bir tarzda ilgilenebilir. Windows programları çeşitli donanımlarda aynı şekilde çalışır.

## 2.2 Windows Programının Başlama Noktası :

Alışılan tarzdaki sıradan bir C programı için programın başlanma noktası *main* olarak adlandırılan fonksiyondur. Bu fonksiyon programın çalışmaya başladığı yerdir. Fakat bir Windows program için başlangıç noktası *WinMain* olarak adlandırılan bir fonksiyondur. *WinMain* fonksiyonunun tanımı aşağıdaki gibidir :

```
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
                   LPSTR lpszCmdParam, int nCmdShow)
```

Bu fonksiyon, Windows'a özel bir fonksiyon çağırma ardışılı olan PASCAL'ı kullanır. Fonksiyon *WinMain* olarak adlandırılmalıdır ve dört parametre almalıdır.

*hInstance* parametresi "örnek tanımlayıcı" ("instance handle") olarak adlandırılır. Windows altında program çalıştırıldığı zaman programı tanımlayan benzeri olmayan bir sayıdır. Bu sayede Windows altında aynı programın bir çok kopyasını çalıştırmak mümkün olur. Her bir kopya bir "örnek" (instance) olarak adlandırılır ve herbiri farklı bir *hInstance* değerine sahiptir. Çok görevli işletim sistemlerinde örnek tanımlayıcı , "görev ID" (task ID) veya "süreç ID" (process ID) olarak kullanılır.

*hPrevInstance* ("önceki örnek") parametresi aynı programın hala aktif olan en sonuncu örneğin örnek tanımlayıcısıdır. Eğer şimdi yüklenen programın daha önce hiçbir örneği yoksa *hPrevInstance* 0 veya NULL değerini taşıyacaktır.

*lpszCmdParam* parametresi programa her bir komut satırı parametresini aktarmayı sağlayan 0 ile sonlanmış katara bir long pointer'dır. Komut satırı parametreleri ile birlikte program ismini yazmak suretiyle bir Windows programı çalıştırılabilir.

*nCmdShow* parametresi pencerenin Windows'da nasıl başlatılacağını gösteren bir sayıdır. 1 ile 7 arasında bir değer alır ve de genellikle SW_SHOWNORMAL ( 1 ) veya SW_SHOWMINNOACTIVE ( 7 ) deperlerini alır.

**2.3 Pencere Sınıfı** (Window Class) **ve Pencerenin Oluşturulması:**

Bir pencere daima bir pencere sınıfını temel almalıdır. Pencere sınıfı, pencere için mesajları işleyen pencere prosedürünü (window procedure) tanıtır. Bir pencere oluşturulmadan önce daima bu pencerenin temel özelliklerini tanımlayan bir pencere sınıfı yaratılmış olmalıdır. Bu nokta unutulmaması gereken önemli bir husustur.

Birden daha fazla pencere tek bir pencere sınıfını temel alabilir. Örneğin Windows'daki edit box'lar aynı pencere sınıfını temel alabilir. Windows programında bir pencere oluşturulmadan önce pencere sınıfı, *RegisterClass* fonksiyonunun kullanımı ile sisteme tanıtılmalı ve saklanmalıdır. Bu fonksiyon sadece tek bir parametre alır; o da WNDCLASS tipinde bir struct'tır. Sistem yapısı gereği programın sadece ilk örneğinde pencere sınıfının saklanmasına ihtiyaç duyulur. Bu şekilde oluşturulmuş bir pencere sınıfı programın tüm ardışıl örnekleri için sistemde tanımlı kalır.

WNDCLASS struct'ı 10 alana sahiptir. En önemli iki alan *lpfnWndProc* ve *lpszClassName* alanlarıdır. *lpszClassName* alanı, daha sonra *CreateWindow* fonksiyonu ile pencere oluşturulurken kullanılacak olan pencere sınıfının ismidir. *lpfnWndProc* alanı bu sınıfı temel alan tüm pencereler için kullanılmış pencere prosedürünün adresidir. Bu pencere prosedürü, pencere için tüm mesajları değerlendirir ve işler. Diğer alanlar bu sınıfı temel alan tüm pencerelerin karakteristiklerini tanımlar.

Daha önce belirttiğimiz gibi pencere sınıfı bir pencerenin genel karakteristiklerini tanıtır. Gerçekten pencere oluşturulurken ise pencere hakkında daha detaylı bilgi *CreateWindow* fonksiyonunda belirtilir. Bir programın sadece ilk örneğinde sisteme bir pencere sınıfı tanıtılmasına rağmen herbir örnek için ayrı bir pencere oluşturulması gerekir. Herbir örnek kendisine özel bir pencereye sahiptir ve tüm pencereler aynı pencere sınıfını temel alır. *CreateWindow* fonksiyonunun ilk parametresi temel alınan ve daha önce sisteme tanıtılmış olan pencere sınıfının ismidir. Bu şekilde pencere ve pencere sınıfı arasında bağlantı kurulmuş olur.

*CreateWindow* fonksiyonu oluşturulmuş pencereye HWND tipinde bir tanımlayıcı(handle) geri döndürür. Windows'daki her pencere bir tanımlayıcya sahiptir. Program içinde pencereye erişilmek ve üzerinde işlem yapılmak isteniyorsa bu tanımlayıcı kullanılır.

## 2.4 Pencere Prosedürü (Window Procedure) ve Mesajların İşlenmesi :

Windows'da çalıştırılan her bir program için Windows bir *mesaj kuyruğunu* (*message queue*) güncel tutar. Bir giriş olayı oluştuğu zaman Windows giriş olayını bir *mesaja* (*message*) dönüştürür ve onu programın mesj kuyruğuna yerleştirir.

Program mesaj kuyruğundan bu mesajları "mesaj çevrimi" olarak bilinen aşğıdaki gibi kod bloğunu çalıştırmak sureti ile alır.

```
while  ( GetMessage (&msg, NULL, 0, 0 ) )
        {
        TranslateMessage ( &msg ) ;
        DispatchMessage ( &msg ) ;
        }
return msg.wParam;
```

Bu kod parçası WinMain fonksiyonunun son kısmında icra edilir ve böylece bu uygulamaya ait olan tüm mesajlar program tarafından değerlendirilir.

Windows'da bir programın oluşturduğu her pencere bir pencere prosedürüne sahip olur. Bu pencere prosedürü ya o programın içindeki ya da bir DLL'deki bir fonksiyondur. Bu pencere prosedürünün ismi pencere sınıfında belirtilen sınıf ismidir. Windows pencereye mesajları pencere prosedürünü çağırmak yoluyla iletir. Pencere prosedürü mesajları ilişkili olduğu pencere için işler ve kontrolü Windows'a geri döndürür.

Nesne yönelik programlamada bir nesne kod ve veriden oluşur. Windows'ta pencere nesneyi oluşturur. Kod, pencere prosedürüdür. Veri ise hem pencere prosedürü tarafından, hem her bir pencere için Windows tarafından ve hem de sistemde mencut olan pencere sınıfı tarafından kullanılan bilgidir.

Bir Windows programı çalışmaya başladığı zaman Windows program için bir "mesaj kuyruğu" (message queue) oluşturur. Bu mesaj kuyruğu programın oluşturabildiği tüm pencereler için mesajları depolar. Program bu mesajları kuyruktan almak için ve uygun pencere prosedürlerine dağıtmak için yukarıda belirtilmiş olan "mesaj çevrimi" (message loop) adı verilen bir kod parçasını içerir. Diğer mesajlar mesaj kuyruğuna yerleştirilmeden doğrudan pencere prosedürlerine gönderilebilir.

Windows programında gerçek çalışma pencere prosedüründe olur. Pencere prosedürü pencere çalışma alanının nasıl görüneceğini ve pencerenin kullanıcı girişlerine ne cevap vereceğini saptar. Pencere prosedürü daima aşağıdaki gibi tanımlanır. Fakat pencere prosedür ismi pencere sınıfında belirtilmiş olan herhangi bir isim olabilir.

long FAR PASCAL WndProc ( HWND hwnd, WORD message, WORD wParam, LONG lParam)

İlk parametre *hwnd*(*handle to a window*), mesajı alan pencereye bir tanımlayıcıdır. Bu tanımlayıcı *CreateWindow* fonksiyonundan geri dönen tanımlayıcı ile aynıdır. Eğer program aynı pencere sınıfından bir çok pencere oluşturmuş ise *hwnd,* mesajı alan özel pencereyi tanımlar.

İkinci parametre mesajı tanımlayan bir sayıdır (özellikle 16 bit işaretsiz tamsayı veya WORD). Son iki parametre ise mesaj hakkında daha fazla bilgi sağlar.

Bir pencere prosedürünün aldığı her bir mesaj, pencere prosedüründeki *message* parametresi olan bir sayı ile tanımlanır. WINDOWS.H başlık dosyasında her bir mesaj parametresi için WM_ ("window message") ön eki ile başlayan tanımlayıcıları belirtilir.

Bir pencere prosedürü mesajı aldığı zaman 0 ile geri dönmelidir. Pencere prosedürünün işlemediği tüm mesajlar, *DefWindowProc* isimli bir Windows fonksiyonuna gönderilmelidir. Aynı şekilde *DefWindowProc*'dan geri dönen değer pencere prosedüründen geri döndürülmelidir.

## 2.5. Windows'un Grafik Aygıt Arabirimi (Graphics Device Interface) :

Oluşturulan pencerenin görüntü yüzeyini boyamak için Windows'un Grafik Aygıt Arabirimi(GAA) kullanılır. Her GAA fonksiyonu ilk parametre olarak *aygıt içereği tanımlayıcı(handle to a device context)* 'sına ihtiyaç duyar.

*Tanımlayıcı(handle)*, Windows'un bir nesneye iç referans yapmak için kullandığı bir sayıdır. Bu tanımlayıcı yine Windows'tan elde edilir ve diğer fonksiyonlarda kullanılır. *Aygıt içerik tanımlayıcısı(handle to a device context - hdc)* sayesinde ise GAA fonksiyonları kullanılır ve pencerenin çalışma alanı istenildiği gibi boyanabilir, düzenlenebilir.

*Aygıt içeriği (device context - DC)* gerçekte GAA tarafından güncellenen bir veri yapısıdır. Printer, plotter ya da video görüntüleyici gibi özel bir görüntü aygıtı ile bağlantılıdır. Video görüntüleyici için bir aygıt içeriği genellikle ekrandaki özel bir pencere ile bağlantılıdır.

Windows GAA fonksiyonları ile çalışırken aygıt içerik yapısındaki değerleri kullanır. Bu değerlere aynı zamanda aygıt içeriğinin "özellikleri" de denir. Örneğin *TextOut* fonksiyonu aygıt içeriğinin özelikleri sayesinde yazılacak yazının rengini, geri plan rengini, x-y koordinatlarının pencerenin çalışma alanına nasıl dönüştürüleceğini ve yazı görüntülendiğinde Windows'un hangi fontu kullanacağını saptar.
Bir program pencereyi isteğe göre düzenlemek için öncelikle aygıt içeriğine bir tanımlayıcı elde etmelidir. Elde edilen aygıt içeriğinin kullanımı sonlandığı zaman program bu tanımlayıcıyı serbest bırakmalıdır. Çünkü bu tip tanımlayıcılar sistem

kaynaklarını bloke ederler. Bu duruma son vermek ve sistem kaynağını tekrar kullanıma açmak için böyle bir işleme başvurulur.

Windows'ta herşey birbiri ile bağlantılıdır. Öyle ki video görüntüleyicide bazı grafikler çizmek istersek bir "aygıt içeriği tanımlayıcısı"(*handle to a device context - HDC*)'na ihtiyacımız vardır. Onu elde etmek için bir "pencere tanımlayıcısı"(*handle to a window - HWND*)'nı elde etmemiz gerekir. Onu almak için de bir pencere oluşturmamız ve pencereyi "mesajları" almaya hazırlamamız gerekir. Mesajları almak ve işlemek için ise bir "pencere prosedürü"(*window procedure*)'ne gereksinim vardır. İşte bu noktada ise Windows programının yazımı başlar.

## 2.6. Windows'ta Çok Görevlilik :

Windows ortamında birden fazla Windows programı aynı anda çalıştırılabilir ve görüntülenebilir. Her bir program ekranda dikdörtgen bir pencereyi kaplar. Kullanıcı ekran üzerinde pencereyi hareket ettirebilir, onun ölçülerini değiştirebilir, programlar arasında geçişler yapabilir ve bir programdan diğerine veri transfer edebilir.

Böyle bir çalışma olanağı tamamen Windows'un bize sağladığı bir imkandır. Bu sayede bir modulun parçaları olan birden fazla uygulama aynı anda çalıştırılmak sureti ile bu uygulamalar arasında çok kolay bir geçişlilik ve esneklik sağlanmış olur. Bu tarz bir çalışma beraberinde bellek yönetimini de getirir.

## 2.7. Windows'ta Bellek Yönetimi :

Bir işletim sisteminde bellek yönetimi hakkında hiç bir çalışma yapılmaksızın çok görevlilik sağlanamaz. Yeni bir program çalışmaya başladığı ve işini bitirip sonlandığı zaman bellek parçalara ayrılmış olur. Sistem bu şekilde oluşan boş bellek alanlarını birleştirmelidir. Bu da bellekteki kod ve veri bloklarının sistem tarafından

hareket ettirilmesini gerektirir. Windows'ta çalışan programlar belleği aşırı yükler; bir program herhangi bir anda belleğin alabileceğinden daha fazla kodu içerebilir. Windows program kodunu bellekten çıkarabilir ve daha sonra programın .EXE dosyasından kodu tekrar yükleyebilir. Kullanıcı bir programın bir kaç örneğini çalıştırabilir. Tüm bu örnekler bellekte aynı kodu paylaşır. Windows altında çalışan programlar "dinamik bağlı kütüphaneler"(*dynamic linking libraries*) olarak adlandırılan diğer .EXE dosyalarındaki rutinleri de paylaşabilir. Windows dinamik bağlı kütüphanelerdeki rutinlerle programı çalışma anında bağlamak için bir mekanizmaya sahiptir. Esasında Windows'un kendisi de dinamik bağlı kütüphanelerin bir kümesidir.

Bellek yönetimi daima Windows'un en önemli konularından biri olmuştur. Windows 1 bile kompleks bir bellek yönetim yapısına sahiptir. Aşağıda Windows 1 bellek yönetiminin bazı özellikleri yer almaktadır :

- Windows aynı programın bir çok örneğini çalıştırdığı zaman her bir örnek için aynı kaynaklar(icon, cursor, menu template ve dialog box template) ve aynı kod segmentleri kullanılır. Çoğu durumda Windows sadece bir programın her bir örneği için tek olan veri segmentlerine ihtiyaç duyar.

- Windows'ta tahsis edilen bellek parçalarının çoğunluğu hareket ettirilebilirdir Çoğu durumda da bu bellek parçası bir programın kod segmentlerini, veri segmentlerini ve kaynaklarını içerir.

- Kod segmentleri ve kaynaklar ekseriyette "istek üzerine yüklenen"(demand-loaded) cinstedir. Bir program onlara özellikle ihtiyaç duyana kadar Windows onları belleğe yüklemez.

- Kod segmentleri ve kaynaklar genelde bellekten alınabilir(discardable) cinstedir. Windows belleği boşaltmaya ihtiyaç duyduğu zaman bellekten segmentleri alır ve daha sonra program onlara gereksinim duyduğu zaman programın .EXE dosyasından onları tekrar belleğe yükler.

Bu bellek yönetim özellikleri belleğe sığamayacak kadar büyük olan programların Windows 1 altında çalıştırılmasına olanak sağlar. Bu bellek yönetim planında önümüze çıkan problem Windows'un hard disk'ten kod segmentlerini ve kaynaklarını sık sık tekrar yüklemesi ve dolayısıyla program performansının kötüleşmesidir. Bu sebeple Windows 2'ye Expanded Memory Specifition (EMS) 4.0 ve Windows 3'e de protected-mode desteği eklenmiştir.

### 2.7.1. Windows'ta Bellek Organizasyonu :

Windows tarafından kontrol edilen tüm bellek alanına "global bellek" ya da "global heap" adı verilir. Bu alan MS-DOS'un Windows'u belleğe ilk yüklediği yerden başlar ve mevcut belleğin üstünde çoğunlukla da fiziksel belleğin üstünde sonlanır. Global bellekten tahsis edilen her bellek bloğu bir segmenttir. Genelde tahsis edilmemiş global bellek alanı "serbest bellek" (free memory) olarak adlandırılır.

Bir Windows programı bir ya da daha fazla kod segmentine ve veri segmentine sahip olabilir. Windows programı belleğe yüklediği zaman kod için global bellekten en az bir segment ve veri için bir segment tahsis edilir.

Windows'un tüm bellek alanındaki her segment Windows'un segmentleri nasıl yöneteceğini belirten belirli özelliklerle işaretlenir. ilk ve en önemlisi segmentlerin ya "sabit"(fixed) ya da "hareket ettirilebilir"(moveable) olarak işaretlenmesidir. Windows diğer bellek tahsisleri için yer gerekliyse bellekte hareket ettirilebilir olarak işaretlenmiş segmentleri hareket ettirebilir. Böyle bir durumda bu segment ile bağlantılı olan tüm işaretçilerin yeniden düzenlenmesi gerekebilir. Sabit olan segmentler bellekte hareket ettirilemez.

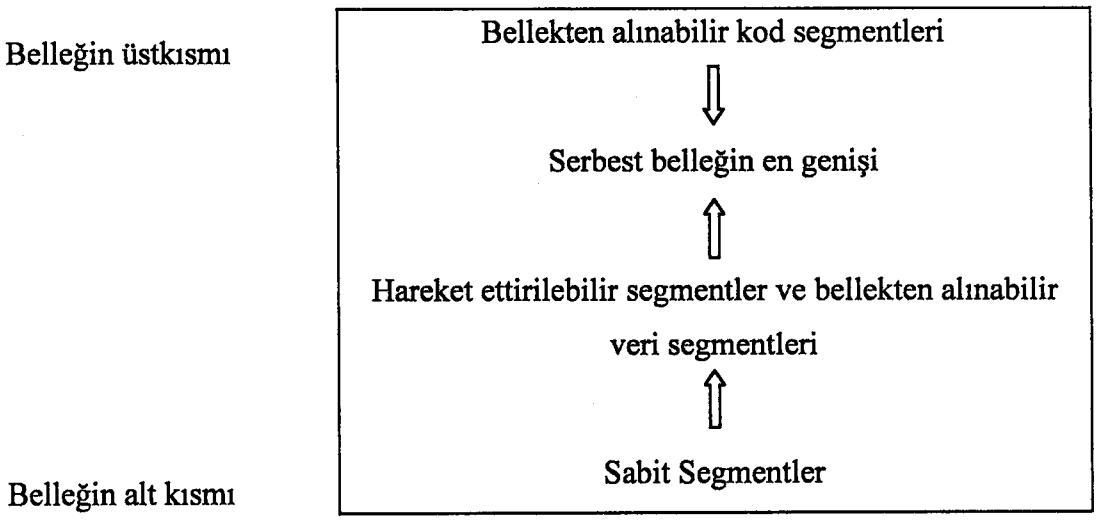
Hareket ettirilebilir segmentler aynı zamanda "bellekten alınabilir" (discardable) olarak işaretlenebilir. Windows ek bellek alanına ihtiyaç duyduğu zaman bu özellikle işaretlenmiş segmentleri bellekten alıp bellekte yer açabilir. Windows bir bellek segmentini serbest bırakmak istediği zaman bellekten alınabilir olarak işaretlenmiş

segmentlerden hangisini seçeceğini saptamak için "yakın geçmişte en az kullanılmış olanı taşı" (least recently used-LRU) algoritmasını kullanılır. Bellekten alınabilir olarak işaretlenmiş segmentler aynı zamanda hareket ettirilebilir olmalıdır. Bununla birlikte hareket etirilebilir segmentler daima bellekten alınabilir olmayabilir.

### 2.7.1.1. Global Bellek Düzeni :

Yukarıda bahsettiğimiz gibi global bellek MS-DOS'un Windows'u ilk yüklediği noktadan başlar ve mevcut belleğin üst kısmında sonlanır. Windows global belleğin alt kısmını sabit segmentlere, üst kısmını ise aynı zamanda hareket ettirilebilir olan bellekten alınabilir olan segmentlere ayırır. Sabit segmentler aşağıdan yukarıya doğru, bellekten alınabilir olan segmentler tersi yönde yukarıdan aşağıya doğru tahsis edilir.

Windows sabit segmentler ve bellekten alınabilir segmentler arasında hareket ettirilebilir segmentleri ve bellekten alınamaz veri segmentlerini yerleştirir. Serbest bellek bloğunun en büyüğü genellikle alınabilir segmentlerin altında yerleştirilir. Aşağıdaki şekilde burada anlatılan global bellek düzeni ve bellek parçalarının genişleme doğrultusu görülmektedir.




Şekil 2.1. Windows'ta bellek organizasyonu.

### 2.7.1.2 Lokal Bellek :

Her Windows programı default veya otomatik olarak adlandırılan en az bir veri segmentine sahiptir. Programın DS ve SS segment işaretçilerinin her ikisi de bu segmente işaret eder. Windows'un idare ettiği "global belleğin" aksine bu otomatik veri segmentine bizim programın "yerel belleği" adı verilir. Windows'un global bellek organizasyonu içinde bizim programın otomatik veri segmenti çoğunlukla hareket ettirilebilmesine rağmen bellekten alınamaz bir segmenttir. Bu segmente DGROUP adı verilir.

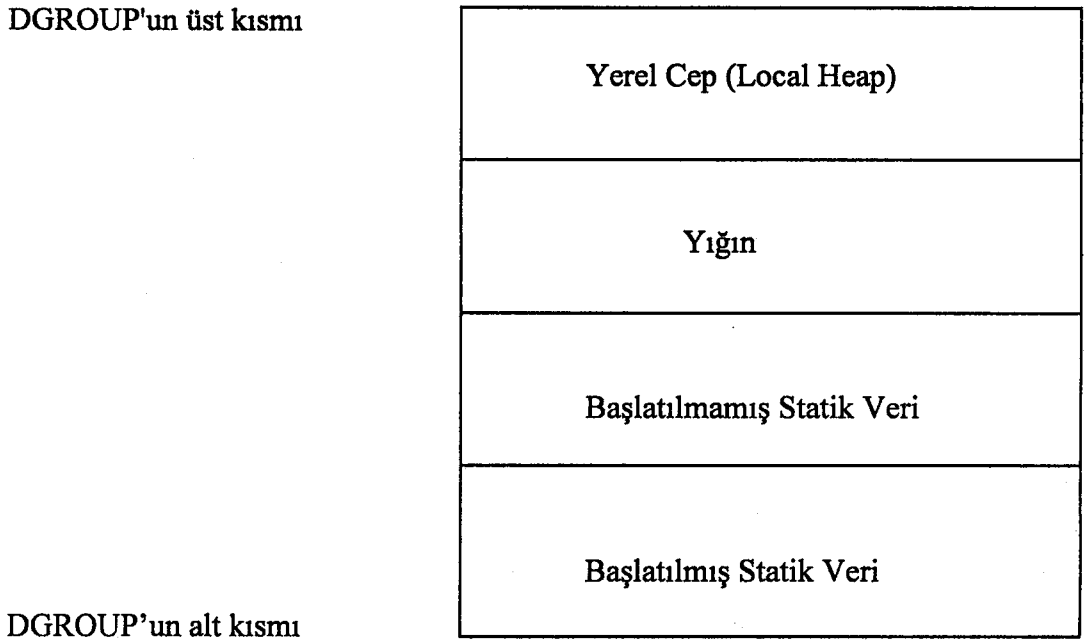
Windows programlarında DGROUP içindeki bellek,dört alanda şekil 2.2 gösterildiği gibi organize edilir. Bu dört alan aşağıda tarif edilmiştir :

- **Başlatılmış statik veri :** Bu alan fonksiyonların dışında tanımlanan başlatılmış değişkenleri, fonksiyonların içindeki başlatılmış statik değişkenleri ve kayan noktalı sayılar ile açık katarları içerir.

- **Başlatılmamış statik veri :** Bu alan fonksiyonların dışında tanımlanan başlatılmamış değişkenlere ve fonksiyonlar içinde statik olarak tanımlanan başlatılmamış değişkenlere sahiptir.

- **Yığın :** Bu alan fonksiyonlarda tanımlanan otomatik veri parçalarını, fonksiyonlara aktarılan veriyi ve fonksiyon çağrıları esnasında geri dönüş adreslerini içerir.

- **Yerel cep (local heap) :** Bu alan program tarafından dinamik tahsislerde kullanılmak üzere ayrılmış serbest bellektir.




Şekil 2.2 DGROUP içinde belleğin organizasyonu.

# 3. BÖLÜM

# NESNE YÖNELİK PROGRAMLAMA ve C++

Tezin konusunu oluşturan konfeksiyon otomasyon yazılımının Microsoft Visual C++ proje geliştirme ortamında gerçekleştirildiği daha önce bahsedilmişti. Bu bölümde Microsoft Visual C++'ın bileşenlerinden biri olan Nesne Yönelik Programlama (Object-Oriented Programming – OOP) yönteminin takdimi yer almaktadır.

## 3.1. Nesne Yönelik Programlama Nedir? :

Nesne Yönelik Programlama (Object-Oriented Programming), programlama sahasına yeni bir yaklaşım getirir. Bu yöntem bulunmadan önce programlama için bazı yöntemler geliştirilmiştir. Gelişimin kritik noktalarında programcının gittikçe komleks hala gelen programı ele alabilmesine yardımcı olmak için çeşitli yaklaşımlar oluşturuldu.

Günümüzde yaygın bir şekilde kullanılan yöntemlerden biri olan yapısal programlama (structured programming) mükemmel sonuçlar vermesine rağmen programın boyutu belli bir ölçüyü geçince performans ve kontrolu gittikçe kötüleşmektedir. Bu şekilde kompleks programların ele alınabilmesi için mutlaka nesne yönelik programlama yöntemi kullanılmalıdır. Nesne yönelik progralama (NYP) yapısal programlamada ortaya konulan ilkelerin en iyilerini temel alır ve bu ilkelerle programı farklı bir tarzda organize etmemizi sağlayan yeni kavramlarla birleştirir. NYP bir problemin ilişkili alt gruplara ayrılmasına olanak sağlar. Herbir alt grup bir nesne (object) olarak ele alınır. Nesnelerin kendi verileri ve bu verilerin

üzerinde çeşitli işlemlerin yapılabilmesine olanak sağlayan komut kümeleri yani fonksiyonları vardır. Böyle bir yöntem pogramın karmaşıklığını azaltır ve programcının daha geniş programları ele alabilmesine olanak sağlar.

Yukarıda bahsettiğimiz *nesne* (object), C'de mevcut olan struct yapısına benzeyen kullanıcı tanımlı bir veri yapısıdır. Fakat nesne, struct yapısına ek olarak hem verileri ve hem de bu veriler üzerinde çeşitli işlemleri gerçeklemeyi sağlayan fonksiyonları içerir. Nesnede içerilmiş olan verilere genellikle *üye değişkenler* (*member variables*) veya bazen de *veri üyeleri* (*data members*) adı verilir. Fonksiyonlara ise genellikle *üye fonksiyonlar* (*member fonctions*) veya bazen de *metodlar* (*methods*) adı verilir.

NYP'da her nesne *sınıf* (*class*) adı verilen kullanıcı tarafından oluşturulmuş veri tipinin bir örneğidir. Sınıf, nesnenin ana yapısını oluşturan bir çalışma çerçevesidir. Bir sınıfın bildirimi (declaration) yapıldığı zaman bu sınıfa ait nesneler için hiçbir bellek tahsisi yapılmaz. Yani nesne oluşturulana kadar bir nesne için bellek ayrılmaz. Kısaca söyleyecek olursak bir nesne bir sınıfın örneğidir.

### 3.2. C++ :

C++, nesne yönelik programlamaya C programcının cevabıdır. C dilinin gelişmiş bir versiyonudur. C'nin temel çözümleri üzerine inşa edilen C++, nesne yönelik programlama için bazı desteklere sahiptir. Ayrıca birçok yeniliği ve onu "daha iyi C" yapan bir çok özelliği içerir. C++ programlama gayretlerini azaltır. C++ 'ın C dilini temel alması nedeniyle C'de yapabildiğimiz herşeyi C++'a uygulayabiliriz. C dilini bilen birinin bu dili öğrenmesi o kadar da zor değildir.

C++ 1980 yılında New Jersey, Murray Hill'de Bell laboratuvarlarındaki çalışmalar esnasında Bjarne Stroustrup tarafından geliştirilmiştir. Başlangıçta "C with classes" adı verilmiş ve bu isim 1983 yılında C++ olarak değiştirilmiştir. 1980'den beri, C++ 1985'te ve bir diğeri de 1990'da olmak üzere iki revizyon geçirmiştir. Hala C++

üzerinde çalışmalar devam etmektedir ve bazı özellikler hala geliştirilmekte ve eklenilmektedir.

### 3.3. Nesne Yönelik Programlamanın Temel Özellikleri :

Tüm NYP dillerini diğer dillerden ayıran temel özellikler şunlardır ; veri kapsamı (data encapsulation), çok şekillilik (polymorphism), devralma (inheritance). Şimdi bu kavramlara kısaca değinelim :

**Veri Kapsamı (Data Encapsulation)**

NYP'da bir sınıfın tanımı içinde bildirilmiş üye fonksiyonar aynı sınıf ait olan tüm veri değişkenlerine erişebilir. Buna karşılık bir nesne programın içinde herhangi bir yerden üye değişkenlerine erişilmesine karşı önlem alabilir. Bir nesnenin üye değişkenlerini programın diğer kısımlarına karşı gizlemesi yeteneğine veri kapsamı (data encapsulation) adı verilir. Bir nesne içinde veri, kod veya her ikisi de veri kapsamı olarak private, protected, public tanımlarından birini alır. Bu tanımlamaya göre veri veya fonksiyonun kapsam alanı yukarıdaki sıraya göre genişler.

**Devralma (Inheritance)**

NYP'da mevcut bir sınıfın özelliklerini değiştirmek sureti ile mevcut herhangi bir sınıftan yeni bir sınıf oluşturulabilir. Böyle bir yaklaşımın gayesi daha özelleşmiş bir sınıf tanımlamaktadır. Bir sınıftan başka bir sınıf türetme özelliğine devralma (inheritance) adı verilir. Sınıfın türetildiği sınıfa taban sınıf (base class) adı verilirken bir taban sınıftan türetilen sınıfa türetilmiş sınıf (derived class) adı verilir. Sınıfın türetildiği bir üst sınıfa ata sınıf (parent class) adı verilir. Türetilmiş bir sınıf bir yada daha fazla ata sınıftan özellikler devralabilir. Böyle bir duruma da çoklu devralma (multiple inheritance) adı verilir.

**Çok Şekillilik (polymorphism)**

Biyoloji de aynı cinsten organizmalarda farklı özelliklerin oluşumuna bu ad verilir. NYP'da ise çok şekillilik, herbir sınıfa uygun muhtelif özelliklere sahip sınıf grupları oluşturulabilmesi özelliğidir. Bir başka deyişle bir fonksiyon ismi ile isimleri aynı yaptıkları farklı çoklu fonksiyon gruplarından o sınıfa uygun olan birinin çağrılabilmesidir.

## 3.4. Sınıf Çeşitleri :

C++'da sınıfların tanımlanması esnasında kullanabileceğimiz bazı özel teknikler vardır. Örneğin;

- *İçiçe sınıflar* (*Nested classes*). Bir sınıfın tanımı diğer diğer sınıfın tanımı içine yerleştirildiği zaman içteki sınıf bir *içiçe sınıf* (*nested class*) olarak adlandırılır.

- *Birleşik sınıflar* (*Composed classes*). Bir sınıfın bildirimini başka bir sınıfın tanımı içine yerleştirdiğimiz zaman bildirimi içeren sınıf bir *birleşik sınıf* (*composed class*) olarak adlandırılır. Bir nesne, diğer bir sınıf içinde bildirimi yapılan bir sınıfa ait olduğu zaman bu nesne *alt nesne* (*subobject*) olarak adlandırılır.

- *Arkadaş sınıflar* (*Friend classes*). Bir sınıfın tanımında diğer bir sınıfı arkadaş sınıf olarak bildirebiliriz. Bir sınıf diğer bir sınıfla arkadaşlık ilişkisi içinde olduğu zaman arkadaş olarak tasarlanan sınıf diğer sınıfın *private* ve *protected* üyelerine erişebilir.

- *Tamamlanmamış sınıf bildirimleri* (*Incomplete class declarations*). Böyle bir duruma sınıfın tanımlaması yapılmadan o sınıfa referans istekleri

oluştuğu zaman gerek duyulur. Bu sayede sisteme referansyapılan sınıfın mevcut olduğu fakat tanımının daha sonra yapılacağı bildirilir.

- *Boş sınıflar* ( *Empty classes*) : Bu sınıflar hiçbir şey yapmazlar fakat programın gelişimi esnasında anlam kazanacakları için bildirimleri yapılır.

### 3.5. Üye Fonksiyonların Çeşitleri :

Bir nensnenin elemanları olan üye fonksiyonlar bazı kategorilere ayrılabilir. Örneğin;

- *Satıriçi üye fonksiyonları* (*Inline member functions* ) : Sınıf tanımları içinde tanımlanmış fonksiyonlar *satıriçi fonksiyonlar* ( *inline functions* ) olarak adlandırılır. Bir sınıf tanımında b,r fonksiyon tanımlandığı zaman derleyici bu fonksiyona inline tanımlaması ile belirtilmiş gibi muamele eder. Yani derleyici, fonksiyonu inline fonksiyon olarak görür.
- *Sanal fonksiyonlar* ( *Virtual functions* ) : Sanal fonksiyon taban sınıfın tanımında bildirilmiş bir fonksiyondur. Sanal fonksiyondan sınıflar türetmek yoluyla kullanılır.
- *Asıl sanal fonksiyon* ( *Pure virtual functions* ) : Asıl sanal fonksiyon, taban sınıfın tanımı içinde bildirimi yapılış bir fonksiyondur. Fakat sadece türetilmiş sınıflarda kullanılır. Asıl sanal fonksiyon taban sınıfın tanımı içinde bildirilmiş olsa bile bildirimin yapıldığı taban sınıf hiçbir şekilde sanal fonksiyonu kullananmaz.
- *Dönüşüm fonksiyonları* ( *Conversion functions* ) : Dönüşüm fonksiyonları veriyi bir tipten diğerine dönştürebilen özel fonksiyonlardır. Dönüşüm fonksiyonları temel veri tiplerini ve aynı şekilde kullanıcı tanımlı veri tiplerini de işleyebilir. Dönüşüm fonksiyonlarının iki çeşidi vardır: *üye dönüşüm fonksiyonları* (*member conversion functions*) ve *kurucu dönüşüm fonksiyonları* ( *constructor conversion functions* ).

- *Constant üye fonksiyonlar* (*Constant member functions* ) : Bir üye fonksiyon bildiriminden önce *const* kelimesinin kullanılması ile constant üye fonksiyon olarak bildirilir. Bir üye fonksiyon constant olarak bildirildiği zaman fonksiyon değeri değiştirilemeyen bir değişkeni geri döndürür.
- *Statik üye fonksiyonlar* ( *Statik member functions* ) : Statik üye fonksiyonlar, statik üye değişkenler olarak adlandırılan özel bir değişken çeşidini erişmek amacıyla tasarlanmış fonksiyonlardır.

# 4. BÖLÜM

# MICROSOFT VISUAL C++ ORTAMI

Tezin konusunu oluşturan konfeksiyon otomasyon yazılımının gerçeklendiği Microsoft Visual C++ proje geliştirme ortamının kısa tanıtımı bu bölümde yer almaktadır.

## 4.1. Microsoft Visual C++ :

Visual C++ nesne yönelik programlama için geliştirilmiş görsel bir programlama ortamıdır. Visual C++, Görsel C++ programlarının yazımını kolaylaştıran önceden yazılmış bir dizi sınıflarla birlikte gelir. Onun sayesinde IBM Kişisel Bilgisayarlar ve Uyumlu Kişisel Bilgisayarlar için Windows temelli programlar geliştirmek C++'ın eski editör yapılarını kullanarak text temelli programlar yazmak kadar kolay hale gelmiştir.

Visual C++ ile birlikte gelen sınıf kütüphanesi Microsoft Foundation Class(MFC)'nin ikinci versiyonudur. MFC 2 bir dizi C++ sınıfından daha fazlasını içerir. Özellikle görselliğe yönelik yeni sistemleri desteklemek için geliştirilmiştir.

Visual C++ ile birlikte gelen diğer etkileşim destekleri ve Visual C++ MFC kütüphanesinde sağlanan sınıflar sayesinde çalışan bir C++ programını hiçbir kod yazmaksızın oluşturmak mümkün olur. Bunun için yapılması gereken şey Visual Workbench 'te menu parçalarını seçmek ve dialog box'larda bir takım kontrollere tıklamak yoluyla uygulamanın çeşidini belirtmek yeterli olur. Daha sonra da Visual

Workbench ve tam bir uygulama için gerekli olan kodları yazmak maksadıyla diğer Visual C++ aletleri kullanabilir.

## 4.2. Microsoft Visaul C++'ın Özellikleri :

Visual C++'taki yeni destekler aşağıda yer almaktadır:

- Visual Workbench(VWB). Bir kabuk altında Visual C++ editörünü, güçlü debugger'ı, tarayıcıyı ve etkileşimsel destkeleri içeren tümleşik geliştirme ortamıdır.

- AppWizard. Tek bir mouse butununa tıklamak suretiyle tam bir C++ yazılım projesini oluşturabilen bir uygulama geliştiricidir. Visual Workshop'ın *Project* menüsünden *AppWizard*'ı seçerek penceresi ve menu barı olan çalışır halde bir C++ uygulaması oluşturulabilir. Daha sonra da diğer Visual C++ desteklerini kullanarak istenilen kodlar ve kaynaklar eklenerek tam bir uygulama yazılabilir.

- AppStudio. Windows kaynaklarını oluşturmak ve yazmak için etkileşimsel bir grafik desteğidir. AppStudio ile ekranda gördüğümüz dialog box, menü, ikon, bitmap ve kursor gibi kaynaklar tasarlanabilir. Oluşturulan kaynakarın kaynak dosyalarını otomatik olarak üretir.

- ClassWizard. Yeni sınıflar, sınıf üye fonksiyonlarına mesaj dönüşümleri ve sınıf üye değişkenlerine kontrol dönüşümleri oluşturmaya olanak sağlayan etkileşimsel bir destektir. ClassWizard ile sınıf oluşturmak için mevcut sınıfların listesinden bir sınıf taban sınıf seçilir ve yeni sınıfın adı belirtilir. Yeni sınıfı tanımlamak ve gerçeklemek için gerekli olan kaynak dosyaları ise ClassWizard tarafından otomatik olarak oluşturulur. Daha sonra da yeni sınıf geliştirilen programa bağlanır. Programın gelişimi esnasında program için gerekli sınıflar ClassWizard sayesinde oluşturulabilir.

- Visual Workbench Browser. Modüller, sabitler, makrolar, değişkenler, semboller ve sınıflar arasındaki ilişkiler hakkında bilgi elde etmek için kullanılabilen bir tarayıcıdır. Bu tarayıcı sayesinde sınıf hiyerarşileri, fonksiyon çağrı ağaçları ve sembol tanımları ile referansları gözlenebilir.

Yukarıda tanımlanan geliştirme desteklerinden başka Visual C++'ın içerdiği diğer özellikler ise şunlardır:

- Çalışma anı kütüphaneleri.

- Windows kütüphaneleri.

- Windows Software Development Kit (SDK) için Online dokümantasyon.

- Bir programın geliştirilmesi esnasında herhangi bir anda kullanılabilen Windows temelli kaynak satır debugger. *Debug* menüsünden *Go* komutunu seçmek suretiyle Visual Workbench içinde hata ayıklayıcıyı çalıştırmak mümkündür.

### 4.3. Visual C++ Uygulaması :

Bir Visual C++ uygulaması MFC, Microsoft Visual C++ tasarım destekleri ve Visual C++ Windows temelli geliştirme ortamını kullanarak tasarlanan ve geliştirilen bir Windows uygulamasıdır.

Bu tümleşik proje geliştirme ortamını kullanarak görsel arabirim elemanları üstüne yoğunlaşmak yoluyla uygulamanızı geliştirebilirsiniz. Visual C++ görsel arabirim elemanlarını "kullanıcı arabirim nesneleri" (user inreface objects) olarak isimlendirir. İlk iş olarak kullanıcı arabirim nesnelerini tasarlarsınız ve daha sonra gerekli olan diğer program kodlarını yazmak ve değişiklik yapmak için Visual C++ desteklerini

kullanırsınız. Visual C++ desteklerini kullanarak uygulamanızın kaynaklarını tasarlayabilir ve mesajları ele almak için fonksiyonel kodu yazabilirsiniz.

Visual C++'ın text editörü, proje penceresi, tarayıcı penceresi, debugger ve kaynak editörü içermesi nedeniyle Windows Sistem Geliştirme Kiti(System Development Kit - SDK) için Visual C++ desteklerini C ya da C++'da standart uygulamalar geliştirmek amacıyla kullanabilirsiniz.

### 4.4. Sınıf Kütüphanesi :

Microsoft Foundation Class Library(MFC) C++ programcısına Microsoft Windows için uygulama geliştirmesine olanak sağlar. Ayrıca sınıf kütüphanesi programcıya eksiksiz bir "uygulama çalışma çerçevesi" verir. Bu uygulama çalışma penceresi uygulamanın geri kalanı ile Windows için kullanıcı arabiriminin bütünleşmesini tanımlar.

MFC genelde bir uygulama çalışma çerçevesi olarak bilinen C++ sınıflarının bir kümesidir. Bu sınıflar Windows işletim sistemi için bir uygulamanın esas parçalarını ve çalışma çerçevesini sağlar. Çalışma çerçevesinin gayesi Windows uygulamalarını tasarlamak ve gerçeklemek için gerekli çabaları azaltmaktır. Çalışma çerçevesinin bir nesne yönelik sınıf kütüphanesi olması nedeniyle MFC'deki mevcut çalışma çerçevelerinin kullanımı hem kolaydır hem de uygulamayı güçlendirir. Çalışma çerçevesinin kaynak kodunu doğrudan yazmak yerine kütüphanedeki çalışma çerçevelerinden yeni ve özelleşmiş sınıflar türetebilirsiniz. Türetilmiş sınıflar nesne yönelik programlamanın özelliklerinden biri olan miras alma ile taban sınıfının tüm özelliklerini ve fonksiyonlarını miras alır. Ayrıca istenilen diğer yeni üye değişkenler ve fonksiyonlar türetilmiş sınıfa eklenebilir ve miras alınan üye fonksiyonlar istenildiği şekilde değiştirilebilir.

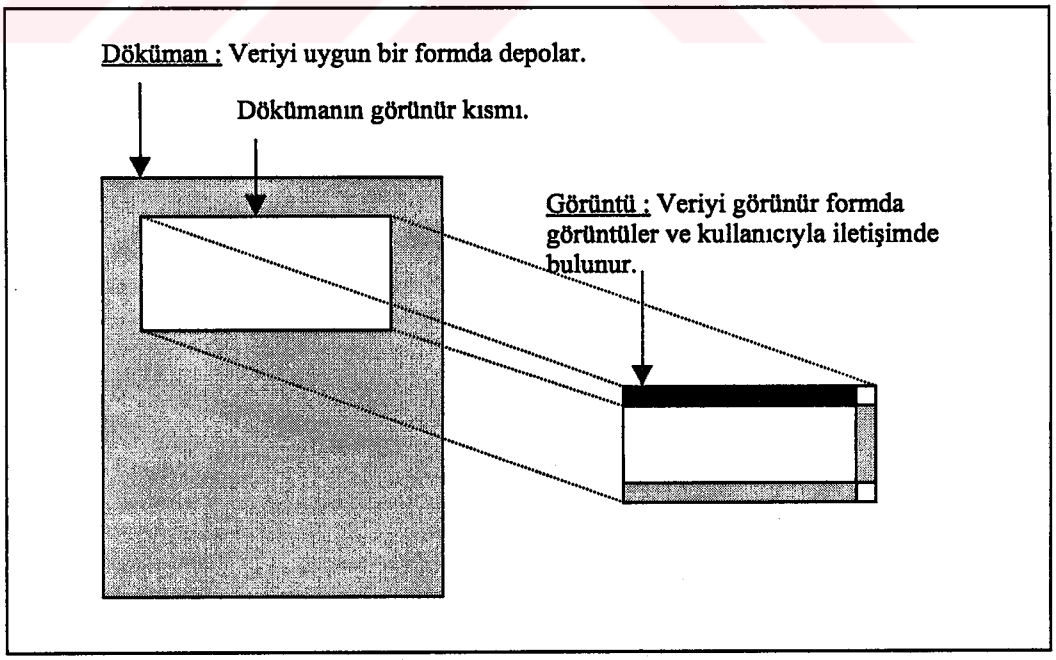
Uygulama çalışma çerçevesinde önemli kavramlar şunlardır:

- Windows uygulamalarında merkezi nesne "uygulama nesnesi"(application object)'dir.
  Uygulama nesnesi bir dizi doküman nesnesini yönetir ve programdaki diğer nesnelere komutları yönlendirir, dağıtır.

- Kullanıcının üzerinde çalıştığı veri birimi bir dökümandır.
  Döküman veriyi yükler, saklar ve güncel tutar.

- Kullanıcı "view" aracılığıyla döküman üzerinde çalışma yapar.
  View bir pencere çerçevesinin çalışma alanına gömülmüş bir penceredir. Bu view dökümanın verisini görüntüler ve eylemlere dönüştürülen mouse ile klavye girişlerini alır.

- Menu ve buton gibi kullanıcı arabirim nesneleri uygulamadaki dökümanlara, view'lara ve diğer nesnelere komutlar gönderir. Bu nesneler komutları yerine getirir.




Şekil 4.1 Bir döküman ve onun görüntüsü arasındaki ilişki.

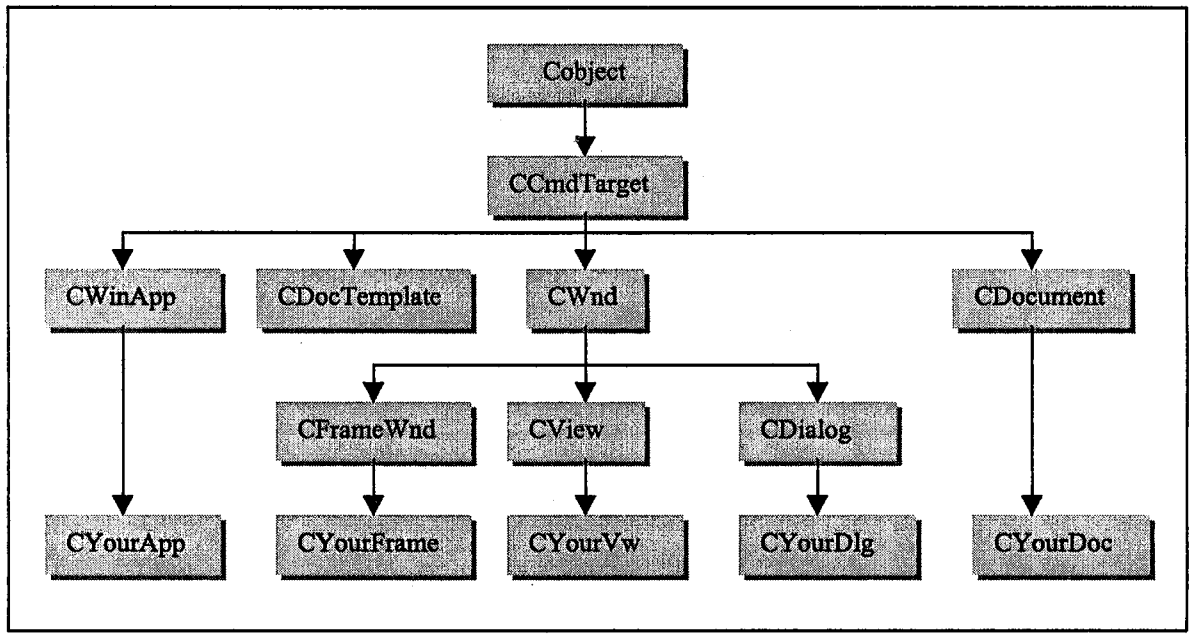
Çalışma çerçevesinin kullanımında bizim yapmamız gerekenler:

- Döküman sınıfında uygulamanın verisini tanımlama.
- Pencere içinde kullanıcının veriyi nasıl göreceğini ve veri ile nasıl etkileşeceğini tanımlama.
- Menüleri, butonları ve diğer kullanıcı arabirim nesnelerini komutlarla bağlama ve daha sonra da komutları yerine getirmek için ele alıcı fonksiyonları(handler functions) tanımlamaktır.

Genel olarak bu şekilde tanımlanan yapının kod yazım aşamasında kullanıcı tarafından gerçeklenmesi gereken yapılar ise şöyle sıralanabilir:

- Dökümanın veri yapsını bildirme ve gerçekleme.
- Dökümanın verisini devamlı kılma. Öyleki bu veri bir dosyaya yazılmak ve okunmak suretiyle bir çalımadan diğerine tekrar tekrar kullanılabilsin.
- Bir view'da veriyi görüntüleme.
- Windows vasıtasıyla alınan klavye ve mouse mesajlarının işlenmesi.
- Menulerden ve destek çubuğundan yöneltilen komutların işlenmesi.
- Çalışma çerçevesi tarafından sağlanan çıkış alma(printing), dilim dilim ilerleme(scrolling) ve pencereyi parçalama(window-splitting) yeteneklerini yükseltmek.

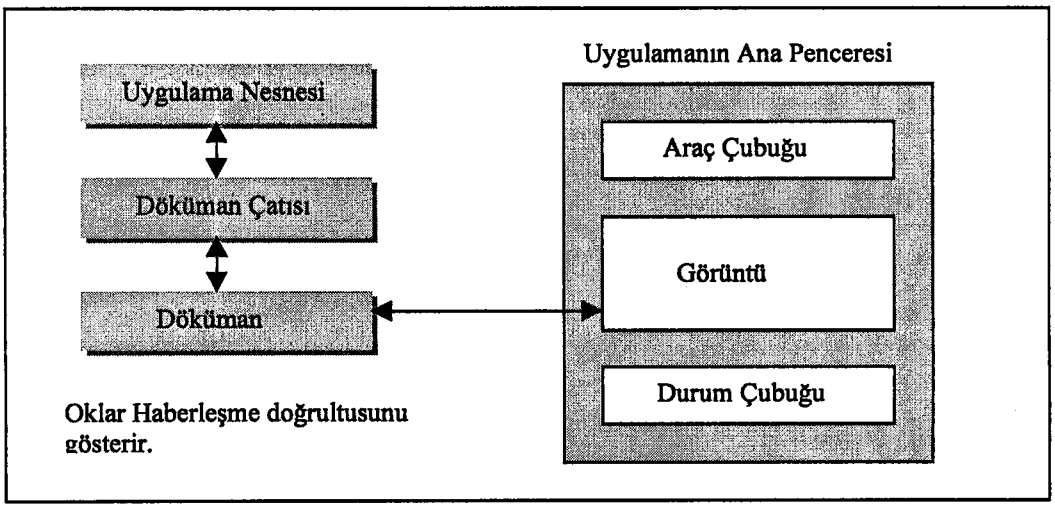
Standart bir uygulamayı oluşturan kod parçalarının çalışma çerçevesine nasıl eklendiği Şekil 4.2'de görülmektedir.




Şekil 4.2. Bir uygulamayı oluşturan temel nesne bileşenlerinin tanımlanması.

### 4.5. Döküman Sınıfı :

Microsoft Foundation Class Library(MFC) ile yazılmış bir uygulama çalışma anında diğer uygulamanın fonksiyonlarını çağırarak ve Windows mesajları göndererek ve alarak haberleşen bir araya gelmiş nesnelerin bir grubudur. Dökümanlar, döküman çatı nesneleri(document template objects) tarafından oluşturulur ve uygulama nesnesi tarafından yönetilir. Kullanıcı view nesnesi aracılığıyla dökümanla etkileşir. Şekil 4.3 bu anahtar nesneler arasındaki ilişkiyi grafiksel olarak gösterir.




Şekil 4.3 Bir uygulamayı oluşturan temel nesneler arası ilişki.

Dökümanın ve diğer nesnelerin bir uygulama çalışma çerçevesinde nasıl oluşturulduğu ve idare edildiği Tablo 4.1'de görülmektdir.

Tablo 4.1. Bir Uygulamadaki Anahtar Nesneler.

| Nesne | Öncelikli Amaç | Diğer Nesnelerle İlişkiler |
|---|---|---|
| Uygulama | Diğer çalışma çerçeve nesnelerinin tümünü idare eder. | Döküman çatılarının listesini güncel tutar. |
| Döküman çatısı | Dökümanları oluşturur ve yönetir. | Verilmiş bir tipten açık dökümanların listesini yönetir. Döküman verilerini bir kullanıcı arabiriminde sunmak için pencere çerçeveleri oluşturur ve görüntüler . |
| Döküman | Veriyi depolar. | Döküman verilerini görüntüleyen görüntülerin (views) listesini idare eder. |
| Görüntü (View) | Bir döküman yardımı ile kullanıcı etkileşimini yönetir. | Bir döküman ile ilişkilendirilir ve bir pencere çerçevesi tarafından sahiplenilir. |
| Pencere Çerçevesi (Frame window) | Bir görüntüyü çerçeve içine alır. | Bir dökümanın ilişkilendirildiği görüntüyü sahiplenir. |

Döküman standart bir Visual C++ uygulamasında kullanıcının File menüsünde Open'ı seçerek açtığı ve Save'i seçerek sakladığı verinin bütününü temsil eden bir birimdir. Döküman, sürekli depolama ortamına genellikle bir disk dosyasına veriyi depolama ve yükleme ile sorumludur. Tipik olarak kullanıcının veriyi idare ettiği bir pencere çerçevesi içinde kullanıcıya sunulur.

Döküman, dökümanın görüntüsü(view) ve pencere çerçevesi arasındaki ilişki genel olarak Şekil 4.1 görülmektedir.

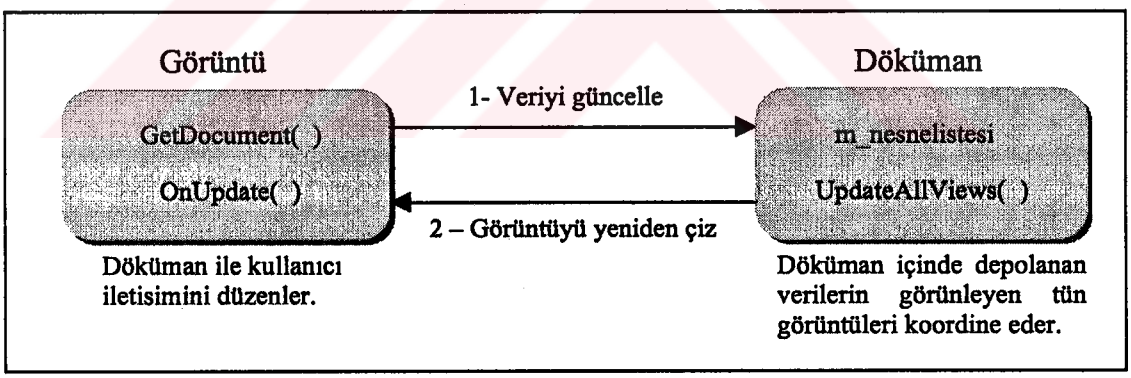
Çalışma çerçevesinde veri ve kullanıcının işlemleri iki ayrı nesne tarafından yerine getirilir. Döküman üye değişkenlerinde program verilerini saklayan, Serialize adı verilen üye fonksiyonu aracılığıyla da bu veriyi okuyup yazabilen bir nesnedir. Görüntü kullanıcının girişlerini ve yazma işlemlerini kabul eden pencere çerçevesinin çalışma alanını doldurur ve veriyi istenildiği şekilde sunar. Dökümanlar veriyi nasıl yöneteceklerini bilirler, görüntüler ise veriyi nasıl görüntüleyeceğini ve işlemleri nasıl yerine getireceğini bilirler.

## 4.6. Görüntü Sınıfı(View Class) :

Döküman uygulamanın verisini tanımlar, depolar ve yönetir. Fakat döküman ilgili tüm etkileşimler bu döküman nesnesi ile ilişkili görüntü nesnesi tarafından yerine getirilir.

*File* menüsündeki *Open* ya da *New* komutu neticesinde yeni bir döküman oluşturulduğu zaman çalışma çerçevesi ayrıca bir "döküman pencere çerçevesi"(documan frame window) oluşturur ve çocuk penceresi olarak pencere çerçevesinin çalışma alanı içinde bir görüntü oluşturur. Oluşturulan bu görüntü dökümanın verisini görüntüler ve mouse, klavye aktivitelerini, menü komutlarını ve kulanıcının döküman üstündeki çalışmalarını yerine getirir. Uygulamaya özel verinin görüntü tarafından kullanıcıya nasıl sunulacağını ve kullanıcı aktivitelerine nasıl cevap verileceğini belirlemek programcının görevidir.

Görüntünün döküman üstünde rolünü açıklayan yapı şekil 4.4'te görülmektedir.




Şekil 4.4. Döküman ve Görüntü.

Şekildeki *GetDocument* fonksiyonu görüntüden dökümana ulaşmayı, *UpdateAllViews* fonksiyonu ise o dökümanla ilişkili tüm görüntülere ulaşmayı sağlar. *OnUpdate* fonksiyonu ise o görüntüde sunulan ve dökümandaki *m_nesneListesi* üye değişkeninde depolanan verinin görüntüsünü günceller.

### 4.6.1. Görüntünün Tanımı :

Görüntü, kullanıcının döküman ilgili aktivitelerini yerine getiren CView sınıfından türetilmiş bir nesnedir. Görüntü özel bir döküman sınıfı ile ilişkilidir ve tipik olarak döküman pencere çerçevesinin çalışma alanını dolduran bir yavru penceredir. Tek döküman arabirim(Single Document Interface SDI) uygulamalarında görüntü ana pencere çerçevesini doldurur. Çoklu döküman arabirim(Multiple Document Interface MDI) uygulamalarında ise döküman pencere çerçevesi uygulamanın ana pencere çerçevesi içinde sırası ile görüntülenir.

## 4.7. Döküman - Görüntü Etkileşimi :

CView sınıfının GetDocument üye fonksiyonunu çağrırarak bir görüntü ilişkili olduğu dökümanın depoladığı veriye erişebilir. GetDocument fonksiyonu ile döküman nesnesine bir işaretçi elde edilir. Bu işaretçi sayesinde görüntü dökümanın tüm public üye fonksiyonlarını çağırabilir ve public üye değişkenlerine erişebilir.

Kullanıcı görüntüdeki veriyi değiştirdiği zaman görüntü bu değişikliği dökümana bildirir ve dökümanın depoladığı veriyi günceller. Bu değişikliği haber alan döküman, UpdateAllViews üye fonksiyonu ile değişikliği onunla ilişkili tüm görüntülere iletir ve veriyi tekrar çizmelerini sağlar.

# 5. BÖLÜM

## KALIP HAZIRLAMA ve PASTAL YERLEŞİM PROBLEMİ

Kalıp hazırlama ve pastal yerleşim problemi Gemicilik, Dericilik, Konfeksiyon, Camcılık gibi bir çok meslek dalında karşılaşılan problemlerdendir. Meslek dalına göre bu problemler bazen küçük bazen de büyük farklılıklar gösterir ama ana temalar arasında büyük benzerlik vardır. Bu tezin özünü oluşturan çalışma ise Konfeksiyon sektörünü baz almaktadır. Burada anlatılacak konular temel alınarak yukarıda adı geçen sektörlerde de problemlerin çözümüne yönelik çalışmalar yapılabilir.

Konfeksiyon sektöründe kalıp tasarım işi ile uğraşan tasarımcılar kişilerden aldıkları ve belli bir birikimle standartlaşan ölçüm değerlerine göre herkese uyabilecek bedenlerde kalıpları dizayn ederler. Kalıp tasarımı yapan kişi herşeyden önce belli bir deneyim sürecinden geçmiş olmalı ve aldığı beden ölçülerini nasıl değerlendireceğini bilmelidir. Günün modasına uyan renk, desen ve şekle göre tasarımlar yapılacağı gibi yöresel ihtiyaçlara cevap veren tasarımlarda yapılabilir. Bu açıdan bakıldığında tasarımlar daha ziyade piyasa talebini karşılayacak şekilde gerçekleştirilir. Tasarımda ince eleyip sık dokumak ve az sayıda sonuç kalıp çıkarmak yerine tasarımı gerçekleştirilen pek çok model arasında en uygun olanları seçmek piyasada kabul görmüş bir yaklaşımdır P. Taylor[7]. Seçilen modellerin kopyaları kağıttan, plastikten vs. yapıldıktan ve son şeklini aldıktan sonra hangi tasarımların piyasaya sürüleceği kararı tabii ki sorumluluğu üzerinde taşıyan kişi tarafından verilir. Bu kişi sermaye sahibi şirket yöneticisi, üst düzey bir amir olabilir.

Piyasa koşullarına uygun tasarımlar belirlendikten sonra, bu tasarımlar en ekonomik şekilde nasıl gerçekleştirilebilir sorusu ortaya çıkar. Ekonomik ve verimli bir üretim

için temel şart ise eldeki malzemeyi minimum sarfiyat ile maximum gelire dönüştürmektir. Bunun için aynı kumaş malzemesinin kullanılmasına karar verilen kalıplar en uygun şekilde o kumaş topuna yerleştirilmeye çalışılır. Kalıpların kumaş topuna yerleşim problemi literatürde pastal yerleşim problemi olarak geçmektedir. Bu problemin çözümünü sağlayan tüm yöntemlerde temel nokta, kalıpları belli bir kumaş genişlik ve uzunluğuna sahip alana minimum sarfiyat ile yerleştirmektir. Genelde kumaş uzunluğu sabit değildir ve yerleşim neticesinde ortaya çıkar. Böyle bir yerleşim planı elle tek tek kalıpları yerleştirmekle gerçekleştirilebileceği gibi belli bir mantığa sahip algoritmalar ile de gerçekleştirilebilir. Elle yerleştirmede bu işi yapan kişi uzun bir deneyimin getirdiği alışkanlıkla kalıpları tek tek kumaş alanına yerleştirir. Fakat elde edilebilecek verim belli sınırların üstüne çıkamaz. Bu işi gerçekleştiren algoritmaların kullanımında ise çeşitli kıstaslar verimi etkiler. Algoritmadan ne kadar iyi verim istenirse o ölçüde de fedakarlıkta bulunmak gerekir. İyi bir sonuç elde etmek için ya uzun bir zaman dilimi ya da fazlaca sistem kaynağı harcamak gerekir. Elde edilecek sonuç yerleşimde bizim gözle görebileceğimiz fakat algoritma yapısının kolayca tespit edemiyeceği bazı eksiklikler oluşabilir. Bu tip yerleşim planı hazırlayan algoritma yapıları kendine göre geniş bir araştırma sahası oluşturmaktadır ve bu konuda çeşitli pek çok akademik çalışma gerçekleştirilmiştir. Görüldüğü gibi iki yönteminde kendine göre zayıf noktaları vardır. Öyleyse sonuç yerleşim planı iki yönetimin de karışımı olmalıdır.

Bu noktada kalıp hazırlama konusunu, baz aldığımız konfeksiyon sektörüne uygun olarak biraz daha detaylı incelemek yararlı olacaktır.

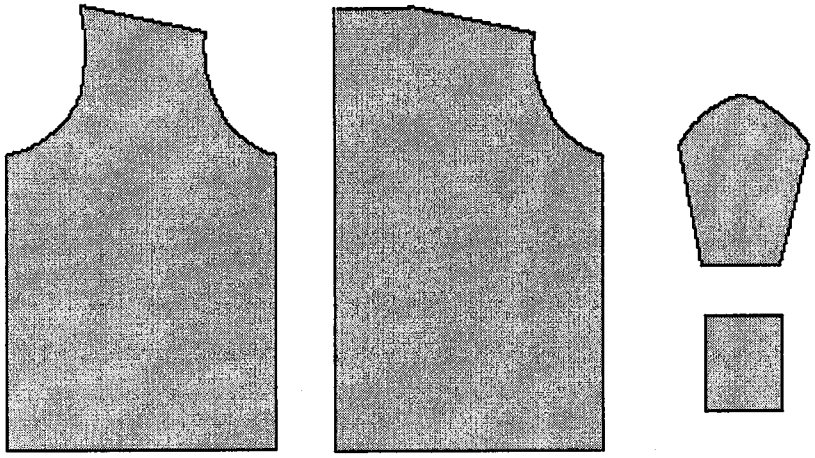
## 5.1. Kalıp Hazırlama Konusu :

Konfeksiyon sektöründe standartlaşmış olan veyahut özel siparişlerde şahıstan alınan ölçüm değerlerine göre önce göze hitap eden tasarım yapılır. Ardından da hazırlanan kalıbın dikime yönelik olarak son şeklini alması için kalıp üzerinde bazı çalışmalar yapılır. Eğer aynı kalıbın farklı ölçülerde benzerlerini oluşturmak gerekir ise o zamanda kalıbın üzerinde serilendirme işlemleri yapılır. Anlattığımız bu yapıyı daha iyi açıklayabilmek için kalıp tasarımının kavramlarını tek tek tanıtmakta yarar vardır.

### 5.1.1. Kalıp Hazırlama Kavramları :

#### 5.1.1.1. Model :

Tasarımda amaçlanan giysi, modeli oluşturur. Bu açıdan bakıldığında model ceket, pantalon, t-shirt vs. olabilir. Eğer tasarım bürosunun birlikte çalıştığı bir çok firma var ise her bir firma için gerçekleştirilen modeller ayrı ayrı saklanır. İstenirse ortak bir modeller kütüphanesi oluşturulur ve tasarım esnasında bu kütüphanedeki modellerden ilham alınabilir. Örnek bir model şekil 5.1'da yer almaktadır.



Şekil 5.1. T-shirt Modeli (Ön, arka, kol ve cep kalıpları.)

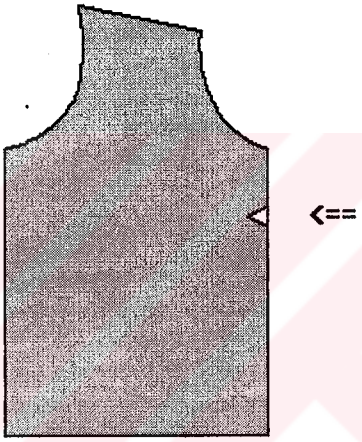
### 5.1.1.2 Kalıp :

Giysi modelini oluşturan her bir temel çizim parçası kalıp adını alır. Dolayısıyla bir model genelde birden fazla kalıba sahip olur. "Kalıp tasarımı" kelime anlamıyla

ve dikime hazır hale getirilen kalıplar birbirine dikilerek veya iliştirilerek modellere dönüştürülür. Şekil 5.1'de yer alan modeldeki her bir çizim parçası bir kalıba

### 5.1.1.3 Çıt :

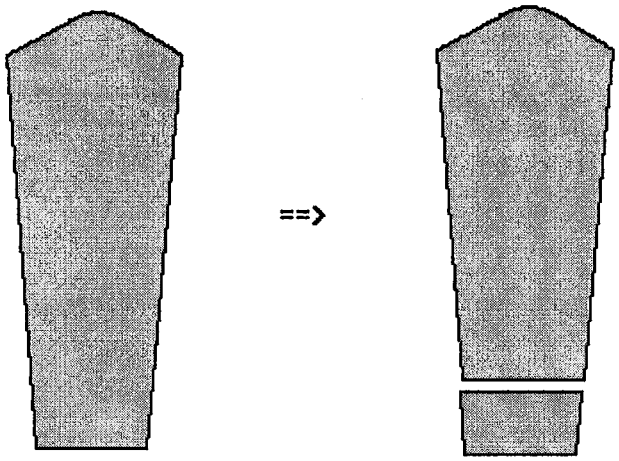
Çıt, tasarımı gerçekleştirilen kalıp kenarlarının bir parçası olan ve kenarların karakteristiğini ( eğri veya düz karakteristiği ) bozmayan özel bir noktadır. Çeşitli tipleri mevcuttur. Ayrıca müşteri firmanın isteğine uygun olarak özel tipleri de tanımlanabilir. Çıtın esas işlevi kesim esnasında kullanılmak üzere kalıp kenarlarına bazı işaretler bırakmaktır. Kalıpların kesimini gerçekleştiren kişiler kesim sırasında çıtın tipine göre o noktalara bazı küçük kesikler atar. Bu kesikler dikimde dikkate alınır. Şekil 5.3'de bir çıt örneği görülmektedir.




Şekil 5.3. Bir çıt örneği

### 5.1.1.4. Pervaz :

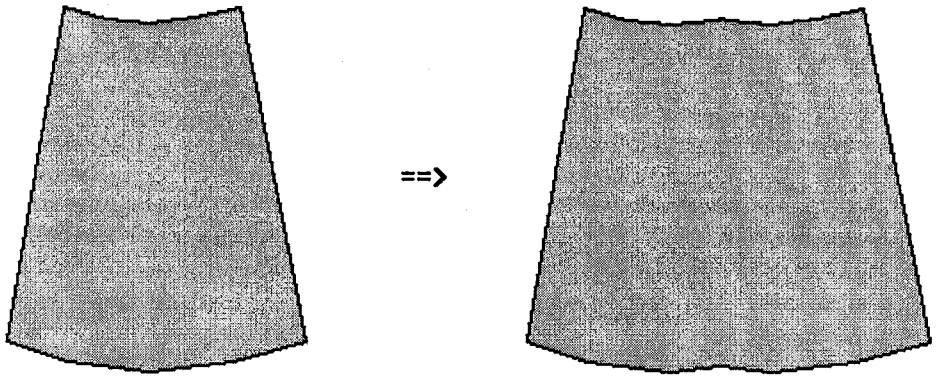
Belli bir genişlikte kalıp kenarlarına paralel olarak bazı parçaların kalıptan çıkarılmasını temel alan bir yaklaşımdır. Böyle bir çalışmaya örnek gömlek kollarında gösterilebilir. Gömlek kolları tasarlandığı ilk anda tek parçadır. Daha sonra kol kalıbının ele yakın uç kısmında kullanımı geleneksel hale gelen manşet diye tabir edebileceğimiz ve dayanıklılığı arttıran bir parça kol kalıbından çıkarılır. Bu işlem ise pervaz ile sağlıklı bir şekilde gerçekleştirilir. Şekil 5.4'de söz konusu gömlek kolundaki pervaz örneği görülmektedir.




Şekil 5.4. Gömlek kolundan pervaz çıkarılması.

**5.1.1.5. Pile :**

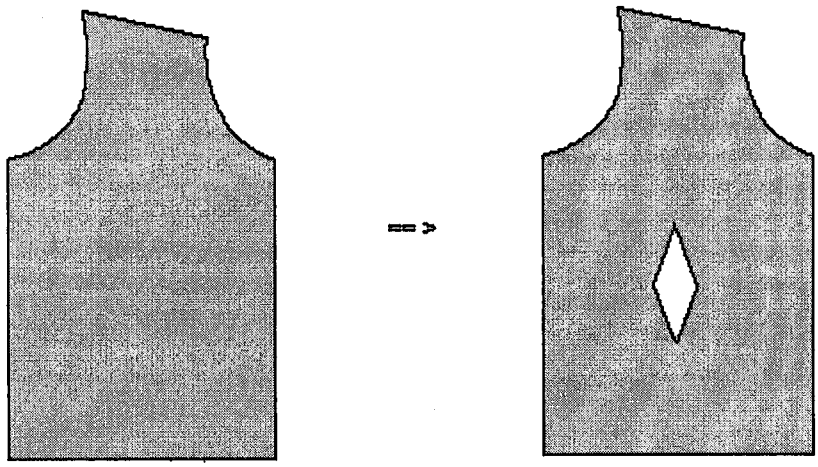
Kalıp üzerinde birbiri üzerine katlanma kenarları, büzgüler oluşturan bir yapıdır. Tasarımı bitmiş yani tasarımı temsil eden kenarları artık değişmeyecek olan bir kalıba uygulanır. Pilenin bir doğrultusu ve bu doğrultuyu kesen iki kalıp kenarı vardır. Bunun için pile tanımı yapılırken öncelikle pilenin temel bileşeni olan pile doğrultusu, iki kalıp noktası yardımı ile tespit edilir. Pilesiz kalıp şekli ile dikim esnasında pile doğrultusundan katlanarak iki kenarı birleştirilen kalıbın iki boyutlu şekli genellikle birbirinin benzeridir. Dolayısıyla pileye sahip bir kalıbın esas görünüş yapısı dikimden sonra belli olur. Bu açıdan bakıldığında tasarımı yapılmış kalıp üzerinde pilenin dikim sonucunda nasıl bir şekil alacağını hayal edebilmek oldukça güçtür. Tasarım esnasında pile noktaları çıt ile işaretlenir ve dikim esnasında da bu çıt noktaları dikkate alınarak pile doğrultusu ile dik olan kenarların biri veya her ikisi de dikilir. Sağa veya sola kat oluşturan pileler, üst veya alt kısmı katsız olan pileler, kanun diye tabir edebiliceğimiz özel bir pile tipi de mevcut pile tipleri arasında örnek olarak gösterilebilir. Birden fazla sayıda pile birbirinden belli bir uzaklıkta ve belli bir pile genişliğinde yer alabilir. Şekil 5.5'de tipik bir pile örneği görülmektedir.



Şekil 5.5. Tipik Bir Pile Örneği (Sola Kat Oluşturan Pile)

**5.1.1.6. Pens :**

Pens pileye benzer bir yaklaşımdır fakat dikim esnasında pileden farklı olarak pens doğrultusu ile aynı olan kenarlar birbirine dikilir. Ayrıca pens tüm bir doğrultu boyunca değil sadece belli bir pens uzunluğunda etkili olur. Dikime getirdiği en önemli yenilik kalıba üçüncü boyutta da çalışmayı eklemesidir. Zaten esas kullanılış amacı da budur. İç pens, dış pens gibi çeşitleri vardır. Örnek bir pens yapısı şekil 5.6'da görülmektedir.

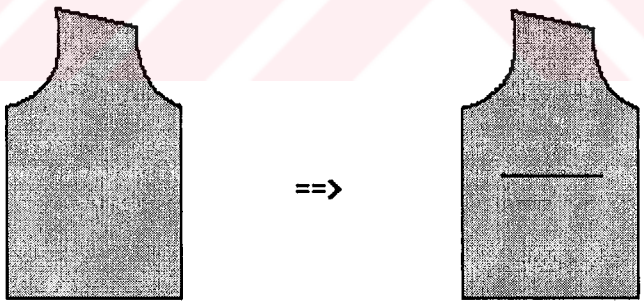


Şekil 5.6. Bir Pens Örneği.(İç pens)

Modellerin tasarımı bittikten sonra kalıpların kumaş topu üzerine yerleştirilmesi işlemine geçilir. Burada bahsedilen kalıpların yerleşim planını hazırlama işlemi pastal hazırlama adını alır. Böyle bir çalışmanın gerçekleştirilebilmesi için kalıplar üzerinde bazı işlemlerin yapılması gerekir. Devamda yer alan kavramlar bu amaca yönelik çalışmaları anlatır.

### 5.1.1.7. Kalıp Düz İpliği :

Pastal yerleşim planı hazırlanırken kalıplar kumaş üzerine istenildiği gibi yerleştirilemez. Kalıpların kumaş üzerine belli bir doğrultuda yerleştirilmesi gerekir. Bu amaçla "düz iplik" diye tabir edeceğimiz bir işaret kalıplara eklenir. Düz ipliğin doğrultusu ve yönü vardır. Yerleşim planında da kalıpların düz iplik doğrultusu ile kumaş doğrultusunun aynı olmasına çalışılır. Düz ipliğin kullanılmasındaki amaç kumaşın esneme yönünü, desenlerin doğrultusunu da dikkate almaktır. Düz ipliğin kalıp tasarımında kullanımına ilişkin bir örnek Şekil 5.7'de görülmektedir.

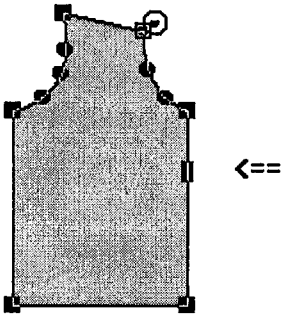



Şekil 5.7. Bir Düz İplik Örneği.

### 5.1.1.8. Çakışma Noktası :

Desenli kumaşların kullanımında söz konusu olan yardımcı bir işarettir. Modeli oluşturan kol ve gövde kalıpları gibi birbiri ile yanyana gelecek parçalardaki desenler arasında geçişin düzgün olması için kullanılır. Pastal yerleşim planında bu çakışma

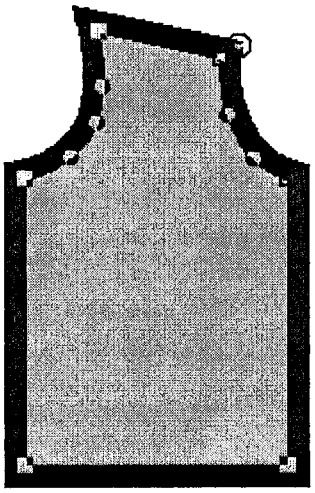
noktaları değerlendirilir. Şekil 5.8'de çakışma noktasının kullanımına ilişkin bir örnek görülmektedir.




Şekil 5.8. Çakışma Noktası Örneği.

### 5.1.1.9. Dikiş Payı (Seam) :

Tasarıma uygun olarak kalıpların birbirine düzgün dikilebilmesi için kalıp profili etrafında bazı payların bırakılması gerekir. Bu paylara dikiş payı adı verilir ve çeşitli tipleri mevcuttur. Eğer kalıp üzerinde başka bir değişiklik yapılmayacak ise en son işlem olarak kalıba dikiş payı uygulanır. Dikim esnasında da bu paylar büyük bir fayda sağlar. Tasarımı bitmiş bir kalıp üzerinde dikiş payının nasıl olacağını gösteren bir örnek şekil 5.9'da görülmektedir.



Şekil 5.9. Dikiş Payı Çalışması.

### 5.1.1.10. Temel Beden ve Beden Setleri

Toplumu oluşturan bireylerin vücut ölçü değerleri çok çeşitlidir. Bu kadar çeşitlilik içinde tasarımlarda kullanılacak ölçü değerleri nasıl tespit edilecektir. Bu işlem için en akılcı yol ölçüm değerleri üzerinde gerçekleştirilen istatistikleri kullanmaktır. İstatistiklere göre normalde insanların boyları arttıkça bel, kalça, dirsek ve bilek gibi vücut ölçü değerleri artmaktadır. Bu açıdan bakıldığında toplumda çoğunlukta bulunan vücut ölçü değerleri standart alınarak ilk model tasarım ölçüleri tespit edilir. Belirlenen ölçülere göre tasarımı yapılan bu modele temel beden adı verilir. Tüm bireylere uygun modelleri gerçeklemek için bu temel bedene göre diğer beden setleri tanımlanır.

Satış pastasından mümkün mertebe daha çok pay alabilmek için hedef kitlemiz tüm toplumdur. Öyleyse çoğunluktaki bedenlere uygun modeller daha çok üretilmeli ve diğer bedenlerin dağılımı da istatistiklere göre belirlenmelidir.

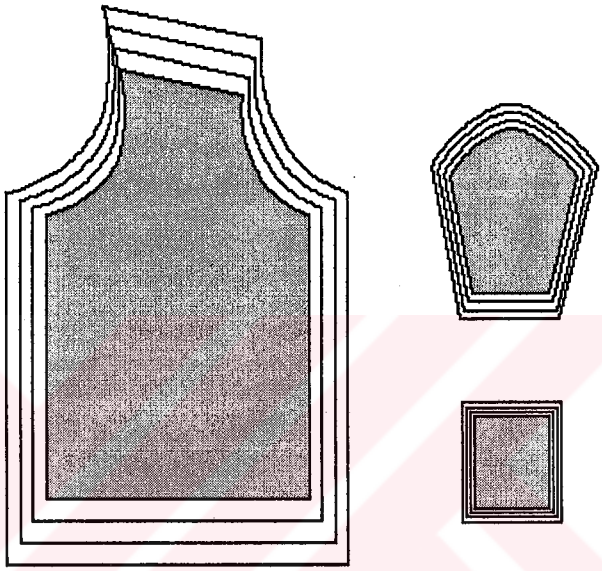
### 5.1.1.11. Serilendirme İşlemi (Grading) :

Gerçek hayatta bir modelin kendisine pazar payı bulabilmesi için o modelin farklı ölçülerde benzerlerini üretmek gerekir. Yani tüm bireylere uygun ölçülerdeki modelleri üretebilmek için o modelin temel bedeni dikkate alınarak diğer beden setleri tanımlanmalıdır. Bu amaçla temel bedenden diğer bedenlere geçişlerin ne olacağı belirtilir. Geçiş değerlerinin tanımlanması işlemine serilendirme adı verilir. Serilendirmede esas parça kalıptır ve işlem kalıplar üzerindeki noktalarda gerçeklenir.

İsteğe göre temel bedeni oluşturan model kalıplarının bazıları, bazı beden setlerinde üretilmeyebilir. Örneğin bir pantolon modelinin temel bedeni 36 numara olsun. Kalıplar 32, 38, 40 numaralı beden setlerinde üretilecek fakat 38 numaralı bedende tek değil iki cep yapılacak olsun. Böyle bir çalışmayı serilendirme işlemi esnasında tanımlamak mümkündür. Çünkü serilendirme işleminde farklı beden setlerini

üretmek amaçlanırken işlem kalıplar üzerinde gerçeklenir. Bu sayede de kalıpların beden setlerindeki dağılımı istenildiği gibi ayarlanabilmektedir. İsteğe göre modeli oluşturan kalıpların -ve hatta kalıp köşelerinin - beden setleri arasındaki geçiş değerleri farklı olabilir.

Şekil 5.10'da modelin temel bedeni ve model kalıpları üzerinde serilendirme işleminin neticesinde oluşan beden setleri görülmektedir.



Şekil 5.10. Model Kalıplarının Serilendirilmesi.

## 5.2. Pastal Yerleşim Konusu :

Modeller tasarlandıktan sonra daha önce de tanımladığımız gibi toplu üretim için pastal yerleştirme işlemine tabi tutulur. Ama pastal yerleşim planı hazırlanmadan önce aynı kumaş malzemesi kullanılacak olan kalıplar pastal siparişi şeklinde düzenlenir. Şimdi bu anlattığımız tanımlamaları daha detaylı inceleyelim.

### 5.2.1. Pastal Siparişi :

Pastal yerleşim planını hazırlamaya yönelik bir ara çalışma safhasıdır. Bu safhada hangi kumaş malzemesinin kullanılmak istendiği ve hangi kalıplardan kaç adet üretilmek istendiği bellidir. Yapılmak istenen bu iki parametreyi uygun şekilde düzenlemektir. Düzenlemeden sonra pastalda kaç kat kumaş kullanılacağı ve assorti diye tabir ettiğimiz kalıp adetlerinin sayısı belli olur.

Örneğin mavi düz kumaşa Gömlek Ön kalıbının S (small) bedeni 25 adet, M (medium) bedeni 15 adet ve L (large) bedeni 20 adet gönderilmek isteniyorsa pastal siparişi neticesinde şu değerler elde edilir. Kullanıcak kalıp adetlerinin EBOB(en büyük ortak böleni) olan 5 sayısı pastaldaki kumaş katının sayısını belirlerken, assorti değerleri S beden için 5 adet, M beden için 3 adet ve L beden için 4 adet olur. Anlatılan hesabın tablo dökümü aşağıdaki tablo 5.1'de görülmektedir.

Tablo 5.1. Pastal Siparişi.

Gömlek ön kalıbı :

| Beden Setleri : | S (small) | M (medium) | L (large) | XL (extra large) |
|---|---|---|---|---|
| Malzeme : Düz mavi kumaş | 25 adet | 15 adet | 20 adet | 0 |

Pastalda kullanılacak kumaş katının sayısı : 5

Assorti Değerleri :

| Beden Setleri : | S (small) | M (medium) | L (large) | XL (extra large) |
|---|---|---|---|---|
| Malzeme : mavi düz kumaş | 5 adet | 3 adet | 4 adet | 0 |

### 5.2.2. Pastal Yerleşimi :

Pastal siparişi yapıldıktan sonra pastal yerleşim planı hazırlanabilir. Pastal hazırlamada en ekonomik olacak şekilde kalıplar kumaş üstüne serilir, çizgiler, kareler ve desenlerin gerektiğinde denk getirilmesine çalışılır ve kalıp parçaları birbirine tutturulur.

"Maliyet" kullanılan kumaş miktarı ile belirlenir ve sonuç "yerleşim planı, pastal remi - kesim planı" olarak adlandırılır , bu kumaş üstünde kalıpların resmidir P. Taylor[7]. Modelin üretim safhasına gelindiğinde elde edilen yerleşim planı ve hangi kumaştan kaç kat kullanılacağı gibi gerekli diğer bilgiler kesim kısmına iletilir. Kesim işlemi, bu işi gerçekleştiren deneyimli elemanlar tarafından veya cutter diye tabir ettiğimiz kesici makinalar tarafından yapılır.

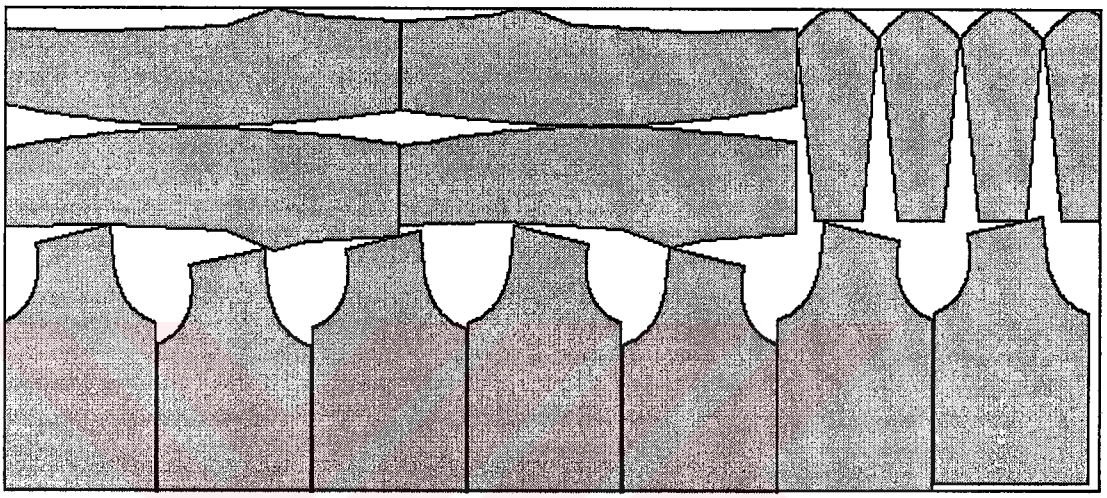
Pastal resmi, belli ebatlardaki kumaş üstüne elimizdeki kalıpları yerleştirmek suretiyle elde edilir. Bu yerleşimde önemli olan faktörler kumaşı minimum ölçülerde sarfetmek, kumaşın doğrultusu ile kalıp düz ipliği doğrultusunu aynı tutmaya çalışmak sayılabilir. Ayrıca birbirine iliştirilecek kalıplarda desenler arası geçişi uyumlu kılmak önemlidir. Bunun için daha önce bahsettiğimiz çakışma noktası özelliğinden yararlanılır.

Pastal resminde kalıplar elle tek tek yerleştirilebileceği gibi bu işi gerçekleştiren algoritmalardan da yararlanılabilir. Bölümün giriş kısmında bahsettiğimiz gibi nihayi pastal resmi iki yaklaşımın karışımıdır. Bazen pastala uydurmak için kalıplar üzerinde bazı değişiklikler yapılabilir. Mesela kalıp noktalarının yerleri değiştirilebilir, kalıplar düz ipliği ile döndürülebilir, kalıplar kesilebilir ve bu kesime uygun olarak dikiş payı eklenebilir. Pastal resmi tespit edildikten sonra oluşan boşlukları tamamlamak için farklı model kalıpları resme dahil edilebileceği gibi siparişteki bazı kalıplardan fazlaca kullanılabilir.

Ayrıca kesim esnasında bu işi gerçekleştiren kişilere bazı toleranslar sağlamak ve diğer sebeplerle kalıpların etrafında bazı küçük mesafeler bırakılır. İstenirse bu mesafeler arttırılabilir. Yerleşim planı belirlendikten sonra kumaş uzunluğunun ne

kadar olduğu, yerleşimde elde edilen verimin ne olduğu gibi bilgiler yerleşimin bir kopyası ile birlikte dosyalanır. Maliyet hesabında bu tip bilgiler değerlendirilir. Ayrıyeten daha sonraki pastal hazırlama işlemlerinde de bu bilgilerden yararlanılabilir.

Hazırlanmış bir pastal resmi örneği Şekil 5.11'de görülmektedir.



Raporlama : Kumaş Boyu : 5150.0 mm.
Verim : 79 %

Şekil 5.11. Pastal Resmi.

# 6. BÖLÜM

## KALIP TASARIM ve PASTAL YERLEŞİM PROBLEMİNE BİLGİSAYAR DESTEKLİ ÇÖZÜM

Kalıp tasarımı ve pastal hazırlama problemlerini bilgisayar desteği ile çözmeye yönelik çalışmalara temel oluşturan ilk uygulamalar Amerikalı Howard Hughes tarafından yapılmıştır. Howard Hughes iki boyutlu uygulamalara çözüm getirmek için bilgisayar kontrollü donanımları içeren geliştirme programını başlatan ilk kişidir. Çalışmanın temelinde yatan fikir herhangi bir çizim yada kesim sürecinin düz yüzeyler üzerinde gerçekleştirilmesidir.

Bu çalışma 1960 yılında Hughes Araştırma Şirketinde başlamıştır. Aynı sene, çalışan bir lazer halka tanıtılmıştır. Sekiz yıl sonra 1968'de Genesco ile çalışan Hughes, lazer ışınıyla çalışan ve o güne kadar kullanılan klasik yöntemlerin, hız ve doğruluk bakımından çok üstünde olan bilgisayar kontrollü kumaş kesme makinasını başarıyla geliştirmiştir.

Geliştirilen teknolojik ilerlemenin, giyim imalattındaki bu tip uygulamaları daha esnek kılacağı dolayısıyla daha güzel ve kullanışlı giysiler üretilebileceği, üretimin çok daha hızlı gerçekleştirilebileceği fark edilmiştir. Giyim sektöründe nesiller boyu tecrübesi olan Autographics'in de yardımıyla Hughes çok zor iki üretim süreçi olan "serilendirme" ve "yerleşim planlama" süreçlerini kolaylıkla başaran bir sistem tasarlamıştır. Hewlett Packard'ın çok güçlü ana bilsayarı tarafından çalıştırılan AM1 adıyla anılan bu sistem giyim imalatçıları için hazırlanmış ilk BDÜ(Bilgisayar Destekli Üretim) sistemidir.

On yıl süresince Hughes Apparel Systems'in sürekli geliştirip sattığı AM1, 1978 - 9 yıllarında diğer bir Amerikan firması olan Gerber tarafından satın alındı. Gerber firması bilgisayar kontrollü toplu kumaş - kesim makinaları geliştirilmesinde uzmanlaşmış ve patenti ile telif hakkını aldığı sistemi kurmuştu. Yerleşim planlama için bir BDT / BDÜ sistemi olan Hughes'in AM1 sistemi, Gerber tarafından toplu kesim amacıyla kullanılmıştı. Gerber AM1'i geliştirmeye devam etti ve onun modern bir sürümü olan AM5'i üretti. Her iki uygulama da Hewlett Packard ana bilgisayar tabanlıydı ve IBM tabanlı bilgisayarlarda da çalıştırılabilmekteydi.

Bu sırada kurulan diğer iki firma daha Amerikan Camsco firması ile Fransız Lectra sistemleri idi. Lectra geliştirme çalışmalarına 1975 de başlamış ve ilk sistemlerini 1978 de pazarlamıştır. Lazer hazırlayıcı geliştirilmesinde de uzmanlaşmışlar ve karton kalıp yada tek katlı kumaş veya deri kesebilen lazerli sistemleri ile birlikte pazarlamışlardır. Firmaların her ikisi de BDÜ sistemleri (serilendirme ve yerleşim planlama) üretiyorlar ve ana sistemle (grafik ve giysi kalıp hazırlama yazılımı) birlikte çalışacak BDT yazılımı geliştiriyorlardı. Gerber firması rekabeti önlemek ve bugünkü kadar güçlü olmayan pazarda tekel oluşturabilmek için 1980 lerin başında Camsco firmasını satın almıştı.

Bu gelişme iki büyük firma olan Gerber ve Lectra'yı dar bir pazarda çalışma durumunda bıraktı. İki firmanın yoğun satış programlarından sonra talep artmaya başladı ve 1980 lerin ortasında sektörün hızlı gelişen bir pazar görümünü alması sebebiyle diğer bilgisayar imalatçıları da dikkatlerini bu alanda yoğunlaştırmaya başladılar. Investronica isimli bir ispanyol firması güçlü bir şekilde pazara girdi. Kalite ve fiyat açısından diğerlerine denk bir bilgisayar sistemi imal ettiler ama Hughes / Gerber sisteminin bir kopyası olduğunu düşünebilirdi. Esasında Investronica kendi donanımlarını üretip kendi yazılımlarını yazmaktaydı ve birinin diğerini kopyaladığını söylemek mümkün değildi. Çünkü 1960 larda tasarım konusunda, Hughes firması tarafından ortaya atılan temel prensiplerin dışına çıkmak mümkün değildi Patrick J. Taylor [7]. Bu açıdan bakıldığında bu alanda gerçeklenen uygulamaların Hughes / Gerber sistemine benzemesi normaldi.
Bizim çalışmalarımız ise giyim sektöründeki kalıp tasarımına ve pastal hazırlamaya bilgisayar desteği ile çözüm getirme konusunda yoğunlaşmıştır. Bu çalışmayı

gerçekleştirirken son teknolojik gelişmelerin kullanılmasına gayret edilmiştir. Çalışmamızda temel alınan proje geliştirme ortamının bileşenleri şunlardır :

- İşletim sistemi : Windows 95
- Proglama ortamı : Visual C++ 4.00
- Veritabanı : Access for Windows 95

Projemiz Windows 95 altında Visual C++ programlama ortamı ile gerçekleştirilmiştir. Dolayısıyla çoklu döküman arabirimi (MDI Multiple Document Interface), çoklu çalışma, kesme, kopyalama, yapıştırma, mouse kullanımının esnekliği gibi Windows programlamanın getirdiği pek çok yenilikler projeye eklenmiştir. Ayrıca Windows işletim sistemi kullanılarak çoklu çalışma (multitasking), görsel kullanıcı standartları (visual standarts), çoklu ekranla çalışabilme, nesne bağlama ve gömme (OLE Object Linking and Embedding) ve genel sürücü destekleri projeye kazandırılmıştır. Visual C++ programlama ortamı kullanılarak daha hızlı, güvenilir program geliştirmeye çalışılmıştır. Bu bileşenlerin kullanımının getirdiği yeniliklerden bölüm 2, 3 ve 4'de daha detaylı bahsedilmiştir.

Projede mevcut program parçaları arasında veri transferini sağlamak, verileri saklamak ve daha sonra tekrar alabilmek, bu işlemleri yaparken mümkün mertebe daha az depolama ortamı harcamak gibi nedenlerle Access for Windows 95 veritabanı ortamı kullanılmıştır. Windows ortamında etkin ve kolay kullanılabilir olan Access for Windows 95'in projede kullanımı ile aşağıda sıralanan temel veritabanı işlemleri sağlanmıştır ;

- Veritabanı sistemi içine yeni boş dosyalar eklemek
- Varolan dosyalara yeni veri eklemek
- Varolan veriyi almak
- Varolan veriyi güncellemek
- Varolan veriyi silmek
- Veritabanı sistemindeki dosyaları silmek veya boşaltmak.

Access yeni bir veritabanı programı olduğu için bütün yeni veritabanlarında olduğu gibi ilişkisel veritabanı (relational database) yöntemini kullanır. İlişkisel veritabanı ile düz veritabanı arasındaki en temel fark ilişkisel veritabanındaki uygun bir tasarım ile tekrarlanan verinin en aza indirilmesinin mümkün olması ve verinin daha kolay

işlenebilir olmasıdır. Düz veritabanı kullanılarak bir sipariş modülü yaratıldığını düşünülürse, her bir sipariş için bir satır kullanarak kayıtlar tutulur fakat aynı firma için verilen her siparişte firma ile ilgili adres, isim gibi detaylar tekrarlanmak zorunda kalır. İlişkisel bir veritabanında ise ayrı bir tabloda tutulan firma bilgileri tekrarlanmayacaktır. Sipariş tablosunda yalnızca firmanın sıra kodu geçecektir. Firma ile ilgili bilgiler başka bir tabloda tutulacaktır. Bir firmanın adresi ile ilgili bir değişiklik yapmak gerektiğinde düz veritabanında her kaydı değiştirmek gerekecektir. İlişkisel veritabanında ise tek bir değeri değiştirmek yeterli olacaktır.

Bu bileşenlerden hareketle projemiz tasarlanmış ve gerçekleştirilmiştir. Projede temel aldığımız bazı kabuller ise şunlardır ;

- Kullanıcı arabirim öğelerinde yer alan mesafe, alan vs. ile ilgili tüm sayısal değerler mm. ( mili metre ) şeklinde tanımlanmıştır.

- Kullanıcının klavye ile girmesi gereken değerler iletişim pencereleri vasıtasıyla elde edilir.

- Projede farenin sağ ve sol tuşları kullanılmıştır. Kalıp seçme, nokta belirtme, kutulama özelliği ile nokta grubu belirtme gibi işlemlerde daha ziyade sol tuş tercih edilirken kalıp noktası eklemeyi bitirme gibi işlemlerde de sağ tuş tercih edilmiştir.

- Kalıp üstünde bir işlem gerçekleştirilecek ise önce fare yardımı ile kalıp veya kalıplar seçilir ardından işlem yapılır.

- Kalıbın seçildiği, o kalıba ilişkin noktaların görünür hale gelmesi veya kalıbın renksiz olması ile belirtilir.

- İşlemler esnasında seçilen noktalar içi boş şekiller olarak belirtilir.

- Projeyi oluşturan modullerin ana çalışma penceresinde o modüle uygun araç çubuğu tanımlıdır. Bu araç çubuğunda bazı Çalışma Düzeyleri

mevcuttur. Bir çalışma düzeyi seçildiğinde o modda çalışma yapılır. Diğer bir çalışma düzeyi belirtilene kadar aynı düzeyde kalınır.

- Araç çubuğunda seçili çalışma düzeyini standart çalışma düzeyine getirmek için fare sol tuşuna çift tıklanılması veya ESC tuşuna basılması yeterlidir.

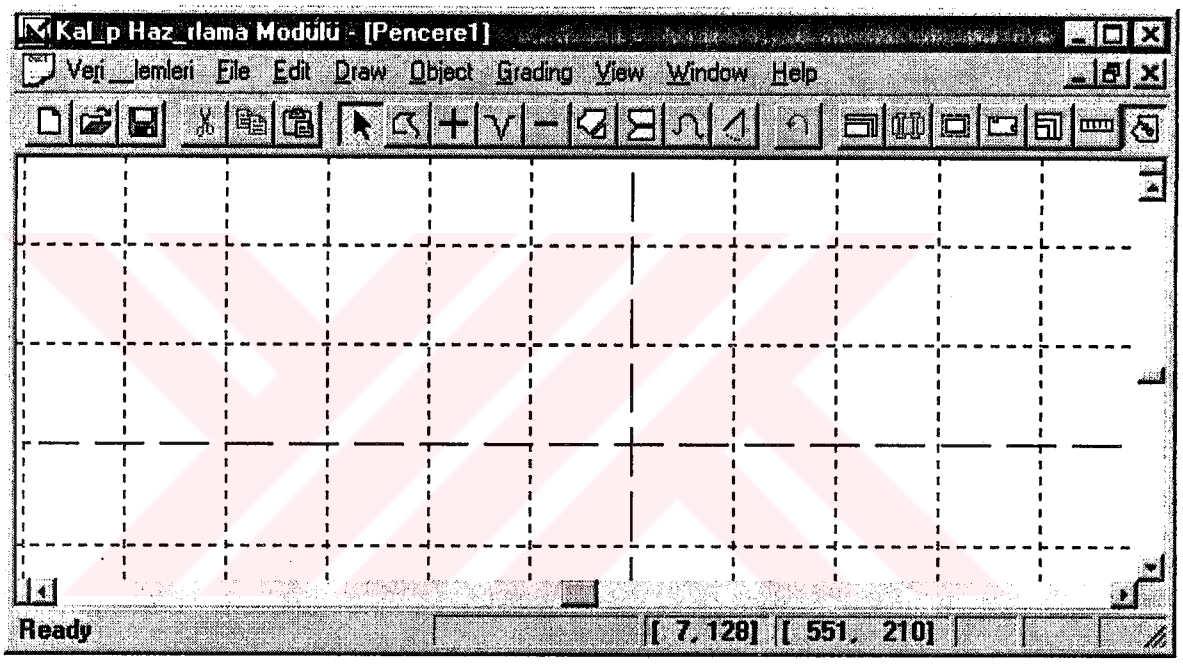
Modülerliği sağlamak ve konuyu daha iyi kavrayabilmek için proje bazı ana başlıklara ayrılmıştır. Söz konusu proje ana başlıkları şunlardır :

1. Model tasarım işlemleri
2. Pastal siparişi işlemleri
3. Pastal planı hazırlama işlemleri

Projeyi oluşturan bu modüllerin anlatımı ilerleyen kısımlarda başlıklar halinde yer almaktadır.

## 6.1. Model Tasarım İşlemleri :

Bu proje modülünde modeli oluşturan kalıpların tasarımına ilişkin çalışmalar yapılmaktadır. Tasarım alanı tasarıma kolaylık sağlaması açısından Windows işletim sisteminde mevcut ilerleme (scroll) özelliğine, bölmeleme (splitter) yapısına ve diğer özelliklere sahiptir. Model tasarım işlemleri menüde, araç çubuğunda (toolbar) ve klavye makrolarında tanımlı işlemlerle gerçeklenir. Model tasarım modülünün genel görünüş yapısı şekil 6.1'de görülmektedir.




Şekil 6.1. Model Tasarım Modülünün genel görünüşü.

### Araç çubuğu yapısı :

Çeşitli işlemlerin ve çalışma düzeylerinin tanımlandığı model tasarım modülünde tanımlı araç çubuğunun yapısı şekil 6.2'de görülmektedir.

Şekil 6.2. Model Tasarım Modülünde mevcut araç çubuğunun yapısı.

Model tasarım modülü bazı alt bölümlere ayrılmıştır ;

1. Görüntü düzenleme işlemleri
2. Kalıp oluşturma ve kalıbı biçimlendirme işlemleri
3. Kalıp manüplasyon işlemleri
4. Pastala yönelik işlemler
5. Veri tabanı işlemleri

Bu alt bölümlerin ayrıntılı incelemesi aşağıda yer almaktadır.

### 6.1.1. Görüntü Düzenleme İşlemleri :

Bu kısımda tasarım alanını oluşturan görüntü yapısının düzenlenmesine ilişkin işlemler tanımlanmaktadır. Bu işlemler menüde tanımlı bazı komutlarla gerçeklenir. Ayrıca klavye tuşlarında tanımlı bazı komutlarda vardır. Bu işlemleri aşağıdaki gibi gruplar halinde tanıtabiliriz.

#### 6.1.1.1. Zoom İşlemleri :

Zoom işlemi kalıplara yaklaşma veya kalıplardan uzaklaşma işlemidir. Bu işlemler esnasında kalıp ölçüleri değişmemektedir. Kullanıcının bazı detayları daha iyi değerlendirmesine olanak sağlayan bir işlemdir. Zoom işleminin çeşitleri şunlardır;

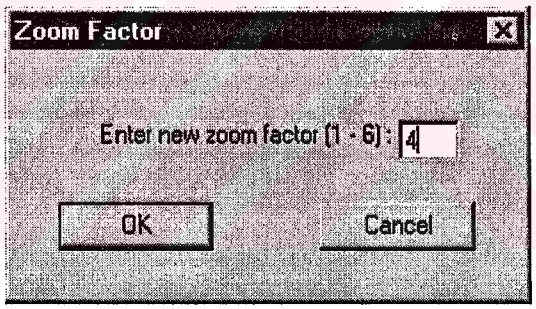
- **Zoom Yaklaşma :** Görüntü menüsü altında tanımlı Zoom Artı menüsü her seçildiğinde kalıplar iki katı büyür. Yani kalıplar iki katı büyük görünecek şekilde kalıplara yaklaşılır. Ayrıca klavyenin sağında yer alan + tuşunu her basıldığında işlem tekrarlanabilir.

- **Zoom Uzaklaşma :** Görüntü menüsü altında tanımlı Zoom Eksi menüsü her seçildiğinde kalıplar yarı yarıya küçülür. Yani kalıplar yarı yarıya küçük görünecek şekilde kalıplardan uzaklaşılır. Ayrıca klavyenin sağında yer alan - tuşunu her basıldığında işlem tekrarlanabilir.

- **Zoom Seçili Kalıplar :** Görüntü menüsü altında tanımlı Zoom menüsündeki Seçili Kalıp menüsü seçildiğinde sadece seçili kalıplar ekranda yer alacak şekilde görüntü düzenlenir.

- **Zoom Tüm Kalıplar :** Görüntü menüsü altında tanımlı Zoom menüsündeki Tüm Kalıplar menüsü seçildiğinde tüm kalıplar ekranda yer alacak şekilde görüntü düzenlenir.

- **Zoom Ayarı :** Görüntü menüsü altında tanımlı Zoom menüsündeki Zoom Ayarı menüsü seçildiğinde şekil 6.3'de görülen iletişim penceresi ekrana gelir. Pencerede yer alan zoom değeri fare hareketi ile ulaşılabilen en küçük çalışma biriminin değeridir. Bu değer milimetre cinsindendir. Zoom değerinin tabii değeri 4 milimetredir.




Şekil 6.3. Zoom Ayarı Penceresinin Yapısı.

### 6.1.1.2. Izgara Yapısı ve İşlemleri :

Tasarım alanını kaplayan yatay ve dikey çizgi yapıları ölçülendirmeyi kolaylaştıran yardımcı bir ızgara yapısı oluşturur. Uzun kesikli çizgiler eksenlerin merkezini göstermekte olup kısa siyah kesikli çizgiler 1 metre aralıkla yerleştirilmiştir. Kırmızı kısa kesikli çizgiler ise 10 metre aralıkla yerleştirilmiştir. Zoom değeri belli değerleri aştığında 1 metre aralıklarla yer alan kısa siyah kesikli çizgiler yok olur. Bunun

sebebi tasarım alanında oluşacak karışık görüntüye mani olmaktır. İstenirse bu ızgara yapısı Görüntü menüsünde tanımlı Izgara menüsü yardımı ile yok edilebilir veya tekrar görünür hale getirilebilir.

### 6.1.1.3. Bölmeleme (Splitter) Yapısı :

Bölmeleme (Splitter) özelliği sayesinde aynı tasarım alanının farklı alanları üzerinde çalışma yapılabilir. Yani ekrana sığmayan farklı kalıplar üzerinde çalışma yapılabilir. Bu özellikten yararlanmak için tasarım alanının solunda ve üstünde yer alan çubukların fare yardımı ile taşınması ve istenilen noktalara doğru ilerleme (scroll) yapılması gereklidir. Tasarım alanını eski haline getirmek için çubukların ilk konumlarına taşınması gerekir.

### 6.1.1.4. İlerleme (Scroll) Özelliği :

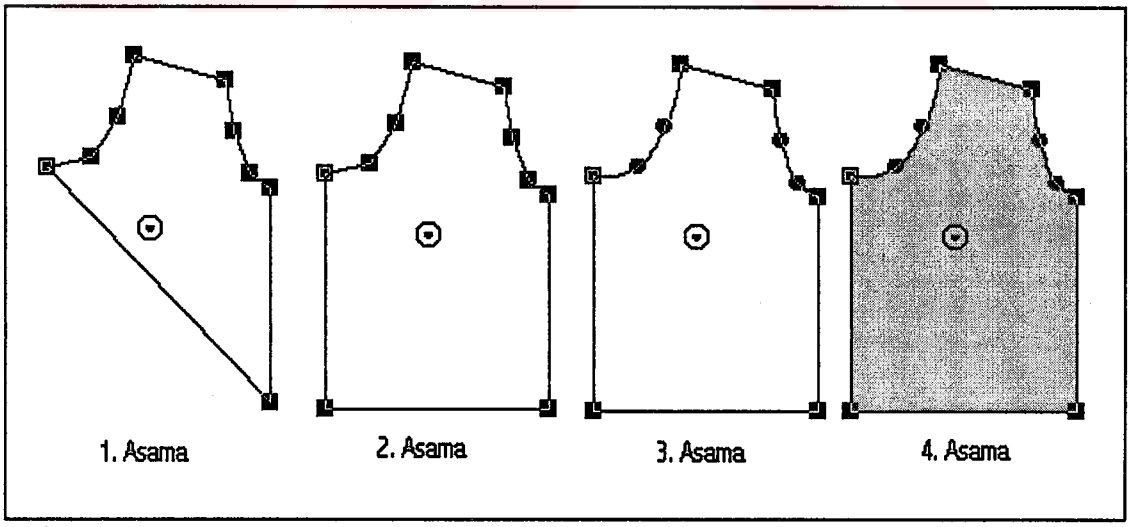
Tasarım alanının ekrana sığmayan kısımlarını görebilmek için kullanılan Windows işletim sistemi desteğidir. Bu özellikten faydalanmak için alanın sağında ve altında yer alan ilerleme (scroll) kutularının yerlerinin istenilen alanın görecek şekilde değiştirilmesi yeterlidir. Böylece geniş bir çalışma sahası elde edilmiş olur. Eski görüntü parçasına ulaşmak için ilerleme kutularını eski yerine getirmek gerekir.

### 6.1.2. Kalıp Oluşturma ve Kalıp Biçimlendirme İşlemleri :

Menüde ve araç çubuğunda, kalıp oluşturma ve kalıbı biçimlendirme amacıyla kullanılmak üzere bazı Yetenekler tanımlıdır. Buna ek olarak araç çubuğunda bazı Çalışma Kademeleri de tanımlıdır. Bu işlevleri tek tek şu şekilde tanıtabiliriz:

#### 6.1.2.1. Kalıp Oluşturma :

Eğri ve düz çizgilerden oluşan kalıpların tasarım işleminde kullanıcı ilgili toolu menü veya araç çubuğundan seçtikten sonra fare ile kalıbı tanımlamaktadır. Farenin sol tuşunda yapılan her tık için bir nokta kalıba eklenmektedir. Sağ tuşa basıldığında kalıp oluşturma işlemi sonlanmakta ve bir kalıp elde edilmektedir. Bu şekilde oluşturulan kalıp sadece düz çizgilere sahiptir. Daha sonraki işlemlerde gerekli olan referans noktası ve kalıp ilk noktası bu aşamada otomatik olarak tanımlanmaktadır. Bu noktalar normalde sistem gereği ilk nokta üzerinde tanımlanmaktadır. Ayrıca kalıp üzerindeki diğer işlemlerin sağlıklı gerçekleştirilebilmesi için kalıp noktaları saat işareti yönünde girilmelidir. Bir kalıp oluşturma örneği şekil 6.4'de görülmektedir.




Şekil 6.4. Kalıp oluşturma örneği.

### 6.1.2.2. Kalıp ve Noktayı Taşıma :

Araç çubuğunda tanımlı temel çalışma modudur. Bu modda yaklaşım önce seç sonra istenirse taşıdır. Eğer farenin sol tuşunun basıldığı yer kalıp üstünde ise kalıp seçilir. Seçilme işlemini kullanıcıya haber vermek için de kalıbın noktaları belirmektedir. Seçili kalıp yada noktaları fare sol tuşunun yardımı ile istenilen yere taşınabilir. Bunun için sol tuşa basıp sürüklemek ve istenilen yere gelindiğinde sol tuşu bırakmak gerekir.

Boş bir yerde fare sol tuşuna basıp sürüklemek sureti ile kalıplar grup halinde seçilebilir. Bu özelliğe kısaca kutu adı verilmektedir. Bu sayede seçilen kalıp veya kalıp grupları üzerinde aşağıda anlatacağımız bazı işlemler gerçekleştirilebilir.

### 6.1.2.3. Eğri - Düz Çizgi Dönüşümü :

Kalıplar ilk tanımlandığında sadece düz çizgilerden oluşmaktadır. Fakat gerçeğe uygun tasarımları yapabilmek için eğri karakteristikteki çizgilere de ihtiyaç duyulur. Araç çubuğunda bu tool seçildiğinde, seçili kalıbın istenilen noktaları eğri noktaya dönüştürülebilmektedir. Kalıbın herhangi bir noktası üstünde fare sol tuşuna basıldığında veya kutu ile grup halinde noktalar belirtildiğinde ilgili noktalar eğri noktaya dönüşmektedir. Dönüşüm ise düz noktanın kare, eğri noktanın yuvarlak olarak gösterilmesi ile belirtilir. Eğri noktalarda, gelen ve giden kenarlar eğridir. Eğrinin çizimi için literatürdeki bazı yöntemler denenmiş ve projeye uygun olan biri seçilmiştir. Eğri - Düz Çizgi Dönüşümüne bir örnek şekil 6.4'de görülmektedir. Şekil 6.4'de yer alan kalıp üzerinde 2. ve 3. asamalar arasında kol ve yaka kısımları için eğri – düz çizgi dönüşümü yapılmıştır.

### 6.1.2.4. Nokta Ekleme :

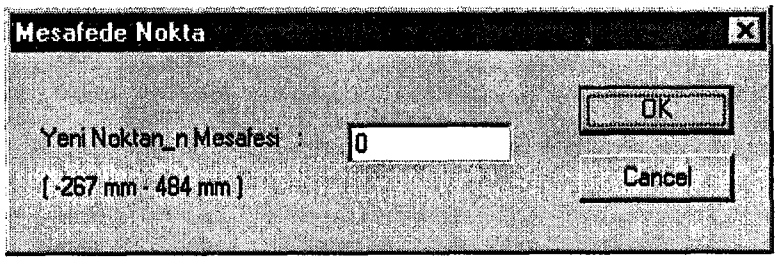
Kalıp oluşturulduktan sonra bazen nokta eklemek gerekebilir. Bunun için nokta eklenilmek istenen kalıp seçilir ve araç çubuğunda tanımlı nokta ekleme yeteneğine basılır. Bu taktirde üç çeşit nokta ekleme yeteneğini barındıran bir menü belirir. Projede tanımlı nokta ekleme yetenekleri şunlardır ;

- Serbest nokta ekleme
- Mesafeli nokta ekleme
- Hareketli nokta ekleme

Araç çubuğunda bu yeteneğe ilişkin buton işlem süresince basılı kalır.

- **Serbest Nokta Ekleme :** Bu yetenek belirtildiğinde seçili kalıbın istenilen noktasından itibaren saat işareti yönünde noktalar eklenebilir. Öncelikle devamında ekleme yapılacak nokta belirtilir. Ardından da fare sol tuşunun her basılışı için kalıba bir nokta eklenir. Fare sağ tuşuna basıldığında işlem iptal olur.

- **Mesafeli Nokta Ekleme :** Eklenecek nokta, kalıpta belirtilen nokta ile bir sonraki noktayı birleştiren çizgi üzerinde yer alır. İşlemin esası çizginin eğimini bozmadan yeni bir noktanın araya eklenmesidir. Devamına eklemenin yapılacağı kalıp noktası belirtildiğinde mesafenin ne kadar olacağını soran bir iletişim kutusu (dialog box) ekrana çıkar. Mesafe girilip enter'a basıldığında ekleme yapılır. Bu işlem istenildiği kadar tekrarlanabilir. Söz konusu iletişim kutusu şekil 6.5'de görülmektedir.

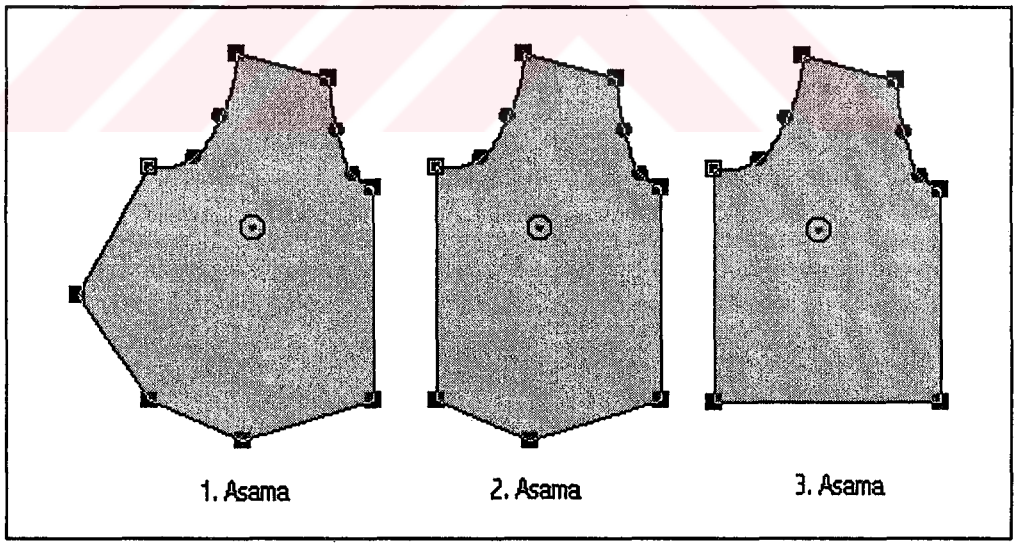



Şekil 6.5. Nokta mesafesinin girildiği iletişim kutusu.

- **Hareketli Nokta Ekleme :** Mesafeli nokta eklemeye benzer fakat bu yetenekte noktalar farenin hareketi ile eklenir. Üstüne eklemenin yapılacağı çizgi belirtildikten sonra her fare basımı için çizgiye bir nokta eklenir. Eklenen noktaların koordinatı fare göstergesinin çizgi üzerine iz düşümüdür. Bu işlemler yapılırken de çizginin eğimi bozulmamaya çalışılır.

### 6.1.2.5. Nokta Silme :

Seçili kalıbın noktaları silinmek istendiğinde bu yetenek kullanılır. İstenen nokta üstünde fare sol tuşuna basıldığında silme işlemi gerçekleşir. Ayrıca kutulama özelliği ile nokta grubu belirtildiğinde grup halinde de silme işlemi gerçekleştirilebilir. Bir başka yetenek belirtilene kadar nokta silme tekrarlanabilir. Şekil 6.6'de nokta silme ile ilgili bir örnek görülmektedir.




Şekil 6.6. Nokta silme çalışması.

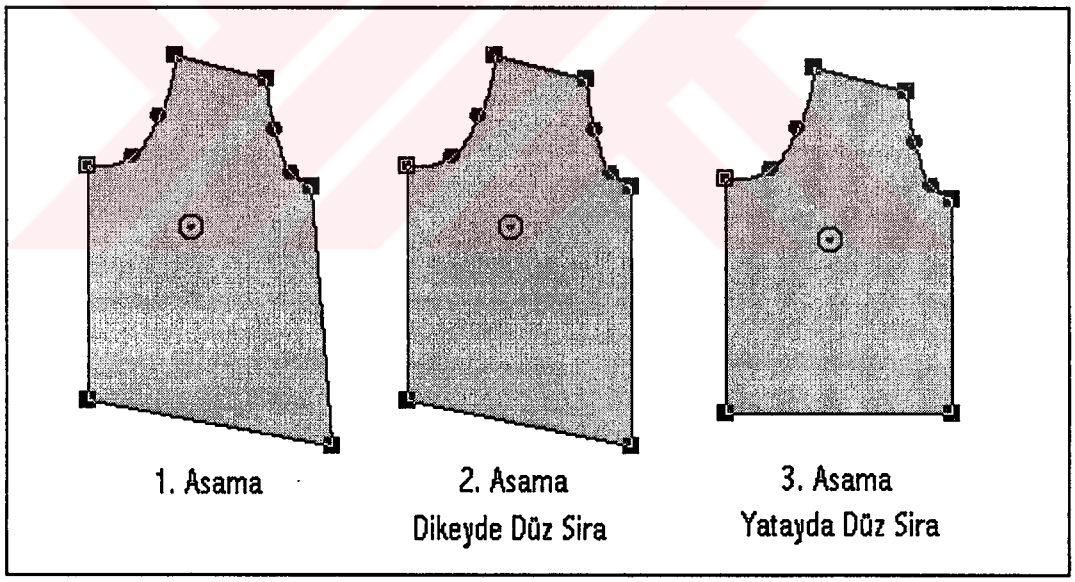
### 6.1.2.6. Düz Sıra :

Aynı hizaya getirmek istediğimiz noktalar varsa düz sıra yeteneğini kullanırız. Bu yetenekte iki noktanın sıra ile belirtilmesi gerekir. Belirtilen noktalar içi boş kare şeklinde görünür. Kullanıcı ilk noktayı seçtikten sonra ikinci noktayı belirttiğinde bir menü belirir. Bu menüde şunlar vardır ;

- **Yatayda Düz Sıra :** Seçilen iki nokta x ekseni doğrultusunda aynı hizaya getirilir.

- **Dikeyde Düz Sıra :** Seçilen iki nokta y ekseni doğrultusunda aynı hizaya getirilir.

Düz sıra çalışmasının neticesinde oluşan kalıp yapısı şekil 6.7'da görülmektedir.

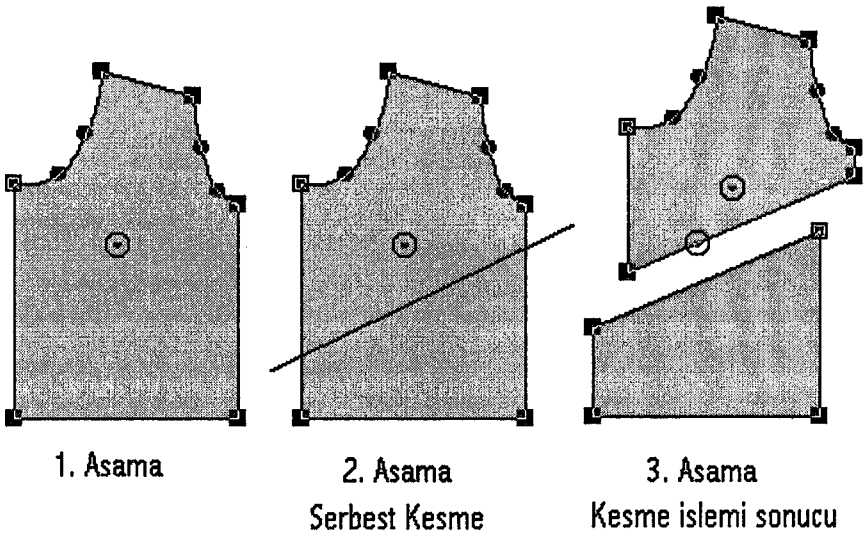



Şekil 6.7. Düz sıra çalışmasının sonucu.

### 6.1.2.7. Kalıp Kesme :

Temel kaide kalıbı istenilen iki parçaya ayırmaktır. Ayıran çizgi yani kesme çizgisi fare sol tuşu yardımıyla belirlenir. Tuşa ilk basıldığında kesme çizgisinin ilk noktası belirir. Fareyi hareket ettirdikçe kesme çizgisinin ikinci noktası da hareket eder. Tuşa bir defa daha basıldığında kesme çizgisi sonlamış olur.. Bu şekilde belirlenen kesme çizgisinin üzerinden geçtiği kalıp iki parçaya ayrılır. Kesme işlemi sadece seçili kalıplar üzerinde gerçekleştirilir. Kalıbı ikiden daha fazla parçaya ayıran kesme işlemi yerine getirilmez. İşlem sonucunda orjinal kalıpta sistemde tanımlı kalır. Bu şekilde gerçekleştirilen çeşitli kesme teknikleri tanımlanmıştır ;

- **Serbest Kesme :** Kalıp, farenin serbest hareketi ile belirlenen çizgi sayesinde iki parçaya ayrılır. Ayrıca kalıbı iki noktasından kesme olanağı da mevcuttur. Fare kalıbın noktası üzerine geldiğinde şeklini değitirir. Bu esnada kesme çizgisi başlatılır ve bir diğer noktada sonlandırılır ise kalıp iki noktasından kesilir.
- **Yatay Kesme :** Kalıbı yatay bir çizgi ile iki parçaya ayırmak esastır. Yatay kesme çizgisinin y değeri, ilk belirlenen noktanın y değerineeşittir. İkinci belirlenen nokta kesme çizgisinin sadece x değerini tanımlar.
- **Dikey Kesme :** Kalıbı dikey bir çizgi ile iki parçaya ayırmak esastır. Dikey kesme çizgisinin x değeri, ilk belirlenen noktanın x değerineeşittir. İkinci belirlenen nokta kesme çizgisinin sadece y değerini tanımlar.




Şekil 6.8 Serbest kesme işlemine bir örnek.

### 6.1.2.8. Çıt :

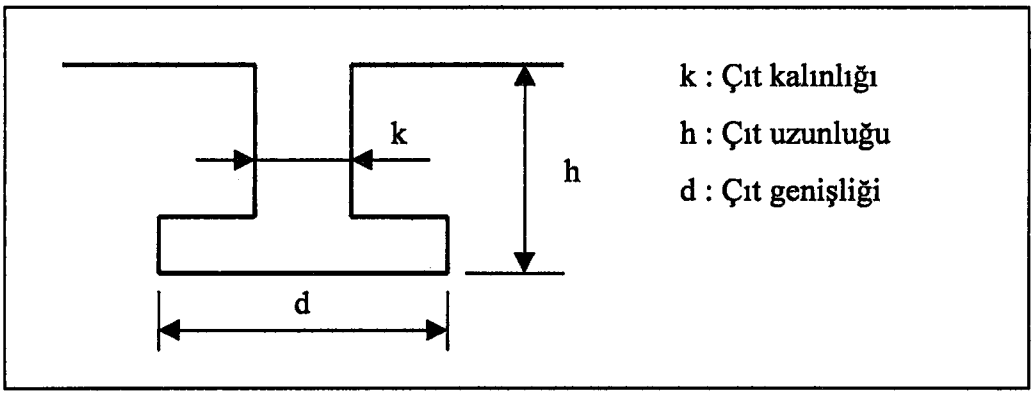
**Tanımı :** Daha önce bahsettiğimiz gibi çıt, tasarımı gerçekleştirilen kalıp kenarlarının bir parçası olan ve kenarların karakteristiğini ( eğri veya düz karakteristiği ) bozmayan özel bir noktadır. Çıtın esas işlevi kesim esnasında kullanılmak üzere kalıp kenarlarına bazı işaretler bırakmaktır. Kalıpların kesimini gerçekleştiren kişiler kesim sırasında çıtın tipine göre o noktalara bazı küçük kesikler atar ve bu kesikler dikimde dikkate alınır.

Çeşitli tipleri mevcuttur. Ayrıca müşteri firmanın isteğine uygun olarak özel tipleri de tanımlanabilir. Bizim projede tanımlı tipleri üçgen, kare ve T çıttır.

**Çözümü :** Kalıp üzerinde çıt noktası oluşturabilmek için önemli kriterler şunlardır ;

- Çıtın Tipi ( Üçgen / Kare / T çıt )
- Çıtın Uzunluğu (.....mm. )
- Çıtın Genişliği (.....mm. )
- Çıtın Kalınlığı (.....mm. )

Yukarıda görüldüğü gibi tüm çıt değerleri mm. ( mili metre ) cinsindedir. Bu kriterleri şekil 6.9'de T çıt üzerinde şöyle tanımlayabiliriz :




Şekil 6.9. Genel olarak çıt yapısı.

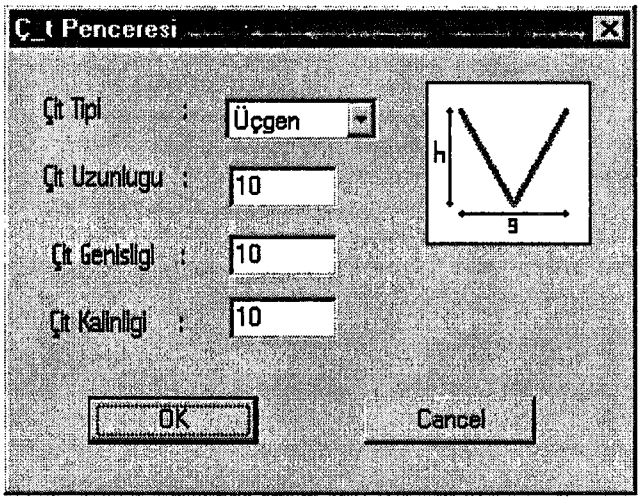
Kullanıcı öncelikle, çıt yerleştirmek istediği yere bu bölümde bahsettiğimiz Hareketli Nokta Ekleme özelliği ile bir nokta ekler. Ardından da araç çubuğundaki Çıt İşlem butonuna basarak Çıt Çalışma Düzeyine geçer. Bu düzeyde eklenen noktayı fare yardımı ile çıt olarak işaretler ve ekrana gelen iletişim kutusunda ilgili değerleri

belirler. Bu iletişim kutusunda yukarıda belirtilmiş olan kriterler yer alır. Çıtın görünüşü bu kriterlerin alacağı değerlere göre değişir.

Ekranda karmaşıklığı önlemek amacıyla çıt konusunda iki çalışma modu tespit edilmiştir; normal çalışma modu ( çıt göstermeme modu ) , çıt göster modu. Çalışma modu araç çubuğunda Çıt Göster/Gösterme butonuna basmak sureti ile belirtilir. Çıt, normal çalışma modunda sadece yassı bir nokta olarak görüntülenirken, çıt göster modunda ise ilgili iletişim kutusunda belirtilen değerlere uygun bir şekilde görüntülenir.

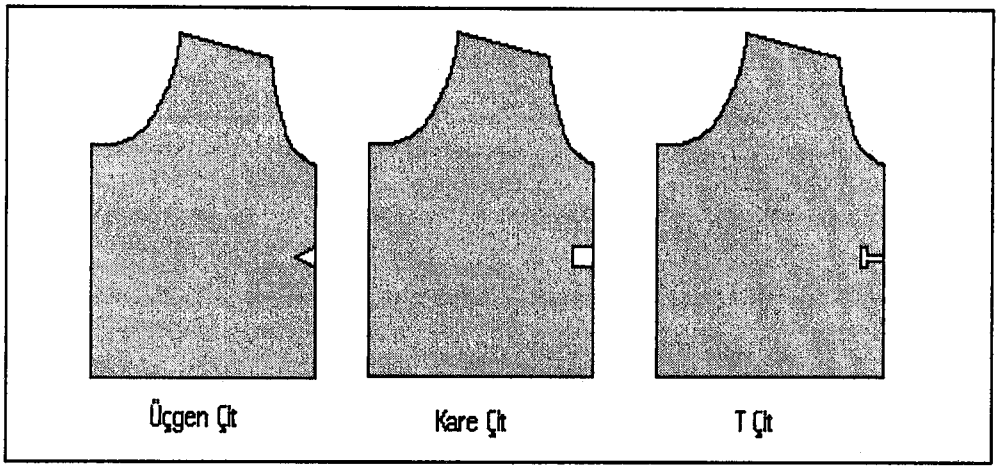
Çıt noktası, diğer noktalar da olduğu gibi farenin sürükle-bırak hareketi ile serbest olarak taşınabilir. Çıt Göster modunda ise çıt, profil noktası yardımı ile taşınır ve bu modda profil noktasının hareketi ile aynı anda çıtın görüntüsü de güncellenir.

Çıtın tipi veya onunla ilgili herhangi bir değer değiştirilecek ise araç çubuğunda Çıt İşlem butonuna basılır ve ekrana gelen menüden Çıt Bilgisi kısmı seçilir. Ardından da istenilen çıt noktası fare desteği ile belirtilir ve ekrana gelen iletişim kutusu yardımıyla değişiklik gerçekleştirilir. Bu işlem seri bir şekilde tekrar tekrar uygulanabilir. Burada bahsedilen iletişim kutusunun yapısı şekil 6.10'da yer almaktadır ;




Şekil 6.10 Çıt iletişim kutusu

Projede yer alan üçgen, kare ve T çıta ilişkin örnek çizimler şekil 6.11'de görülmektedir. Bu çizimler çıt gösterme modunda elde edilmiştir.




Şekil 6.11 Çıt tipleri

### 6.1.2.9. Pervaz :

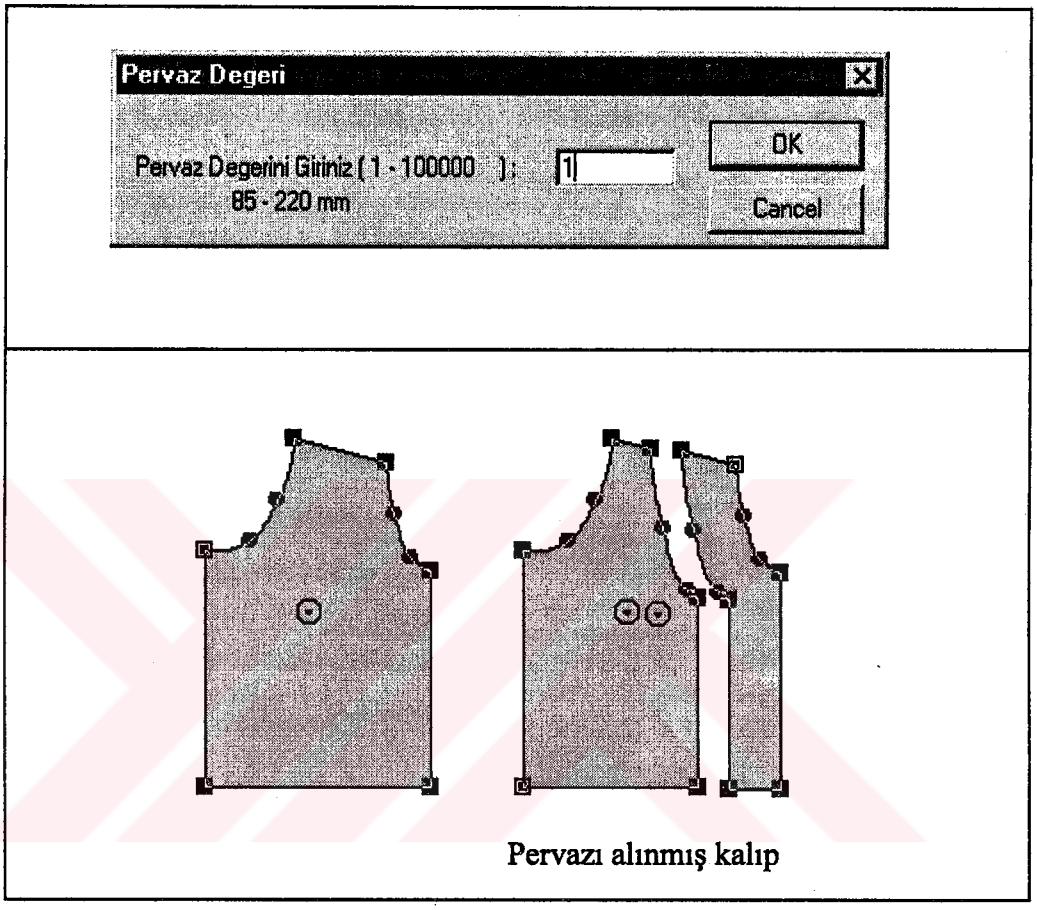
**Tanımı :** Pervaz, kalıp kenarlarına paralel olarak belli bir genişlikte bazı parçaların kalıptan çıkarılmasını temel alan bir yaklaşımdır.

**Çözümü :** Kalıptan ayrılacak bir pervaz parçası için önemli kriterler şunlardır ;

- Pervazın başlangıç noktası
- Pervazın son noktası
- Pervazın genişliği ( mm. - mili metre)

Kullanıcı pervazı oluşturabilmek için araç çubuğunda tanımlı Pervaz İşlem butonuna basarak Pervaz Çalışma Düzeyine geçer. Ardından da yukarıda belirtilmiş olan kriterleri sırası ile yerine getirir. İlk ve son noktayı belirttikten sonra ekrana gelen iletişim kutusu ile pervazın genişliğini tespit eder. İletişim kutusunda Kabul tuşuna basıldığında pervaz parçası, kalan kısım ve orjinal kalıp şeklinde üç parça oluşur. Pervazın belirtilmiş olan ilk ve son noktaları düz nokta karakteristiğinde olmalıdır.

Pervaz genişliğinin belirtildiği ileitişim kutusunun yapısı ve oluşturulan bir pervazın görüntüsü şekil 6.12'de yer almaktadır ;




Şekil 6.12. Pervaz iletişim kutusu ve pervaz örneği.

### 6.1.2.10. Pile :

**Tanımı :** Kalıp üzerinde birbiri üzerine katlanma kenarları, büzgüler oluşturan bir yapıdır. Tasarımı bitmiş yani tasarımı temsil eden kenarları artık değişmeyecek olan bir kalıba uygulanır. Pilenin bir doğrultusu ve bu doğrultuyu kesen iki kalıp kenarı vardır. Bunun için pile tanımı yapılırken öncelikle pilenin temel bileşeni olan pile doğrultusu, iki kalıp noktası yardımı ile tespit edilir. Tasarım esnasında pile noktaları çıt ile işaretlenir ve dikim esnasında da bu çıt noktaları dikkate alınarak pile doğrultusu ile dik olan kenarların biri veya her ikisi de dikilir. Sağa veya sola kat oluşturan pileler ve kanun diye tabir edebiliceğimiz özel bir pile tipi de mevcut pile tipleri arasında örnek olarak gösterilebilir. Birden fazla sayıda pile birbirinden belli bir uzaklıkta ve belli bir pile genişliğinde yer alabilir.

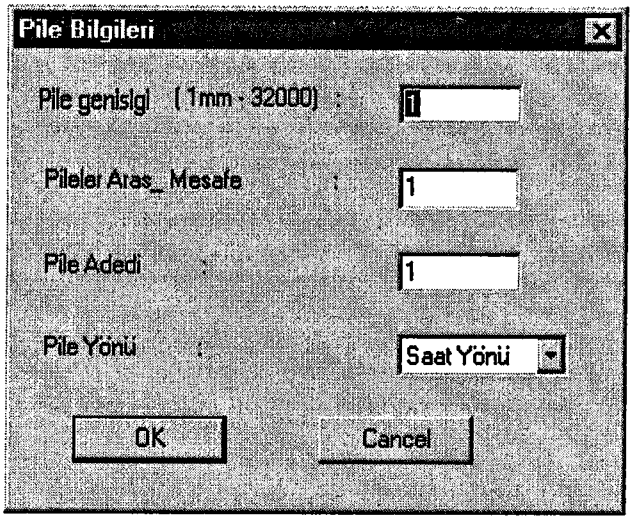
**Çözümü :** Kalıp üzerinde pile veya pileler oluşturabilmek için önemli kriterler şunlardır ;

- Pile Doğrultusu
- Pile Genişliği ( mm - milimetre )
- Pile Adedi
- Pileler Arası Mesafe ( mm - milimetre )
- Pilenin Katlama Yönü( Sağa veya Sola )

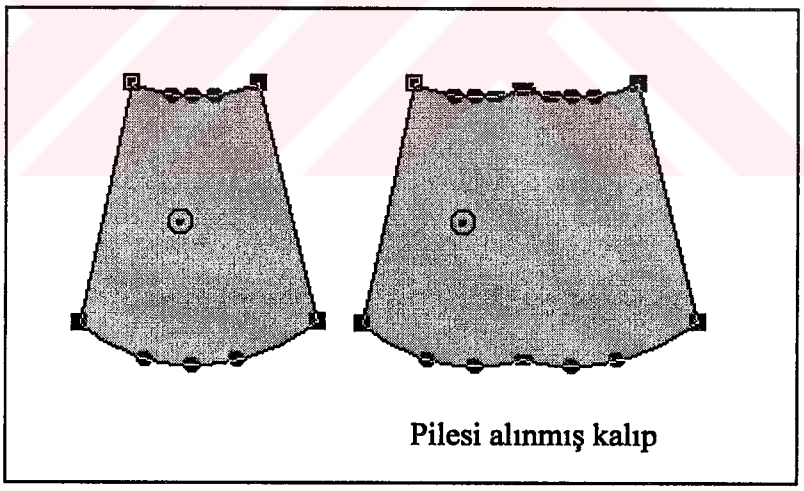
Pileler arası mesafe değeri, sadece pile adedi birden fazla ise kullanılır.

Kullanıcı pile oluşturabilmek için araç çubuğunda tanımlı Pile İşlem butonuna basarak Pile Çalışma Düzeyine geçer. Ardından da yukarıda belirtilmiş olan kriterleri sırası ile yerine getirir. Pile doğrulutsunu tespit eden iki kalıp noktası fare yardımı ile belirttildikten sonra ekrana gelen iletişim kutusu vasıtasıyla diğer dört pile kriteri belirlenir. İletişim kutusunda Kabul tuşuna basıldığında isteğe uygun pileler oluşturulur. Eğer gerçekleştirilmesi mümkün olmayan pile değerleri girilmiş ise belirtilen pile genişliği ve pileler arası mesafe değerine uygun olarak sağlanabilecek maksimum pile adedinin ne olduğu kullanıcıya bir mesaj kutusu desteği ile bildirilir.

Pile kriterlerinden son dörtünün belirlendiği iletişim kutusunun yapısı şekil 6.13'de ve oluşturulmuş örnek bir pilenin çizimi şekil 6.14'de görülmektedir ;




Şekil 6.13. Pile kriterlerinin tespit edildiği iletişim kutusunun yapısı




Şekil 6.14. Pile örneği.

### 6.1.3. Kalıp Manüplasyon İşlemleri :

Menüde ve araç çubuğunda, kalıp üzerinde çeşitli manüplasyon işlemlerini gerçeklemek üzere bazı Yetenekler tanımlıdır. Bunlara ek olarak çeşitli popup menüler de tanımlıdır. Bu işlevleri tek tek şu şekilde tanıtabiliriz:

#### 6.1.3.1. Kalıp Döndürme :

Kalıplar oluşturulduktan sonra bazen onları döndürmek gerekebilir. Döndürme işlemi hem tek bir kalıba hem de seçilmiş bir grup kalıba uygulanabilir. İşlem için klavyede tanımlı tuşlar kullanılabileceği gibi Object menüsündeki Rotate menüsü altında tanımlı menüler de kullanılabilir. Döndürme işlemi uygulanırken kalıp veya kalıplarla ilgili tüm bileşenler ( kalıp düz ipliği, referans noktası vs. ) bu işlemden etkilenir. Projede tanımlı kalıp döndürme yetenekleri şunlardır ;

- Klavye tuşları ile kalıp döndürme işlemi
- Açılı kalıp döndürme işlemi
- Düz ipliğe göre kalıp döndürme işlemi

- **Klavye Tuşları ile Kalıp Döndürme İşlemi :** Döndürme işlemi uygulanacak kalıp veya kalıplar seçildikten sonra Ctrl + Sağ Ok tuşuna her basıldığında 5 derece saat işareti yönünde, Ctrl + Sol Ok tuşuna her basıldığında 5 derece saat işaretinin tersi yönde döndürme işlemi gerçekleştirilir.

- **Açılı Kalıp Döndürme İşlemi :** Object - Rotate menüsü altında tanımlı Angle menüsü seçilmek sureti ile istenilen dönüş açısı değeri belirtilebilir. Menü seçildiğinde ekrana dönüş açısının "derece" olarak belirtileceği bir iletişim kutusu gelir. Dönüş açısı değeri -360, 360 arasında değişebilir.

- **Düz İpliğe göre Kalıp Döndürme İşlemi :** Bu tip döndürme işleminde temel unsur döndürme işleminin kalıp düz ipliğine göre gerçeklenmesidir.

Düz ipliğe göre iki çeşit kalıp döndürme işlemi gerçekleştirilebilir. Bu iki tip kalıp döndürme işlemleri Object - Rotate menüsü altında tanımlıdır ve isimleri aşağıda yer almaktadır ;

- Düz İpliği Yataya Döndürme
- Düz İpliği Dikeye Döndürme

### 6.1.3.2. Simetri İşlemleri :

Tasarımı yapılmış kalıpların çeşitli şekillerde simetrisini almak gerekebilir. Bu işlemi gerçeklemek için ya bu amaçla tanımlı bir popup menü ya da Object menüsünde Symmetry menüsü altında tanımlı menüler kullanılabilir. Söz konusu popup menü simetrisi alınmak istenilen kalıp üstünde sağ fare tuşuna basılarak elde edilebilir. Ama öncelikle kalıbın sol fare tuşu ile seçilmesi gerekir. Tanımlı simetri çeşitleri şunlardır ;

- X'de Simetri
- X'de Simetri +
- Y'de Simetri
- Y'de Simetri +
- Eksene göre Simetri
- Eksene göre Simetri +

Bu yetenekler menüde tanımlı isimleri ile şu işlevleri yerine getirirler ;

- **X'de Simetri :** Bu yöntem seçilir ise kalıbın kendi merkezinde X eksenine göre simetrisi alınır.

- **X' Simetri + :** Kalıbın X eksenine göre simetrik kopyası elde edilir. Orjinal kalıp da tanımlı kalır.

- **Y'de Simetri :** Kalıbın kendi merkezinde Y eksenine göre simetrisi elde edilir.

- **Y'de Simetri + :** Kalıbın Y eksenine göre simetrik kopyası elde edilir. Aynı zamanda orjinal kalıp da tanımlı kalır.

- **Eksene göre Simetri :** Fare yardımı ile seçilen iki kalıp noktasının oluşturduğu bir eksene göre kalıbın simetrisi elde edilir.

- **Eksene göre Simetri + :** Fare yardımı ile seçilen iki kalıp noktasının oluşturduğu bir eksene göre kalıbın simetrik kopyası elde edilir. Aynı zamanda orjinal kalıp da tanımlı kalır.

### 6.1.3.3. Kalıp Düzenleme (Edit) İşlemleri :

Tasarım alanında tanımlı bir veya bir kaç kalıbı silmek, kesmek, kopyalamak ve yapıştırmak gerekebilir. Tüm bu işlemler Düzenleme menüsü altında tanımlıdır. Bu işlemler esnasında Windows ortamının bize sağlamış olduğu clipboard adı verilen bir yapıdan faydalanılır. Öncelikle üzerinde işlem yapılacak kalıp veya kalıplar fare yardımı ile seçilir. Daha sonrada seçilen kalıp(lar) üzerinde aşağıda tanımı yapılan Düzenleme işlemleri gerçeklenebilir ;

- **Silme (Delete) :** Düzenleme menüsü altında tanımlı Silme menüsü yardımı ile gerçeklenir. Kalıp veya kalıplar hem görüntüde hem de veri yapısından silinir. Bu yöntemde silinen kalıp(lar) yeniden elde edilemez.

- **Kesme (Cut) :** Düzenleme menüsü altında tanımlı Kesme menüsü yardımı ile gerçeklenir. Kalıp veya kalıplar sadece görüntüde silinir. Silinen kalıp(lar) aşağıda tanımı yapılan Yapıştırma yöntemi ile tekrar elde edilebilir.

- **Kopyalama (Copy) :** Düzenleme menüsü altında tanımlı Kopyalama menüsü yardımı ile gerçeklenir. Kalıp veya kalıpların kopyası çıkarılabilir. Bu menü seçildiğinde görüntüde bir değişiklik olmaz.

Sadece kalıp(lar)ın kopyası clipboard'a atılır. Daha sonradan Yapıştırma yöntemi ile kalıp(lar)ın kopyası elde edilebilir.

- **Yapıştırma (Paste) :** Düzenleme menüsü altında tanımlı Yapıştırma menüsü yardımı ile gerçeklenir. Yukarıda tanımlanan Kesme veya Kopyalama yöntemi ile clipboard'a atılmış kalıp(lar)ın kopyası veya kopyaları elde edilebilir. Burada temel unsur sadece en son üzerinde Kesme veya Kopyalama işlemi yapılan kalıbın veya grup seçilmiş kalıpların Yapıştırma ile tekrar elde edilebilmesidir.

### 6.1.3.4. Kalıp Kopyalama :

Yukarıda tanımlanan kalıp kopyalama yöntemine ek olarak fare göstergesi seçili kalıp veya kalıplar üstünde iken Ctrl tuşuna basılır ve fare sol tuşu tıklanırsa kalıp veya kalıpların kopyası elde edilebilir. Bu çalışma Kalıp Düzenleme (konu 6.1.3.3) konusunda tanımlı Kopyalama ve Yapıştırma işleminin ard arda gerçeklemesi ile elde edilen çalışmanın bir benzeridir. Fakat burada clipboard ara birimi kullanılmaz. Aynı yöntemle Ctrl tuşu basılı tutulduğu sürece, istenildiği kadar kalıp kopyası elde edilebilir.

### 6.1.4. Pastala Yönelik İşlemler :

Tasarımı yapılan kalıpların sağlıklı bir şekilde pastala yerleşimini gerçeklemek için kalıplar üzerinde bazı çalışmaların yapılması gerekir. Menüde ve araç çubuğunda, bu amaçla bazı Yetenekler tanımlıdır. Bunlara ek olarak ayrıca araç çubuğunda çeşitli çalışma düzeyleri tanımlıdır. Bu işlevleri tek tek şu şekilde tanıtabiliriz:

#### 6.1.4.1. Kalıp Düz İpliği :

Düz iplik kavramı, pastal yerleşim planı hazırlanırken kalıpların kumaş üzerine belli bir doğrultuda yerleştirilmesini sağlamak için kullanılır. Düz ipliğin bir doğrultusu ve bir yönü vardır. Düz iplik yönü birinci noktadan ikinci noktaya doğrudur. Yerleşim planı hazırlanırken kalıpların düz iplik doğrultusu ile kumaş doğrultusunun aynı olmasına çalışılır. Pastala yönelik bir çalışma olan bu işlemi gerçeklemek için önce üzerinde çalışma yapılacak kalıp seçilir. Ardından da Nesne menüsü altında tanımlı Düz İplik Menüsü kullanılır. Eğer daha önce tanımlanmış düz iplik yapısı varsa, yeni oluşturulan düz iplik yapısı onun yerini alır. Çeşitli düz iplik tipleri mevcuttur ;

- **Serbest Düz İplik :** Düz İplik menüsü altında tanımlı Serbest Düz İplik menüsü kullanılır. Bu menü belirtildikten sonra seçili kalıp içinde fare sol tuşu basılıp sürüklenir ve bırakılırsa yeni bir düz iplik tanımlanmış olur. Bu şekilde oluşturulan düz iplik yapısı belli bir doğrultuya sahip değildir, doğrultu farenin serbest hareketi ile belirlenir.

- **Yatay Düz İplik :** Düz iplik menüsü altında tanımlı Yatay Düz İplik menüsü kullanılır. Bu menü belirtildikten sonra seçili kalıp içinde fare sol tuşu basılıp sürüklenirse yatay bir düz iplik belirir. Farenin yatay doğrultudaki iz düşümü düz iplik doğrultusunu oluşturur. Tuş bırakıldığında ise yeni düz iplik tanımlanmış olur.

- **Dikey Düz İplik :** Düz iplik menüsü altında tanımlı Dikey Düz İplik menüsü kullanılır. Bu menü belirtildikten sonra seçili kalıp içinde fare sol tuşu basılıp sürüklenirse dikey bir düz iplik belirir. Farenin dikey doğrultudaki iz düşümü düz iplik doğrultusunu oluşturur. Tuş bırakıldığında ise yeni bir düz iplik tanımlanmış olur.

### 6.1.4.2. Dikiş Payı (Seam) :

**Tanımı :** Dikiş payı, tasarıma uygun olarak kalıpların birbirine düzgün dikilebilmesi için kalıp profili etrafında bırakılan paylara verilen isimdir. Daha ziyade tasarımı bitmiş bir kalıp üzerinde uygulanan bir çalışmadır. Konfeksiyon sektöründe kullanılan çeşitli dikiş payı tipleri mevcuttur. Projede tanımlı dikiş payı tipleri P, DP, DA ve G 'dir. Diğer tipler ve ekstra tipler projenin ilerleyen versiyonlarında tanımlanacaktır.

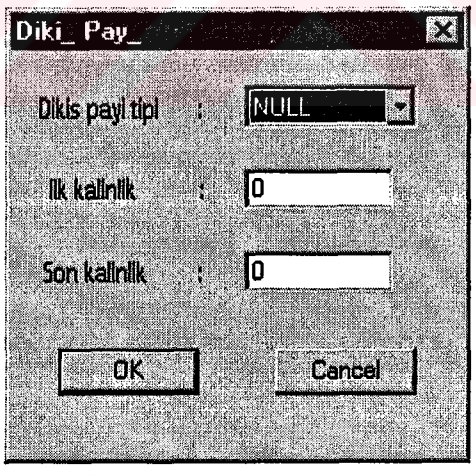
**Çözümü :** Kalıp üzerinde dikiş payı tanımı yapabilmek için önemli kriterler şunlardır ;

- Dikiş Payının Tipi
- Dikiş Payının İlk Kalınlığı ( mm - milimetre )
- Dikiş Payının Son Kalınlığı ( mm - milimetre )
- Ek Bilgiler :
  - Dikiş Payını Kısaltma Miktarı
  - Dikiş Payını Kısaltma Açısı vs.

Projemizde öncelikle gerçeklenen P, DP, DA ve G tipi dikiş payları için önemli olan kriterler ilk üç kriterdir. Öteki kriterler daha sonra tanımlanacak dikiş paylarına ilişkin kriterlerdir. Dikiş payı işlemi kalıbı oluşturan noktalar üzerinde tanımlanır ve noktalardan hareketle tüm kalıp için bir dikiş payı görünüşü elde edilir. Bu açıklamadan hareketle ; dikiş payı ilk kalınlığının seçilen noktaya yaklaşırken sahip olunan dikiş payı mesafesi olduğu ve dikiş payı son kalınlığının ise seçilen noktadan uzaklaşırken sahip olunan dikiş payı mesafesi olduğu söylenebilir. Bu tip bir dikiş

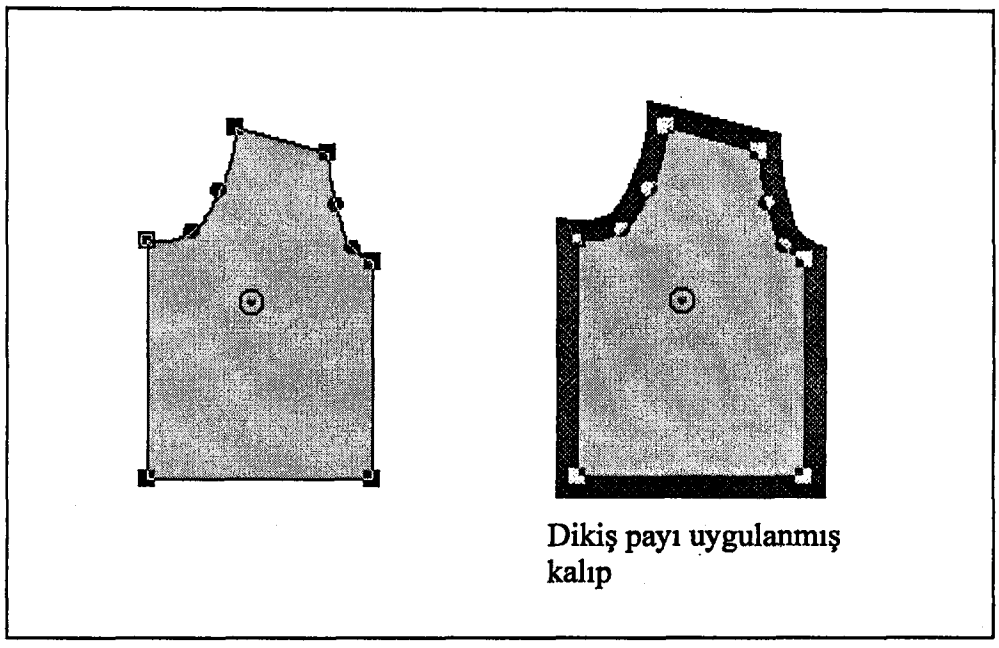
payı yapısının gerçeklenmesi için projede bir takım geometrik yaklaşımlar kullanılmıştır.

Kullanıcı dikiş payı oluşturabilmek için öncelikle dikiş payı işleminden etkilenecek kalıbı veya kalıpları belirtir ve araç çubuğunda tanımlı Dikiş Payı İşlem butonuna basarak Dikiş Payı Çalışma Düzeyine geçer. Ardından da yukarıda belirtilmiş olan kriterleri sırası ile yerine getirir. Eğer kalıp(lar) için daha önceden tanımlanmış bir dikiş payı bilgisi varsa bu çalışma düzeyine geçildiğinde bu bilgi dikiş payı görünüşü olarak kalıp(lar) etrafında belirir. Aksi takdirde dikiş payı tanımı olmayan kalıp noktaları NULL değerine sahip olur. Bu düzeye geçildiğinde fare sol tuşu desteğinde kalıp noktalarının dikiş payı bilgisi tek tek veya grup seçme ile tanımlanabilir veya değiştirilebilir. Bir kalıp noktası üstünde fare sol tuşu tıklandığı takdirde ekrana gelecek iletişim penceresinin yapısı şekil 6.15'da görülmektedir. Bu pencerede mevcut bilgiler yukarıda tanımlı kriterlerden olan dikiş payı tipi, dikiş payı ilk kalınlığı ve dikiş payı son kalınlığıdır. Bu bilgiler girildikten sonra TAMAM tuşuna basıldığında yapılan değişiklikler hemen ekranda belirir.




Şekil 6.15. Dikiş Payı İletişim Penceresinin Yapısı.

Bir kalıp üzerinde gerçeklenen dikiş payı işlemine örnek bir çalışma şekil 6.16'de görülmektedir. Örnekte ilk şekilde kalıbın hiçbir noktası için daha önce dikiş payı tanımı yapılmamıştır. İkinci şekilde ise aynı kalıbın noktaları için uygun dikiş payı tanımları yapılmıştır.




Şekil 6.16 Dikiş Payı Çalışması.

### 6.1.4.3. Serilendirme İşlemi (Grading):

**Tanımı :** Toplumu oluşturan bireylerin vücut ölçü değerleri çok çeşitlidir. Bu kadar çeşitlilik içinde tasarımlarda kullanılacak ölçü değerleri nasıl tespit edilecektir. Bu işlem için en akılcı yol ölçüm değerleri üzerinde gerçekleştirilen istatistikleri kullanmaktır. İstatistiklere göre normalde insanların boyları arttıkça bel, kalça, dirsek ve bilek gibi vücut ölçü değerleri artmaktadır. Bu açıdan bakıldığında toplumda çoğunlukta bulunan vücut ölçü değerleri standart alınarak ilk model tasarım ölçüleri tespit edilir. Belirlenen ölçülere göre tasarımı yapılan bu modele "temel beden" adı verilir. Tüm bireylere uygun modelleri gerçeklemek için bu temel bedene göre diğer beden setleri tanımlanmalıdır. Bu amaçla temel bedenden diğer bedenlere geçiş değerlerinin tanımlanması işlemine serilendirme adı verilir. Serilendirmede esas parça kalıptır ve işlem kalıplar üzerindeki noktalarda gerçeklenir.

Daha ziyade tasarımı bitmiş bir kalıp üzerinde uygulanan bir çalışmadır. Konfeksiyon sektöründe kullanılan çeşitli serilendirme tipleri mevcuttur. Projede

tanımlı serilendirme tipleri IG, ve IRG'dir. Diğer tipler ve ekstra tipler projenin ilerleyen versiyonlarında tanımlanacaktır.

**Çözümü :** Kalıp üzerinde serilendirme tanımı yapabilmek için önemli kriterler şunlardır ;

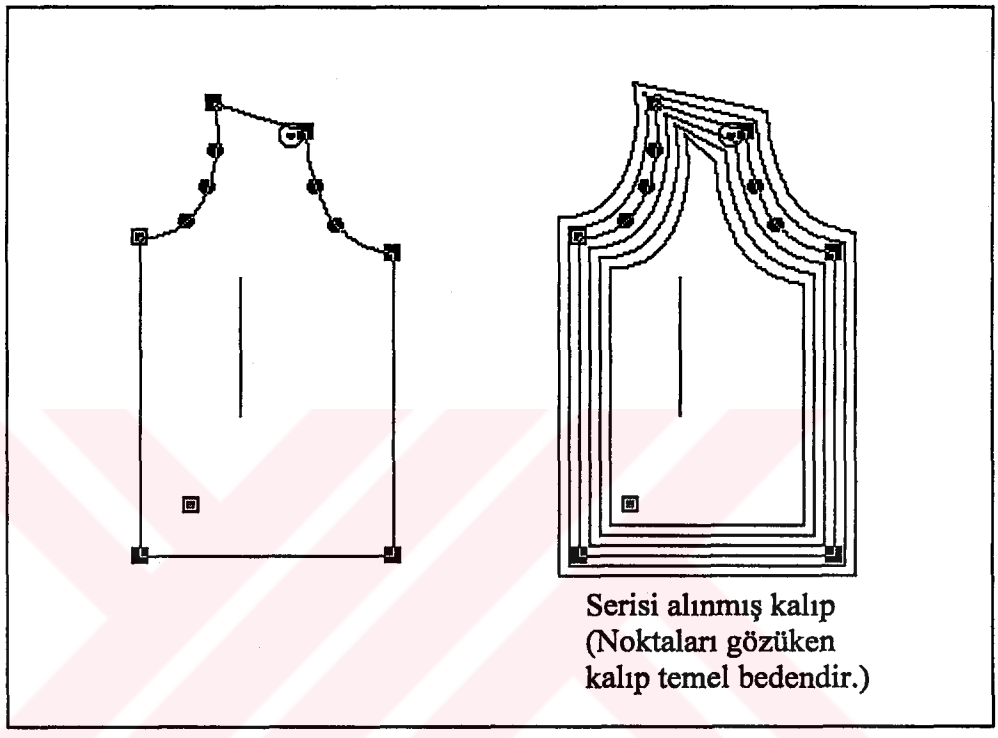
- Beden Setlerinin Dizisi
- Temel Beden
- Serilendirme Tipi
- X Sapması ( mm - milimetre )
- Y Sapması ( mm - milimetre )

Serilendirme işlemi kalıbı oluşturan noktalar üzerinde tanımlanır ve noktalardan hareketle tüm kalıp için bir serilendirme görünüşü elde edilir. Bu işlem esnasında serilendirme görünüşünün elde edilebilmesi için projede bir takım geometrik yaklaşımlar kullanılmıştır.

Modeli oluşturan kalıplar üzerinde serilendirme işlemini uygulayabilmek için öncelikle o modelin Veri_İşlemleri menüsünde yer alan Model Ouşturma menüsü yardımı ile kaydedilmiş olması gerekir. Çünkü bu işlem esnasında serilendirme kriterleri arasında yer alan ve seriledirme işlemi için çok önemli olan Beden Setlerinin ne olduğu ve Temel Bedenin hangisi olduğu gibi sorulara cevap verilir.

Kullanıcı, model bilgilerini tanımlayıp, kaydettikten sonra serilendirme işlemini gerçekleştirmek için öncelikle serilendirme işleminden etkilenecek kalıbı veya kalıpları belirtir ve araç çubuğunda tanımlı Serilendirme İşlem butonuna basarak Serilendirme Çalışma Düzeyine geçer. Ardından da yukarıda belirtilmiş olan kriterleri sırası ile yerine getirir. Eğer kalıp(lar) için daha önceden tanımlanmış bir seriledirme bilgisi varsa bu çalışma düzeyine geçildiğinde bu bilgi serilendirme görünüşü olarak kalıp(lar) etrafında belirir. Aksi takdirde serilendirme tanımı olmayan kalıp noktaları NILG (yani hiçbir seri bilgisi tanımlanmamış demektir) değerine sahip olur. Bu düzeye geçildiğinde fare sol tuşu desteğinde kalıp noktalarının serilendirme bilgisi tek tek veya grup seçme ile tanımlanabilir veya değiştirilebilir.

Bir kalıp üzerinde gerçeklenen serilendirme işlemine örnek bir çalışma şekil 6.17'de görülmektedir. Örnekte ilk şekilde kalıbın hiçbir noktası için daha önce seriledirilme bilgisi tanımı yapılmamıştır. İkinci şekilde ise aynı kalıbın noktaları için uygun seriledirme bilgisi tanımları yapılmıştır.




Şekil 6.17. Serilendirme işlemi çalışması.

### 6.1.5. Veri Tabanı İşlemleri :

Tasarımı yapılan modelleri daha sonra tekrar kullanabilmek için modele ilişkin bazı bilgileri depolamak gerekir. Bu depolama işlemi esnasında bir takım dosyalama yöntemlerine başvurulur. Bu amaçla önce model hangi firmaya ait ise o firmaya ilişkin bilgiler depolanır. Ardından da model depolama işlemi gerçeklenirken daha önce tanımlanmış firma bilgileri yardımı ile modelin hangi firmaya ait olduğu belirtilir.

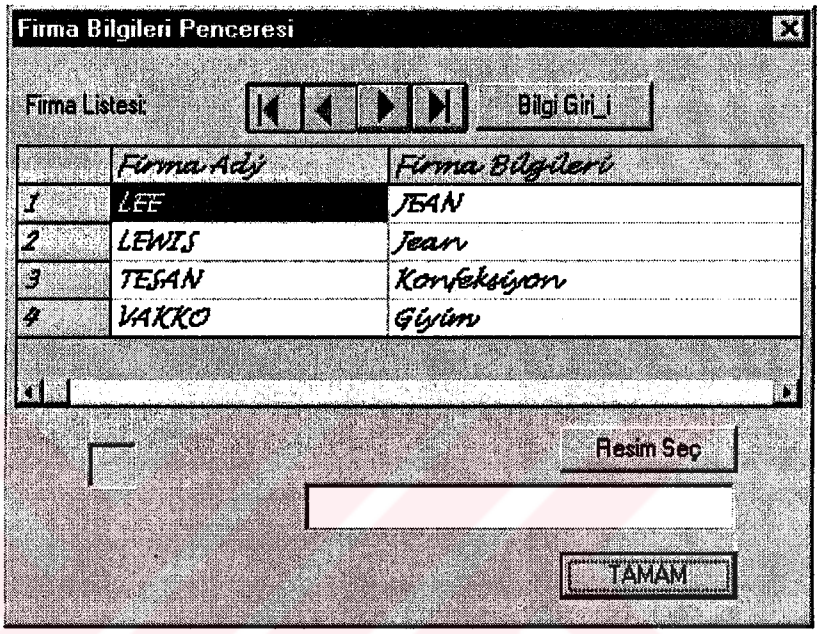
Kalıp tasarımında veri tabanı işlemlerini gerçeklemek için menü çubuğunda Veri İşlemleri menüsü tanımlanmıştır. Bu menünün kullanılabilmesi için öncelikle ekranda açık bir tasarım alanın bulunması gerekir. Aksi takdirde menü çubuğunda bu menü görülemez. Veri İşlemleri menüsü altında tanımlanmış yetenekler şunlardır;

1. Firmalar
2. Model Oluştur
3. Modeli Oku
4. Modeli Kaydet
5. Kalıp Bilgileri
6. Model Bilgileri

Bu işlevleri sırası ile şu şekilde tanıtabiliriz:

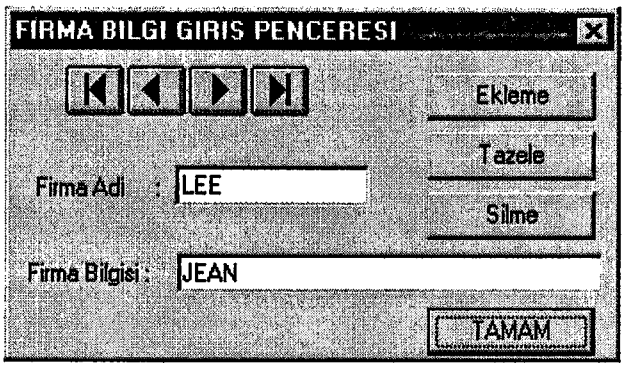
### 6.1.5.1. Firma Bilgileri :

Veri İşlemleri menüsü altında tanımlı Firmalar menüsü ile gerçeklenir. Bu menü seçildiğinde ekrana gelecek olan iletişim kutusunun yapısı şekil 6.18'de görülmektedir.




Şekil 6.18. Firma Bilgileri Penceresinin yapısı

Yukarıda görülen iletişim penceresinde firma adı, firma bilgileri ve görsellik sağlamak için her firmaya bir resim seçme olanağı tanımlanmıştır. Bu sayede daha önce oluşturulmuş firma bilgilerine ulaşılabileceği gibi firma resimleri de değiştirilebilir. Diğer firma bilgilerini değiştirmek veya yeni bir firma tanımlamak için Bilgi Girişi Butonuna basılması gerekir. Bu butona basıldığında Şekil 6.19'de görülen Firma Bilgi Giriş Penceresi ekranda belirir.

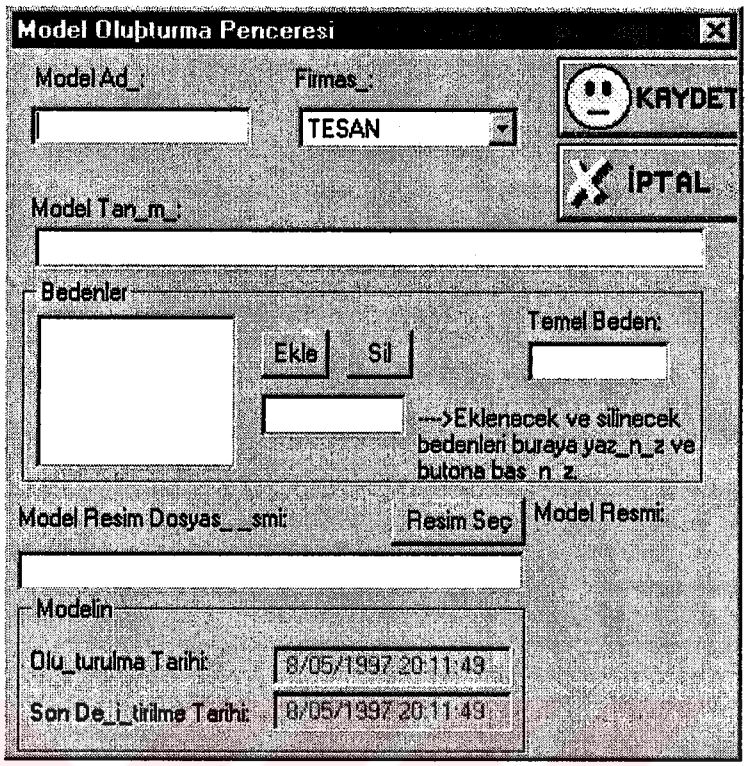



Şekil 6.19. Firma Bilgi Giriş Penceresi

Yukarıda yapısı görülen pencere sayesinde firma bilgilerinde bir takım değişiklikler yapılabilir. Firma Bilgi Giriş Penceresindeki Ekleme Butonuna basıldıktan sonra Firma Adı ve Firma Bilgisi girilir ve TAMAM butonuna basılırsa yeni bir firma veri tabanına eklenmiş olur. Ayrıca bu pencere sayesinde tanımı daha önce yapılmış firma bilgileri silinebilir.

### 6.1.5.2. Model Oluşturma :

Tasarımı yeni yapılmış bir modeli depolamak amacıyla kullanılır. Veri işlemleri altında tanımlı Model Oluşturma menüsü seçildiğinde ekrana gelecek olan iletişim kutusunun yapısı şekil 6.20'de görülmektedir. Bu iletişim kutusu ile model adı, model tanımı, modelin hangi firmaya ait olduğu, modeli oluşturma tarihi, modeli değiştirme tarihi ve model resmi gibi bilgiler tespit edilebilir. Ayrıca serilendirme işlemine yönelik bilgilerde belirlenebilir.

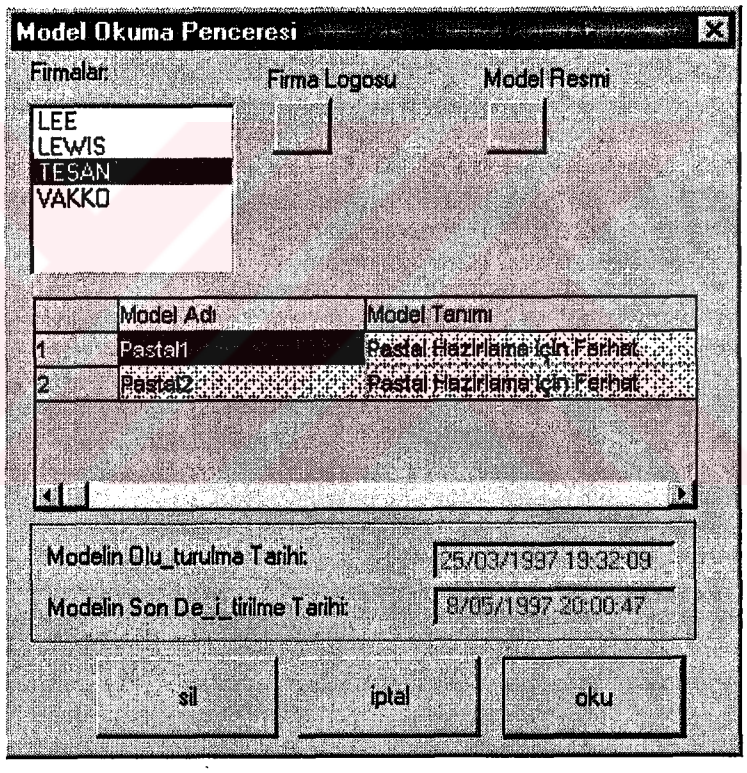



Şekil 6.20 Model Oluşturma Penceresinin Yapısı

Yukarıda yer alan iletişim penceresinin yapısını şu şekilde tarif edebiliriz ; Firma adlı kısım daha önce tanımlanmış firma adlarının seçilebildiği bir combo kutudur. Eğer oluşturulan model, tanımı yapılmamış bir firmaya ait ise önce Firma menüsü seçilerek firma tanımlanır. Bedenler adlı kısım serilendirme işlemine yönelik bilgilerin girilebildiği gurptur. Bu kısımda ortada yer alan giriş kısmına beden isimleri yazılır ve Ekle tuşuna basılırsa beden adları soldaki listeye eklenir. Var olan beden isimlerini silmek için listeden silinmek istenen beden adı seçilir ve Sil tuşuna basılır ise silme işlemi gerçeklenmiş olur. Temel Beden kısmında ise oluşturulan modelin hangi temel beden olduğu yazılır. Eğer bu Bedenler kısmına uygun bilgiler yazılmaz ise serilendirme işleminde bazı sorunlarla karşılabilir. Görselliği sağlamak için her modele bir resim seçilebilir. Alttaki Tarih bilgileri ise makina tarafından otomatik olarak tespit edilir.

### 6.1.5.3. Model Okuma :

Tasarımı ve depolama işlemi daha önce yapılan modellere ilişkin bilgilere ulaşabilmek için kullanılır. İstenirse seçilecek model, açılmış olan tasarım alanına eklenebilir veya veri tabanında tamamiyle silinebilir. Modeli silme işlemi ile yanlışlıkla oluşturulmuş veya artık kullanılmayacak modeller veri tabanından çıkarılmış olur. Veri işlemleri altında tanımlı Model Oku menüsü seçildiğinde ekrana gelecek olan iletişim kutusunun yapısı şekil 6.21'de görülmektedir. Bu menü ile istenen firmanın istenen modeline ulaşılabilir. Ardından da model okuma veya model silme gerçeklenebilir.



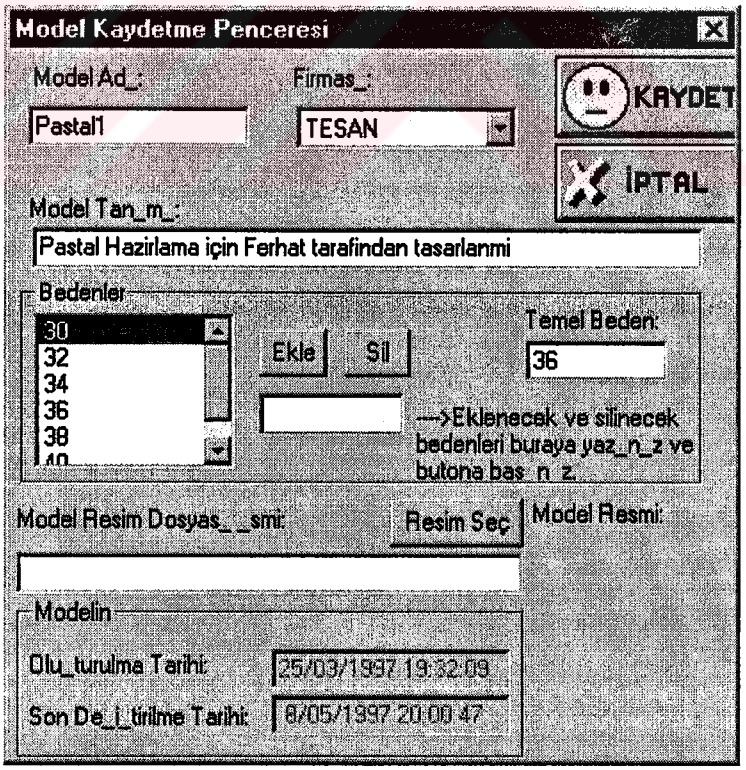

Şekil 6.21 Model Okuma Penceresinin Yapısı

Yukarıda yer alan iletişim penceresinin yapısını şu şekilde tarif edebiliriz ; Firmalar adlı kısım daha önce tanımlanmış firma adlarının seçilebildiği bir listedir. Listede seçilen firma adı değiştirildilçe aşağıdaki ızgara yapısında o firmaya ait modellere ilişkin bilgiler gelir. İletişim penceresinin altında yer alan tarih bilgileri ise ızgarada seçilen modele ilişkin bilgilerdir. Eğer en alttaki Sil tuşuna basılır ise seçilen model

veri tabanından tamamiyle silinir. Oku tuşuna basılır ise seçilen model açık ve aktif olan kalıp tasarım alanına eklenir. İptal tuşu sayesinde de işlemden vazgeçilebilir.

### 6.1.5.4. Model Kaydetme :

Tasarımı ve depolama işlemi daha önce yapılan modellere ilişkin bilgileri değiştirmek için kullanılır. Veri İşlemleri altında tanımlı Modeli Kaydet menüsü seçildiğinde Şekil 6.22'de görülen iletişim penceresi ekrana gelir. Bu iletişim penceresinin çıkması için daha önceden, aktif tasarım alanına bir modelin okunmuş olması veya yeni bir modelin tanımlanmış olması gerekir. Dolayısıyla iletişim penceresindeki bilgiler aktif tasarım alanında tanımlı en sonuncu modele ilişkin bilgilerdir. Gerekli değişiklikler yapılır ve Kaydet tuşuna basılır ise aktif tasarım alanında bulunan tüm kalıplar bu model tanımı altında veri tabanına kaydedilir.

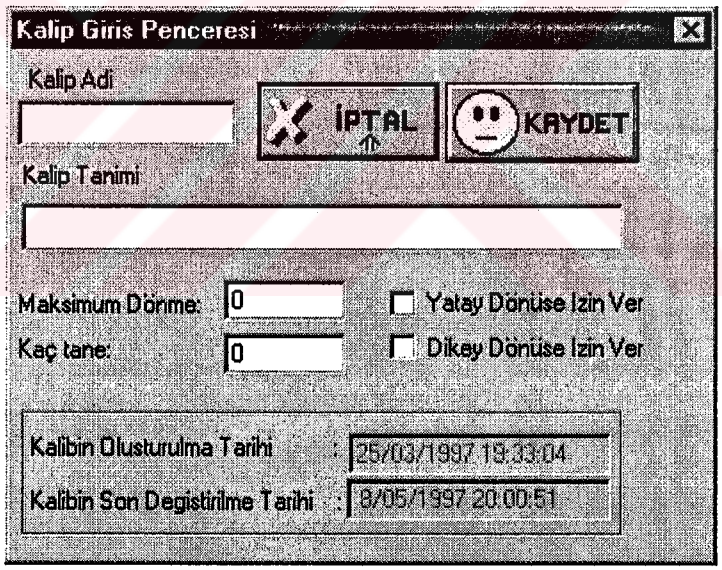



Şekil 6.22 Model Kaydetme Penceresinin Yapısı

Yukarıda görülen iletişim penceresinin yapısı şekil 6.20'de yer alan model oluşturma penceresinin yapısı ile aynıdır. Fakat bu iletişim penceresinde bulunan veri alanları boş değil aktif tasarım alanında mevcut modele ilişkin bilgilerdir.

### 6.1.5.5. Kalıp Bilgileri :

Modeli oluşturan kalıplara ilişkin bilgilerin düzenlendiği veri tabanı kısmıdır. Veri İşlemleri altında tanımlı Kalıp Bilgileri menüsü seçildiğinde Şekil 6.23'de görülen iletişim penceresi ekrana gelir. Bu iletişim penceresi sayesinde modeli oluşturan herbir kalıp parçası için tanımlamalar yapılabilir ve bu bilgiler veri tabanında depolanabilir.

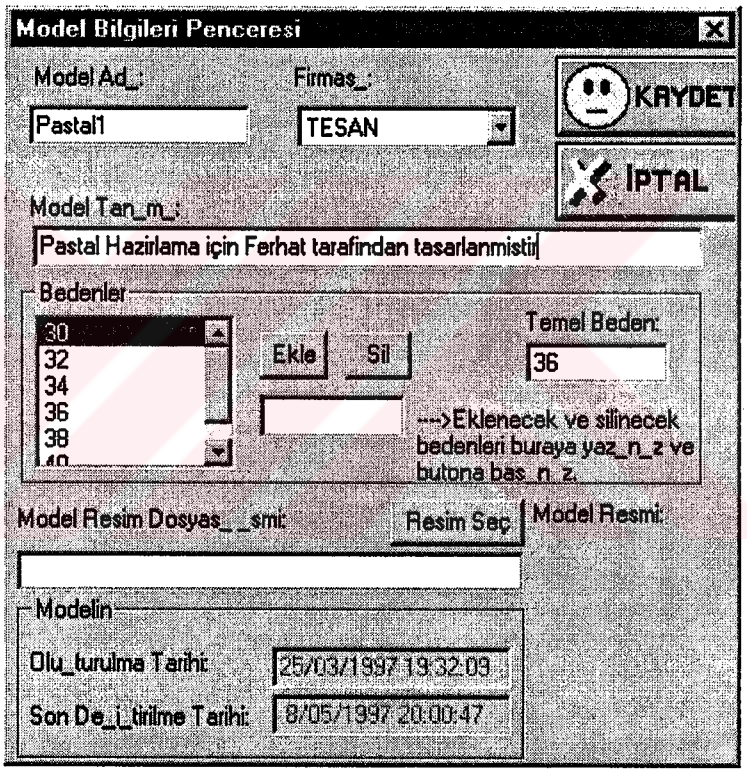



Şekil 6.23 Kalıp Giriş Penceresinin Yapısı

Yukarıdaki iletişim penceresi ile modeli oluşturan kalıplarında düzenlenmesi sağlanmış olur. Pencerede mevcut alanlarda tanımlı bilgiler aktif tasarım alanındaki seçili kalıba ilişkin bilgilerdir. Eğer alanlar boş ise daha kalıbın tanımı yapılmamıştır. Gerekli tanımlamalar veya değişiklikler yapıldıktan sonra Kaydet tuşuna basılır ise veri tabanına depolama işlemi gerçeklenmiş olur.

### 6.1.5.6. Model Bilgileri :

Tasarımı ve depolama işlemi daha önce yapılan modellere ilişkin bilgileri görüntülemek için kullanılır. Veri İşlemleri altında tanımlı Model Bilgileri menüsü seçildiğinde Şekil 6.24'de görülen iletişim penceresi ekrana gelir. Bu iletişim penceresinin çıkması için daha önceden, aktif tasarım alanına bir modelin okunmuş olması veya yeni bir modelin tanımlanmış olması gerekir. Dolayısıyla iletişim penceresindeki bilgiler aktif tasarım alanında tanımlı en sonuncu modele ilişkin bilgilerdir.



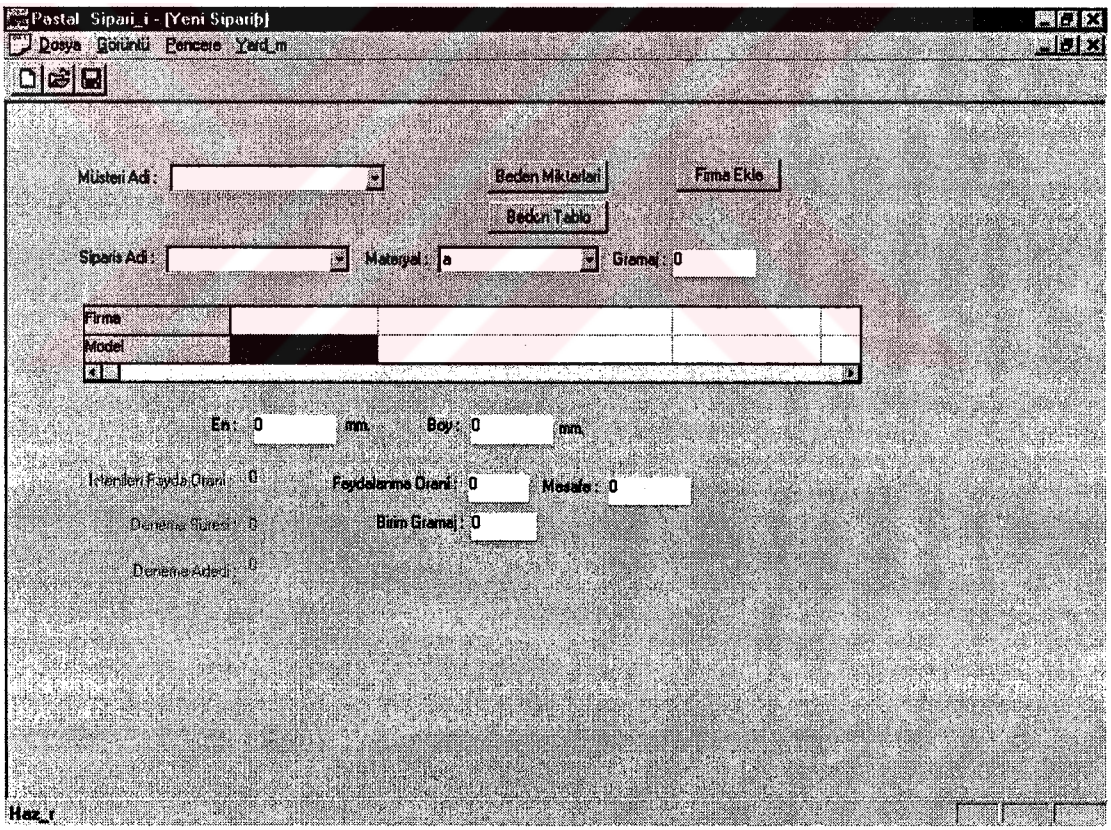

Şekil 6.24 Model Bilgileri Penceresinin Yapısı

Yukarıda görülen iletişim penceresinin yapısı şekil 6.20'de yer alan model oluşturma penceresinin yapısı ile aynıdır. Fakat bu iletişim penceresinde bulunan veri alanları boş değil aktif tasarım alanında mevcut modele ilişkin bilgilerdir.

Bu pencerenin tanımlanmasındaki amaç, sadece Model Bilgilerinin görüntülenmesidir. Fakat bu esnada da gerekli değişikliklerin yapılabilmesine olanak sağlamak için iletişim penceresinin yapısı Model Oluşturma ve Model Kaydetme penceresinin yapısı ile aynı tasarlanmıştır.

## 6.2. Pastal Siparişi İşlemleri :

Model tasarım modülünde modellerin tasarımı bittikten sonra bu modellerin giysiye dönüştürülmesi aşamasına gelinir. Bu aşamada aynı kumaş malzemesinden üretilecek kalıpları barındıran bir kalıp yerleşim planı yani pastal yerleşim planı hazırlanıp kesim odasına göderilir. Pastal yerleşim planı hazırlanırken hangi kalıpların hangi malzeme ile üretileceğinin bilinmesi gerekir. Bu amaçla Model Tasarım Modülü ile Pastal Planı Hazırlama Modülü arasında pastal siparişinin düzenlemesi için Pastal Sipariş Modülü adında yardımcı ara bir modül tanımlanmıştır. Bu bölüm ise adı geçen Pastal Sipariş Modülünün tanıtımını içermektedir. Pastal Sipariş Modülünün genel görünüş yapısı Şekil 6.25'te görülmektedir.




Şekil 6.25 Pastal Sipariş Modülünün görünüş yapısı.

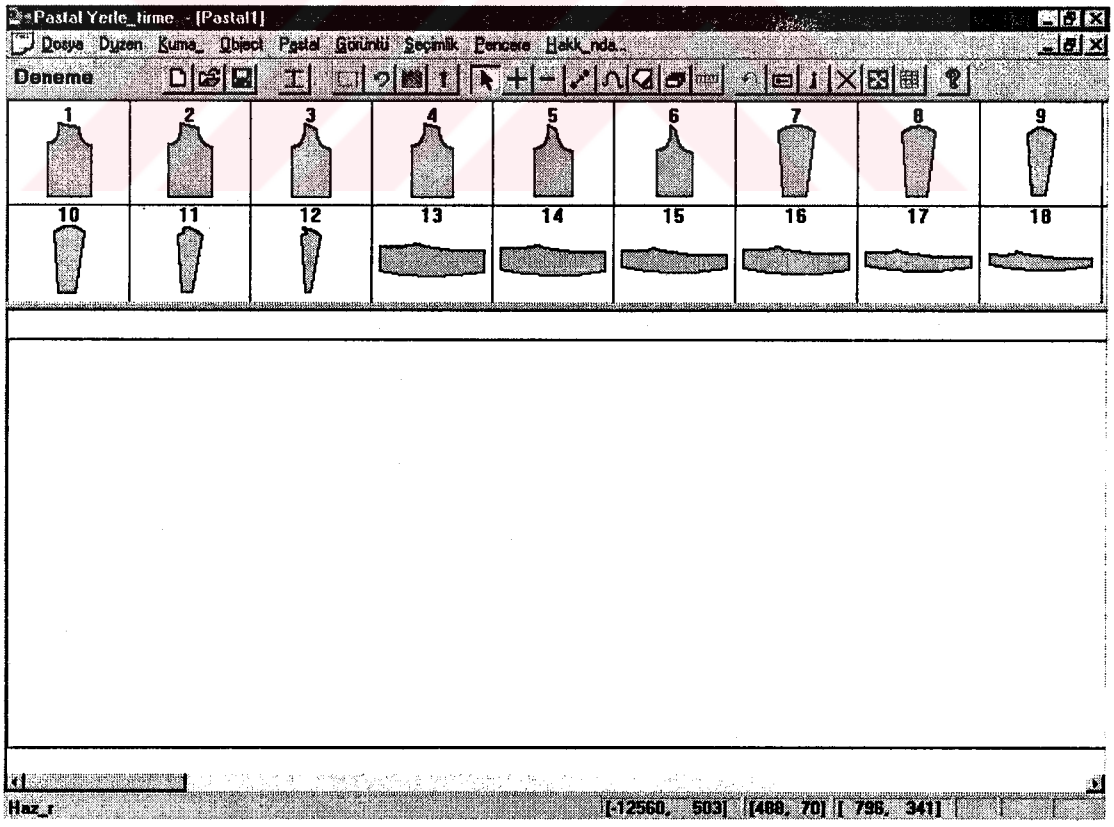
Pastal sipariş modülünün gürünüş yapısındaki beyaz boş alanlar kullanıcı tarafından belirlenecek yerlerdir. Müşteri adı ve sipariş adı alanları kullanıcı isteğine göre doldurulacak alanlardır. Diğer boşluklar daha önce tanımı yapılmış veya tanımı daha sonra yapılacak bilgilerle doldurulur. Materyal alanında mevcut materyallerden yani kumaş materyallerinden biri seçilir. Seçilecek olan materyal pastal planın hazırlanması sırasında kullanılacak olan malzemeyi belirler. Bu materyal kareli, çizgili kumaşta olabilir. Pastal siparişinde gramaj alanında ise seçilen materyalin gramaj değeri görünür. Firma ve Model ızgara yapısı sayesinde farklı farklı firmaların farklı farklı modelleri için pastal siparişi yapılabilir. Seçilen firma modellerini oluşturan kalıplar içinde aynı malzemeden üretilecek kalıplar Pastal Planı Hazırlama modülünde stok alanında görünecektir. Beden miktarları butonuna basılmak sureti ile seçilen firma modelleri için hangi bedenlerden ne kadar sipariş edileceği belirlenir. Aynı şekilde bu değerler de Pastal Planı Hazırlama modülünde kalıpların stok sipariş değerini belirleyecektir. En ve Boy bilgileri pastal yerleşim planı neticesinde elde edilecek bilgilerdir.

Bu açıklamalardan da açıkça anlaşıldığı üzere Pastal Sipariş Modülü sayesinde Pastal Hazırlama Modülünün stok değerleri ve malzeme bilgisi belirlenir. Firmaya ait modelleri oluşturan kalıplardan ne kadar üretileceği bu modülde belirlenir.

## 6.3. Pastal Planı Hazırlama İşlemleri :

Model tasarım alanında gerçeklenen modeller için pastal siparişi yapıldıktan sonra en son işlem pastal planının hazırlanmasıdır. Pastal planı, aynı malzemeden üretilecek model bileşenlerinin (kalıp, beden vs.) kumaş üzerine minimum kayıp ile yerleşimlerinin tasarlandığı veya düzenlediği bir çalışmanın sonucunda ortaya çıkan kalıp yerleşim planına verilen isimdir.

Pastal planını hazırlamak için kullanılacak proje modülünün ekranı iki kısımlı olarak tasarlanmıştır. Ekranın üst kısmında siparişi yapılan model bileşenlerinin siparişe uygun olarak seçilebilmesine olanak sağlayan bir "Stok Alanı", alt kısmında ise sipariş malzemesi olan kumaşın yer aldığı ve pastal yerleşiminin hazırlandığı "Pastal Alanı" vardır. Bu iki kısmı ayıran çubuğun konumu fare yardımı ile değiştirilebilir. İstenirse stok alanı tamamen küçültülüp sadece pastal alanı üzerinde çalışılabilir. Bu iki kısım hakkındaki geniş açıklamalar aşağıda başlıklar halinde anlatılmaktadır. Pastal planı hazırlama modülünün genel görünüş yapısı şekil 6.26'de görülmektedir.




Şekil 6.26 Pastal Planı Hazırlama modülünün görünüşü.

### 6.3.1. Stok Alanı :

Stok alanında modellerin bileşenleri (kalıplar, bedenler vs.) , düz ipliği soldan sağa doğru olacak şekilde iki sıra çalışma kutuları halinde görüntülenir. Herbir kutunun üst kısmında o kutuda yer alan kalıbın kaç adet sipariş edildiği yazılıdır. Bu değerler stok sipariş değeri adını alır.

Fare sol tuşu bir kutu üzerinde basıldığı zaman o kutuda yer alan kalıbın aşağıda yer alan pastal planına yerleştirilmek istendiği belirtilmiş demektir. Eğer pastal planında daha önce yerleşimi yapılmış kalıplar arasında herhangi bir örtüşme yok ise belirtilen kalıp aşağıdaki pastal planına geçer ve stok sipariş değeri bir eksilir. Aksi takdirde belirtilen kalıbı içeren kutu, seçimin algılandığını fakat yerleşimin bu aşamada gerçeklenemiyeceğini göstermek için renk değiştirir. Pastal planında kalıp örtüşmesi giderildikten sonra yukarıdaki seçili kutudaki kalıp ilk fare basımı ile pastal alanına geçer. Pastal alanına seçili kalıp değilde bir başkası yerleştirilmek istenirse daha önceden seçilmiş kalıbı değiştirmek için stok alanında istenen diğer kutu belirtilebilir. Stok alanında daha önceden yapılan seçim iptal edilmek istendiği takdirde stok alanında herhangi bir yerde fare sağ tuşunun tıklanması yeterli olur.

İstenirse stok sipariş adetinden daha fazla kalıp pastal planına yerleştirilebilir. Bu takdirde stok sipariş değerinden fazla yapılan herbir yerleşim için stok sipariş değeri eksi değerlere doğru ilerler.

Stok sipariş alanında yapılabilecek diğer fonksiyonel çalışmalar aşağıda maddeler halinde sıralanmıştır;

#### 6.3.1.1. Stok Değeri Bitmiş Olanları Göster / Gösterme İşlemi :

Stok adet değeri bitmiş (yani sıfır veya sıfırın altında) olan kalıpların stok alanında görüntülenmesi istenmeyebilir. Böyle bir işlemdeki amaç pastal siparişinde belirtilen çalışma yükünün ne kadarının yani sipariş edilen kalıpların hangilerinin yerleşiminin

tamamlandığını görebilmektir. Araç çubuğunda tanımlı butona basmak sureti ile stok adet değeri bitmiş olan kalıpların görüntülenmesi veya görüntülenmemesi sağlanabilir.

### 6.3.2. Pastal Planı Hazırlama Alanı :

Bu çalışma alanı direkt olarak pastal yerleşim planının hazırlandığı alandır. Tasarım ekranının alt kısmında yer alır. Alanın en ve boy bilgileri yerleşimin yapılacağı kumaş malzemesine ilişkin bilgilerdir. Kumaşın gerçek boy bilgisi yerleşim tamamlandıktan sonra belli olur.

Bu alanda kalıpların tek tek veya grup halinde kumaş üzerine minimum kayıpla yerleşimleri tasarlanır. Temel çalışma şekli stok alanındaki kalıpları fare yardımı seçip pastalda taşıma ve ardından da kalıbı sürükleyip yön vermek sureti ile kalıbı pastaldaki boşluklara yerleştirmektir. Eğer taşıma işleminden sonra kalıp sürüklenerek yön doğrultusu verilmez ise kalıp bırakılan yerde kalır. Bu sayede sadece kalıp taşıma işlemi gerçeklenmiş olur.

Pastal planı hazırlama modülündeki işlemleri şu gruplar altında toplayabiliriz;

1. Kalıplar Üzerindeki İşlemler
2. Kalıp Noktası Üzerindeki İşlemler
3. Pastal Üzerindeki İşlemler

Pastal yerleşimi tasarlanırken kullanıcıya sağlanmış fonksinel işlevler hakkında gerekli açıklamalar aşağıda başlıklar halinde ele alınmıştır.

### 6.3.2.1. Kalıplar Üzerindeki İşlemler :

Projede kalıplar üzerinde gerçeklenebilecek işlemler bu başlık altında toplanmıştır. Kalıplar üzerinde taşıma, yönlendirme, kesme gibi çeşitli işlemler gerçeklenebilir. Bu işlemler hakkındaki gerekli açıklamalar aşağıda yer almaktadır;

#### 6.3.2.1.1. Kalıp Taşıma ve Yönlendirme İşlemi :

Bir kalıp veya kutuluma özelliği ile kalıp grubu seçilirse fare hareketi sayesinde kalıplar taşınabilir. Fare hareketi kalıpların taşınmasına sebep olur. Bu işlem bir daha ki fare sol tuşu basımına kadar devam eder. Fare sol tuşuna basılıp sürüklendiği takdirde yönlendirme doğrultusunu gösteren bir çizgi belirir. Fare sol tuşu bırakıldığında yönlendirme çizginin gösterdiği doğrultuda kalıplar yönlendirilir ve uygun bir pastal boşluğuna yerleşir. Bu yerleşim esnasında kalıplar etrafında tanımlanan miktarda mesafe kalmasına gayret edilir. Kalıplar arası mesafe değeri Ayarlar menüsünde tanımlı Pastal Ayarları menüsü yardımı ile istenilen değere ayarlanabilir. Eğer kalıplar seçildikten sonra fare sol tuşuna bir daha basıldığında sürükleme işlemi gerçekelenmez ise kalıp yönlendirme işlemi de devre dışı kalır.

#### 6.3.2.1.2. Kesme İşlemleri :

Araç çubuğunda tanımlı Kalıp Kesme butonuna basılarak Kalıp Kesme Çalışma düzeyine geçilebilir. Ayrıca Nesne menüsünde tanımlı Kalıp Kesme menüsü altındaki menüleri de kullanarak kalıp kesme konusunda çalışmalar yapılabilir. Pastal planı hazırlama alanında gerçeklenebilecek üç değişik kalıp kesme şekli aşağıda yer almaktadır;

- **Serbest Kesme :** Araç çubuğunda tanımlı Kalıp Kesme butonuna basılarak Kalıp Kesme Çalışma Düzeyine geçilebilir. Bu çalışma

düzeyinde seçili kalıp veya kalıplar serbest fare hareketi ile belirlenecek kesme çizgisi sayesinde kesilebilmektedir. Eğer fare göstergesi seçili kalıp noktaları üzerine getirilirse gösterge şeklinin değiştiği görülecektir. Böyle bir işlem kalıbı noktalarından kesmek için kullanılmaktadır. Eğer kalıbı noktalarından kesmek istersek; fare göstergesi kalıp noktası üzerinde şekil değiştirdiğinde fare sol tuşuna basılır ve kesme noktasının belirtilen noktaya ne kadar uzak olduğu ekrana gelecek iletişim penceresi sayesinde belirtilirse kalıp noktalarından kesilir.

- **Yatay Kesme :** Nesne menüsü altındaki Kalıp Kesme menüsünde tanımlı Yatay Kesme menüsü ile Kalıp Kesme Çalışma Düzeyine geçilebilir. Fakat fare hareketi ile belirlenen kesme çizgisi yatay bir kesme çizgisidir. Bu kesme çizgisi ile kalıp veya kalıplar yatay bir şekilde kesilebilir. Kesme çizgisinin yatay y konumunu fare sol tuşunun ilk basım noktası belirler.

- **Dikey Kesme :** Nesne menüsü altındaki Kalıp Kesme menüsünde tanımlı Dikey Kesme menüsü ile Kalıp Kesme Çalışma Düzeyine geçilebilir. Fakat fare hareketi ile belirlenen kesme çizgisi dikey bir kesme çizgisidir. Bu kesme çizgisi ile kalıp veya kalıplar dikey bir şekilde kesilebilir. Kesme çizgisinin dikey x konumunu fare sol tuşunun ilk basım noktası belirler.

#### 6.3.2.1.3. Gruplama İşlemleri :

Stok alanında mevcut bir kalıp için yapılan tüm siparişlerin topluca yerleşimlerini sağlamak amacıyla tasarlanmış bir yetenektir. İki şekilde uygulanabilir ; serbest fare hareketi ile tüm sipariş ya tek tek ya da topluca belli bir eksende ve simetride yerleştirilebilir. Araç çubuğunda tanımlı Gruplama butonuna basılarak Gruplama Çalışma düzeyine geçilebilir. Bu işlemde uygulanabilen gruplama çeşitleri şunlardır;

- **Serbest Gruplama :** Araç çubuğunda tanımlı Gruplama butonuna basıldığında ekrana çıkan menüden Serbest Gruplama menüsü seçilirse Gruplama Çalışma düzeyine geçilebilir. Bu çalışma düzeyinde stok alanında seçilen kalıp için yapılmış tüm siparişler serbest fare hareketi ile tek tek yerleştirilebilir. Bu özellik sayesinde, yerleşim hazırlanırken tekrar tekrar stok alanında seçim yapmaya gerek kalmaz.

- **X Yönünde Yerleşim :** Araç çubuğunda tanımlı Gruplama butonuna basıldığında ekrana çıkan menüden X'te Yerleşim menüsü seçilirse Gruplama Çalışma düzeyine geçilebilir. Bu çalışma düzeyinde stok alanında seçilen kalıp için yapılmış tüm siparişler, serbest fare hareketi ile ilk kalıbın yerleştirilmesinden sonra pozitif X ekseni yönünde otomatik olarak yerleştirilebilir. Siparişler X ekseni yönünde otomatik olarak yerleştirilirken bir başka kalıpla herhangi bir örtüşme oluşursa veya pastal sınırları dışına çıkılırsa yerleşim durdurulur ve sadece örtüşme oluşana kadar yerleşimleri yapılan kalıplar kalır.

- **Y Yönünde Yerleşim :** Araç çubuğunda tanımlı Gruplama butonuna basıldığında ekrana çıkan menüden Y'de Yerleşim menüsü seçilirse Gruplama Çalışma düzeyine geçilebilir. Bu çalışma düzeyinde stok alanında seçilen kalıp için yapılmış tüm siparişler, serbest fare hareketi ile ilk kalıbın yerleştirilmesinden sonra pozitif Y ekseni yönünde otomatik olarak yerleştirilebilir. Siparişler Y ekseni yönünde otomatik olarak yerleştirilirken bir başka kalıpla herhangi bir örtüşme oluşursa veya pastal sınırları dışına çıkılırsa yerleşim durdurulur ve sadece örtüşme oluşana kadar yerleşimleri yapılan kalıplar kalır.

- **X Yönünde Simetrili Yerleşim :** Araç çubuğunda tanımlı Gruplama butonuna basıldığında ekrana çıkan menüden X'te Simetrili Yerleşim menüsü seçilirse Gruplama Çalışma düzeyine geçilebilir. Bu çalışma düzeyinde stok alanında seçilen kalıp için yapılmış tüm siparişler, serbest fare hareketi ile ilk kalıbın yerleştirilmesinden sonra pozitif X ekseni yönünde otomatik olarak yerleştirilebilir. Siparişler X ekseni yönünde

otomatik olarak yerleştirilirken bir başka kalıpla herhangi bir örtüşme oluşursa veya pastal sınırları dışına çıkılırsa yerleşim durdurulur ve sadece örtüşme oluşana kadar yerleşimleri yapılan kalıplar kalır. Bu gruplama türünde otomatik yerleşim esnasında kalıpların üç çeşit simetrisi alınabilir; X simetrisi, Y simetrisi, X-Y simetrisi.

- **Y Yönünde Simetrili Yerleşim :** Araç çubuğunda tanımlı Gruplama butonuna basıldığında ekrana çıkan menüden Y'de Yerleşim menüsü seçilirse Gruplama Çalışma düzeyine geçilebilir. Bu çalışma düzeyinde stok alanında seçilen kalıp için yapılmış tüm siparişler, serbest fare hareketi ile ilk kalıbın yerleştirilmesinden sonra pozitif Y ekseni yönünde otomatik olarak yerleştirilebilir. Siparişler Y ekseni yönünde otomatik olarak yerleştirilirken bir başka kalıpla herhangi bir örtüşme oluşursa yerleşim durdurulur ve sadece örtüşme oluşana kadar yerleşimleri yapılan kalıplar kalır. Bu gruplama türünde otomatik yerleşim esnasında kalıpların üç çeşit simetrisi alınabilir; X simetrisi, Y simetrisi, X-Y simetrisi.

**6.3.2.1.4. Kalıp Döndürme İşlemi :**

Pastal hazırlama alanında kalıpların döndürülmesine olanak sağlamak için tasarlanmış bir yetenektir. Seçili kalıp veya kalıplar ya klavye yardımıyla ya da bazı menüler yardımı ile döndürülebilir. Nesne menüsü altında tanımlı Kalıp Döndürme menüsü seçildiğinde ekrana gelen iletişim kutusunda kalıbın kaç derece döndürüleceği yazılmak sureti ile istenilen kalıp dönüşü elde edilebilir. Ayrıca Ctrl+Yukarı Ok tuşu ile saat işaretinin tersi yönde 10 derecelik ve Ctrl+Aşağı Ok tuşu ile saat işaretinin yönünde 10 derecelik kalıp dönüşü elde edilebilir. Dönme derecesi Ayarlar menüsünde tanımlı Pastal Ayarları menüsü yardımıyla istenilen değere ayarlanabilir.

### 6.3.2.1.5. Düzenleme İşlemleri :

Düzenleme menüsü altında Kalıp Tasarım modulünde olduğu gibi kesme, kopyalama, yapıştırma gibi özellikler tanımlanmıştır. Çalışma şekilleri benzemesine rağmen bazı temel farklılıkları vardır. Düzenleme menüsü altında tanımlı özellikler şunlardır ;

- **Kesme :** (Stoğa atma) Düzenleme menüsünde tanımlı Kesme menüsü ile bu işlem gerçekleştirilebilir. Seçili kalıp mevcut iken bu menü seçilirse seçili olan kalıp veya kalıplar pastal planından silinir ve yukarıdaki stok alanına atılabilir.

- **Kopyalama :** (Clipboard'a kopyalama) Pastal planında mevcut bir kalıbın kopyası çıkarılmak istenirse Düzenleme menüsünde tanımlı Kopyalama menüsü ile bu kalıpların kopyası clipboard'a kopyalanır. Clipboard'a yapılacak bir daha ki kopyalamaya kadar bu kopyalanan kalıplar clipboard'da kalır.

- **Yapıştırma :** Düzenleme menüsünde tanımlı Yapıştırma menüsü sayesinde daha önceden yapılan kopyalama işlemi ile clipboard'a kopyalanmış kalıp pastal planına alınır. Bu esnada da kalıbın stok sipariş değeri eksiltilir.

- **Silme :** Düzenleme menüsü altında tanımlı Silme menüsü sayesinde pastal planında mevcut kalıp veya kalıplar pastal planından ve stok alanında silinmiş olur. Stok alanından silinir yani o kalıpların stok sipariş değeri artırılmaz. Bu işlem özellikle kalıp kesme ile elde edilen kalıp parçalarının yok edilmesi için kullanılır.

#### 6.3.2.1.6. Düz İpliği Göster / Gösterme İşlemi :

Normal çalışma durumunda pastal yerleşim planında bulunan kalıpların düz ipliği karmaşıklığı önlemek için görüntülenmemektedir. Kalıp düz iplikleri üstünde bir takım çalışmalar yapılacak ise düz ipliği görüntülemek için araç çubuğunda tanımlı Düz İplik Göster / Gösterme butonuna basılarak gerekli işlevsellik sağlanabilir. Bu buton basılı kaldığı sürece kalıp düz iplikleri görüntülenir.

### 6.3.2.2. Kalıp Noktası Üzerindeki İşlemler :

Bu işlemler gerçeklenirken öncelikle üzerinde çalışma yapılacak kalıp veya kalıplar seçilir. Ardından da araç çubuğunda mevcut Çalışma Düzeyi belirtilip işlemler gerçeklenir. Bu kapsamda yer alan işlemler şunlardır;

#### 6.3.2.2.1. Noktaları Göster / Gösterme İşlemi:

Çalışma esnasında tasarım alanında görüntü karmaşasını önlemek için kalıp noktalarının görüntülenmesi istenmeyebilir. Zaten noktalar üzerinde işlem yapılmayacak ise noktaların görüntülenmesinin de bir manası yoktur. Bu amaçla araç çubuğunda Nokta Göster/Gösterme butonu tanımlanmıştır. Projede normalde noktalar görüntülenmemektedir. Eğer noktaların görüntülenmesi isteniyorsa araç çubuğunda tanımlı butona basılması gerekir. Bu sayede noktalar üzerinde daha rahat çalışılabilir.

#### 6.3.2.2.2. Nokta Taşıma :

Araç çubuğunda tanımlı Nokta Taşıma butonuna basılarak Nokta Taşıma Çalışma düzeyine geçilebilir. Kullanıcıya iki türlü kalıp noktası taşıma işlemi sağlanmıştır. İlki model tasarım modülünde olduğu gibi kalıp noktasını fare sol tuşu yardımı ile yakalayıp sürüklemek ve istenilen uygun bir yere bırakmak şeklinde çalışır. İkinici çalışma şekli de ölçekli bir nokta taşıma biçimidir. Ölçekli çalışmada bir iletişim penceresi yardımı ile seçilen noktanın şimdiki konumundan X ekseni doğrultusunda ve Y ekseni doğrultusunda ne kadar uzağa taşınacağı belirtilir. Hangi çalışma türünün kullanılacağı nokta taşıma çalışma düzeyine geçerken belirtilir.

#### 6.3.2.2.3. Nokta Ekleme ve Silme :

Araç çubuğunda tanımlı Nokta Ekleme veya Nokta Silme butonuna basılarak Nokta Ekleme Çalışma Düzeyine veya Nokta Silme Çalışma Düzeyine geçilebilir. Burada tanımlı nokta ekleme türleri ve nokta silme işlemi Model Tasarım modülünde tanımlanmış olan nokta ekleme türleri ve nokta silme işlemi ile aynıdır. Çalışma şekilleri de aynıdır. Bu konu ile ilgili gerekli açıklamalar model tasarım modülü hakkındaki açıklamalar bölümünde bulunabilir.

#### 6.3.2.2.4. Eğri - Düz Çizgi Dönüşümü :

Araç çubuğunda tanımlı Eğri - Düz Çizgi Dönüşüm butonuna basılarak Eğri - Düz Çizgi Çalışma Düzeyine geçilebilir. Burada tanımlı eğri - düz çizgi dönüşümü Model Tasarım modülünde tanımlanmış olan eğri - düz çizgi dönüşüm işlemi ile aynıdır. Çalışma şekilleri de aynıdır. Bu konu ile ilgili gerekli açıklamalar model tasarım modülü hakkındaki açıklamalar bölümünde bulunabilir.

### 6.3.2.3. Pastal Üzerindeki İşlemler :

Hazırlanan pastal yerleşim planı üzerinde gerçeklenebilen işlemler bu başlık altında bir araya getirilmiştir. Bu işlemlerin gerçeklenmesine yönelik komutlar menüde ve araç çubuğunda tanımlanmıştır. Bu işlemler hakkındaki detaylı açıklamalar aşağıda yer almaktadır ;

#### 6.3.2.3.1. Pastal Son Çizgisi :

Pastal üzerine yerleştirilen kalıpların kesiminin gerçekleştirilebilmesi için gerekli kumaş uzunluğunun ne kadar olduğunu gösteren bir sistem desteğidir. Pastal yerleşim planında kırmızı bir çizgi ile görüntülenir. Pastal son çizgisi aynı zamanda pastal verimliliğinin hesaplanmasında kullanılan kumaş boyunu belirler. Kalıp hareketleri esnasında pastal uzunluğunu tespit etmek için sistem tarafından sürekli güncel tutulur.

#### 6.3.2.3.2. Zoom İşlemleri :

Görüntü menüsü altında tanımlı Zoom menüsü sayesinde pastal planında istenilen zoom değeri elde edilebilir. Bu menü altındaki Zoom menüsü seçildiğinde ekrana gelecek iletişim kutusu sayesinde minimum çalışma miktarı milimetre olarak belirtilebilir. Aynı zamanda bu iletişim kutusunda o anki minimum çalışma miktarı milimetre olarak görüntülenir. Klavyedeki Artı (+) tuşu ile minimum çalışma değeri artılabilir, eksi (-) tuşu ile de minimum çalışma değeri azaltılabilir. Bu sayede kalıplara yaklaşılıp, uzaklaşılabilir. Ayrıca Görüntü menüsünde tanımlı Tüm Pastal menüsü sayesinde de tüm pastal yerleşim planı görüntülenebilir.

#### 6.3.2.3.3. Örtüşme Kontrolü :

Bu çalışma bir işlem değildir. Sadece kullanıcıya sağlanmış bir destektir. Pastal alanında mevcut kalıplar arasında bir örtüşme olup olmadığını sistem her kalıp hareketinde kontrol eder. Eğer bir örtüşmeye rastlanır ise bu durum kullanıcıya bir şekilde rapor edilir ve bazı işlemlerin gerçeklenmesi yasaklanır. Çünkü kesime hazır bir pastalda kalıplar arasında bir örtüşmenin olmaması gerekir.

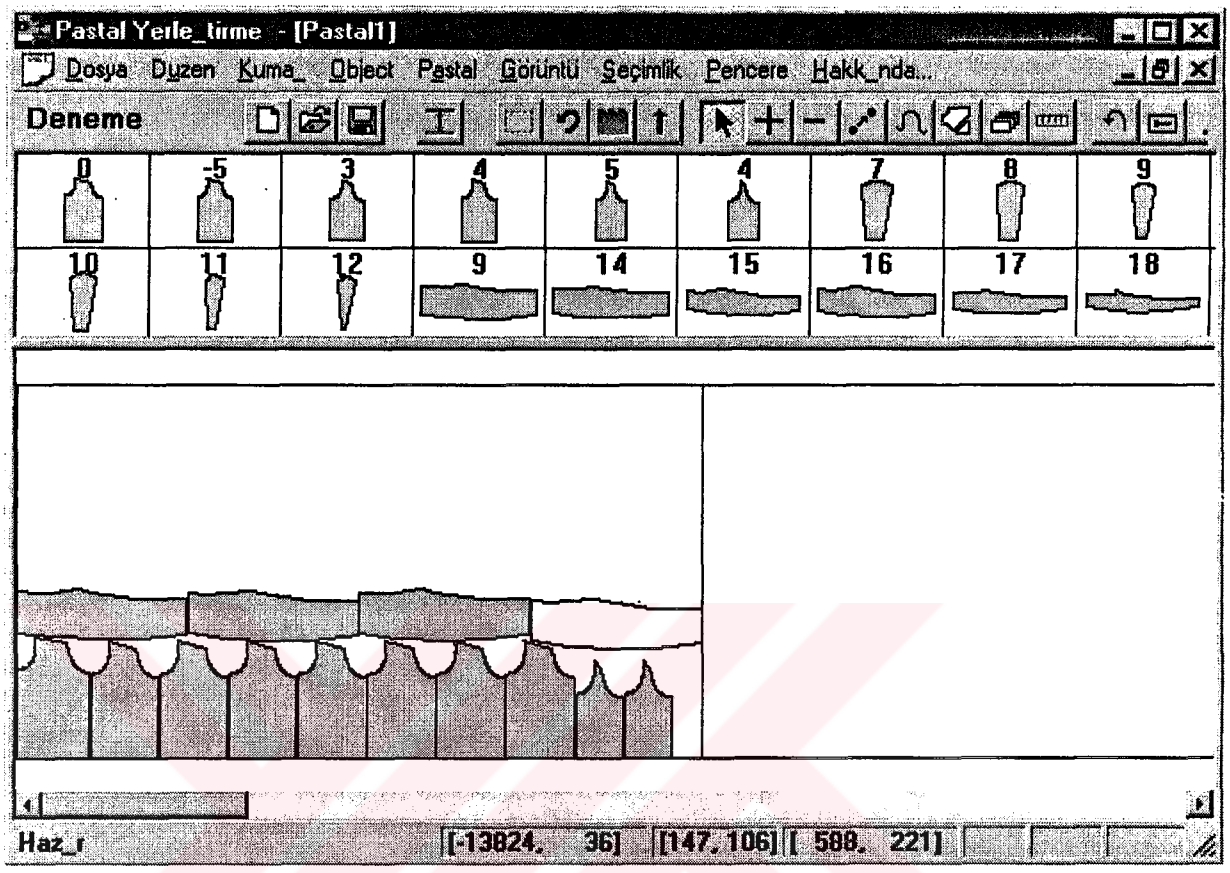
#### 6.3.2.3.4. Otomatik Pastal Hazırlama :

Stok alanında mevcut kalıpların pastal planını otomatik olarak hazırlamak için tasarlanmış bir yetenektir. Otomatik pastal planı hazırlanırken kullanılan algoritma maksimum verimlilik sağlamayı amaçlamaktadır. Pastal menüsünde tanımlı Otomatik Pastal menüsü sayesinde bu çalışma elde edilebilir.

#### 6.3.2.3.5. Pastal Verimliliğini Hesaplama :

Pastal verimliliği, pastal eni ve pastal son çizgisinin belirlediği pastal uzunluğu ebatlarında bir kumaş üzerine yerleştirilen kalıpların ne kadar verimli bir şekilde yerleştirildiklerini tespit eden bir sayısal değerdir. Pastal menüsü altında tanımlı Verimlilik menüsü seçildiğinde pastal verimliği hakkındaki gerekli açıklamaları içeren bir iletişim penceresi ekrana gelecektir.

Örnek bir Pastal Planı Hazırlama görüntüsü aşağıda şekil 6.27'te yer almaktadır;




Şekil 6.27 Pastal Planı Hazırlama çalışmasına bir örnek.

Yukarıdaki görüntüde üst kısım pastal sipariş modülünde siparişi yapılan kalıplara ilişkin stok alanı iken alt kısım pastal yerleşim planının hazırlandığı kısımdır. Alt kısım tamamiyle kumaşı simgelemektedir. Öyleyse iki yatay çizgi kumaş enini, kırmızı dikey çizgi ise kumaş boyunu göstermektedir. Siparişten fazla yerleşimi yapılan her kalıp için sipariş adet değeri eksi değerlere doğru gitmektedir. Bu durum stoktaki ikinci kalıpta görülmektedir. Stok alanında görülen kalıpların serisi, dikiş payı ve çıtı alınmıştır. Yerleşim yapıldıkça kumaş boyunu simgeleyen kırmızı çizgi güncellenir. İçi boş görülen kalıp ise o anda üzerinde işlem yapılan kalıptır. Üzerinde çalışılan kalıbın işi bitip sıra diğer bir kalıba geldiğinde eski kalıp içi dolu hale gelir. Eğer eski kalıbın pastal alanındaki herhangi bir kalıp ile örtüşmesi var ise o kalıp çizgili olarak görüntülenir.

# SONUÇLAR ve ÖNERİLER

Konfeksiyon sektöründeki ihtiyaca cevap verecek şekilde güncel bilgisayar uygulama geliştirme ortamları kullanılmıştır. Uygulamanın kullanıcı dostu olmasına yani uygulamanın çeşitli kademelerinde yön verici açıklamalarla kullanıcıya yardımcı bir otomasyon yazılımı olmasına dikkat edilmiştir. Bu sayede az deneyime sahip kullancılarında bu otomasyon uygulamasını rahatça kullanabilmesi ve piyasada daha geniş bir pazara sahip olması hedef alınmıştır.

Otomasyon yazılımında bir bilgisayar destekli tasarım ( CAD ) ve bilgisayar destekli üretim ( CAM ) çalışması gerçeklenmiştir. Sonuç çıktı olarak pastal yerleşim planını içeren bir plotter çizimi elde edilmektedir. Daha sonra bu plotter çizimi pastal raporunda belirtilen miktarda üst üste serilmiş kumaş tabakaları üzerine konulmakta ve pastalda görülen model kalıplarının kesimi elle idare edilen kumaş kesicilerle gerçeklenmektedir. Bu işlem esasında piyasa ihtiyacı olan daha ileri bir bilgisayar destekli üretim aşaması olan bilgisayar destekli kesici ( cutter ) vasıtasıyla otomatik olarak gerçeklenebilir. Fakat mali külfetinin büyüklüğü sebebiyle bu çalışma aşaması projede gerçeklenmemiş ve diğer projelerde gerçeklenebilir olarak bir yere not edilmiştir.

Kalıp tasarım ve pastal üretimini içeren uygulamalara Cam, Deri, Ayakkabı, Gemi gibi diğer sektörlerde de ihtiyaç duyulmaktadır. Gerçekleştirilen bu otomasyon yazılımı üzerinde yapılacak değişikliklerle söz konusu sektörlerin ihtiyacına cevap verecek ürünlere dönüştürülebilir.

Tez kapsamında tanıtımı yapılan otomasyon yazılımı bir proje grup çalışması olarak kamu ve özel sektör desteğini almış ve üniversite ortamının içinde olmanın getirdiği olanaklardan da yararlanmış bir çalışmanın ürünüdür. Bu açıdan bakıldığında böylesi çalışmaların hayata geçirilmesi için mutlaka piyasada faaliyet gösteren firmaların bu tip çalışmalara destek olması gereklidir.

# KAYNAKLAR

[1] **Charles Petzold**, Programming Windows, Microsoft Press, 1990

[2] **M. Erhan Sarıdoğan**, C++ ve Nesne Yönelik Programlama, Sistem Yayıncılık, 1994

[3] **Herbert Schildt**, Teach Yourself C++, Osborne McGraw - Hill, 1992

[4] **Mark Andrews**, Visual C++ Object - Oriented Programming, Sams Publishing, 1993

[5] **Namir Clement Shammas**, Secrets of the Visual C++ Masters, Sams Publishing, 1993

[6] Microsoft Press, Introducing Visual C++ Microsoft Visual C++ Development System for Windows and Windows NT Version 2.0, 1994

[7] **Patrick Taylor**, Giyim Endüstrisinde Bilgisayarlar, Gaye Filmcilik Matbaacılık San. ve Tic. A.Ş., M.E.B. ( Milli Eğitim Bakanlığı ), 1995

[8] Ant Coorperation, User's Guide For Pattern Design, Grading and Markermaking.

[9] **Andrew S. Glassner**, Graphics Gems, Academic Press, 1990

[10] Schime Seiki MFG., LTD, PGM –2, August 1991

# ÖZGEÇMİŞ

Ferhat Torgalöz 1972 yılı İstanbul doğumludur. Liseyi İstanbul Şehremini Lisesinde okudu. 1989 senesinde buradan mezun oldu ve İstanbul Teknik Üniversitesi Uçak ve Uzay Bilimleri Fakültesi Uçak Mühendisliği bölümüne kayıt oldu. Bölümün ilk sınıfında bölüm birincisi olarak İstanbul Teknik Üniversitesi Kontrol ve Bilgisayar Mühendisliği bölümüne yatay geçiş yaptı. Lisans diplomasını 1994 senesinde aldı. Aynı sene İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü Kontrol ve Bilgisayar Mühendisliği bölümünde yüksek lisans çalışmasına başladı. Halen İstanbul Teknik Üniversitesi KOSGEB Teknoloji Geliştirme Merkezinde Araştırma-Geliştirme Mühendisi olarak çalışmaktadır.