İSTANBUL TEKNİK ÜNİVERSİTESİ ✶ FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR ORTAMINDA TASARLANMIŞ PLANLAR İÇİN OTOMATİK OLARAK CEPHE ÜRETİLMESİ

YÜKSEK LİSANS TEZİ

Mimar Burhan BAHÇECİOĞLU

Tezin Enstitüye Verildiği Tarih : 16 OCAK 1995
Tezin Savunulduğu Tarih : 07 ŞUBAT 1995

Tez Danışmanı : Prof. Dr. Nigan BAYAZIT
Diğer Jüri Üyeleri : Prof. Dr. Nihat TOYDEMİR
: Doç. Dr. Gülen ÇAĞDAŞ

ŞUBAT 1995

ÖNSÖZ

Program AutoCAD ile çizdirilmiş bir plan cephesinin çiziminin otomatik olarak yapılmasına duyulan ihtiyacı gidermek için hazırlanmıştır.

Program dili olarak AutoCAD Release 12 üzerinde çalışabilen AutoLISP programlama dili seçilmiştir. AutoCAD'in geliştirilebilme özelliğinin AutoLISP ile kullanılabilmesinden dolayı bu dil seçilmiştir. AutoLISP, listeleme esasına dayalı LISP (List Processing) programlama dilinin geliştirilerek üzerine AutoCAD komutlarının ilavesi ile geliştirilmiş bir dildir.

Program AutoCAD'in vermiş olduğu imkanlardan bazılarının kullanılabileceği düşünülerek yazılmış ve bunların kullanımı ile daha kolay ve kullanışlı hale getirilmeye çalışılmıştır. Bunlardan Programlanabilir Diyalog Kutusu (Programmable Dialogue Boxes) olanağı kullanılıp, her türlü bilgilerin bu diyalog kutularından girilmesi sağlanmıştır.

Yüksek Lisans Programı süresince her türlü desteğini esirgemeyen ve Programın geliştirilmesi esnasında değerli bilgi, görüş ve tecrübelerini aktaran tez danışmanım Sayın Prof. Dr. Nigan BAYAZIT'a ve bu çalışmada emeği olan her kişi ve kuruluşa TEŞEKKÜR EDERİM.

Burhan Bahçecioğlu
1995 ŞUBAT

İÇİNDEKİLER

ŞEKİL LİSTESİ

ÖZET

AutoCAD ortamında çizilmiş bir bina planının ilk eskiz cephelerinin otomatik olarak çizdirilmesi hedeflenmektedir. AutoCAD üzerinde çalışabilen, AutoLISP programalama dili kullanılarak yazılan, otomatik cephe çizimine yardımcı olabilecek bir program geliştirilmektedir.

1. Bölümde çalışmanın amaçları açıklandıktan sonra, konunun önemi ve BDT'nin (Bilgisayar Destekli Tasarım) kullanımı ile yapılabilecek işlemlerden kısaca sözedilmektedir. Bilgisayar kullanımının getireceği kolaylıklar da kısaca ele alınmıştır. Ayrıca program mantığı hakkında kısa bir açıklama yapılmaktadır.

2. Bölümde Bilgisayarın yapısı hakkında kısa bilgiler verilerek, Teknolojideki gelişmenin Bilgisayar Destekli Mimari tasarım üzerindeki etkileri anlatılmaya çalışılmıştır.

3. Bölümde Literatürde bilgisayarla bina cephelerinin tasarımdaki çalışmalar taranmış ve konu ile ilgili bulgular ve kullanılan teknikler anlatılmaya çalışılmıştır.

4. Bölümde programın yazılış sırasındaki kabuller, metodu ile ilgili bilgiler gerekli ve açıklayıcı olacak şekiller kullanılarak anlatılmaya çalışılmıştır. Burada Akış Diyagramı yardımı ile programın genel mantığı hakkında genel bir fikir verilmiştir. Programın kullanımı için gerekebilecek kullanım kılavuzu ve programın kullanımı için gerekli önerilerden bahsedilmiştir.

Sonuçlar bölümünde ise programın kime yönelik olduğu ve geliştirilebilme yönleri konusunda bilgi verilmektedir.

SUMMARY

The aim of this work is to attain automatic drawing of preliminary sketches of facades from a building plan drawn using AutoCAD. This work has been performed to satisfy the need for automatically drawing for the facades of a plan. The basic goal is the minimising of the time and labor that are required by normal methods for creating the facades of any plan drawn using AutoCAD. The starting point was the idea that the user should not have to specify again the window and door places and the external contours of the building for drawing a facade after specifying these once in the plan. The facades created by the program will be the first preliminary sketches and the real facade will be formed by the interventions of the user to these preliminary sketches.

AutoLISP has been chosen as the programming language. AutoLISP is a language formed by adding the AutoCAD commands to LISP, which is a list oriented language. Application programs running on AutoCAD can be written using this language, and these run free of problems. This work is an application program which can serve as an example to such applications. Some of the main reasons for choosing AutoLISP as the programming language are:

-The program will produce facades in AutoCAD for plans created also with AutoCAD,
-AutoCAD commands can be called from this language,
-Dialogue boxes can be employed from within AutoLISP.

Also, another facility of AutoCAD R12, DCL (Programmable dialogue boxes) has been employed. Using DCL was beneficial because data entry is very easy and correct, and the values assigned to variables are displayed on the screen. Through this facility, the user may make all modifications until the last moment and change the values assigned to the variables in anyway he/she desires. This operation is very important because the user sees the values entered simultaneously in the screen and can modify these up to the last moment.

All data required by the program during the drawing is obtained from the plan, and during the drawing of the facade, from the user. The data obtained from the plan are the outer wall levels and door and window places. The user is prompted to enter other values that may be required during the drawing of the facade. Some of these are as below:

- Text to be displayed below the facade belonging to the facade being drawn.
  - Which project does the facade belong to
  - The direction of the facade
  - How the text will be written
  - The text style to be used
  - The height of the text
  - The angle of the text
- The floor name of the facade being drawn
  - Ground floor
  - Normal story
  - Roof
- Whether there will be a roof over the last floor
- Scale to determine the level of detail of the drawing:
  - 1/50
  - 1/100
  - 1/200
- Number of stories (1 if the ground floor to be drawn)
- Height of the story
- Height of the parapet
- Height of the window
- If a normal floor is being drawn, the bottom elevation of the 1st Normal floor
- The direction of the facade to be drawn
  - North
  - South
  - East
  - West
- Data related to the windows in the facade
  - The thickness of window frame
  - The number of vertical window sections
  - The situation of the vertical sections in the windows (equal or differing intervals)
  - If applicable, opening leafs
- Data relating to the roof
  - Roof parapet height (the default value is equalized to the window parapet height)

- Roof eaves width
- Bottom elevation of the roof parapet (not required if the roof is to be drawn with the floors)

After the above values are collected from the user via dialogue boxes, the drawing for the specified direction will be displayed on the screen.

As will be mentioned in subsequent literature review sections of this work, there are some techniques developed to allow the computer to perceive the drawn objects by recognition and to perform the desired operations using them. These techniques are:

- Edge Detection
- Contour Following
- Form Recognition

The method used in this work is similar to the "Contour Following" technique. But, the method used is not exactly the same. According to the new technique developed, objects which are walls are determined from the plan of the building. The obtained wall objects are screened through a second process leaving the wall objects belonging to the desired direction. In order to determine the outer contours, the outmost edges of the obtained wall objects were used. The middle points of the same object besides the two edges are used for the door and window places. The facade is created by combining window door levels extracted from the plan with all the data obtained from the user concerning the front.

Hence, the chapters, subject titles and contents reviewed in this work is as below:

In the 1st Chapter, a short comparison between the traditional methods and usage of computers is made and the possibilities the computer offers to the user during the production of a facade are discussed. Besides this some summary information is provided for the jobs to be performed with this work and the facilities it will offer the designer. The aim here is to further the automation of the drawing of facades which are currently drawn as a normal plan, and to provide a way for easily drawing the automatic facade of a plan already drawn and its alternatives and to satisfy the need for automatic drawing of facades. Even though a complete front cannot be drawn exactly in the manner contemplated, a

partial drawing is made automatically with the parametric values being entered by the user and others being extracted from the plan.

In the 2nd Chapter, some information about the computer system is furnished. To achieve this the structure of the computer system was examined and the fundamental change in architectural applications due to the daily usage of CAD techniques were explained. The fact that many factors play a role in this operation was explained. Then, the subject of components of a computer system was considered and information about the components was supplied. The computers were defined as electronic tools that are able to store data in their internal memory and to use command series from these stored data known as programs to perform these operations. Some short explanations about some of the components of a computer system; hardware, central processor, data and software; were included. A typical computer hardware was illustrated and short explanations about the computer hardware were provided. It was explained that the whole system is controlled by a CPU directed by an arithmetical and logical unit. It was stated that any device which converts any type of data to electric signals; a punch card reader, a keyboard, a TV camera, a microphone or sensor or any other device; can be used as a computer input device and that programs are a series of commands written in a computer language. Simple and complicated programs were illustrated by figures. Finally in this chapter, the effect of technological development on the possibility of Computer Aided Architectural Design was discussed.

In the 3rd Chapter, information about the existing literature concerning the design of building facades with a computer was provided and computer graphics in the definition of building facades were considered. It was stated that the role of computer graphics in CAD is a controversial subject, and that some people view computer graphics and CAD to be synonymous, others argue that advanced and encompassing research should be undertaken before computer graphics can be used for designing. It was explained that others claim even the simplest of computer graphics results in large savings in design costs. Also the subjects of syntax, object recognition, elevation studies, picture libraries and three dimensional graphics were reviewed under short headings. Some information was provided concerning to the subject of descriptive design under the heading of data-based building elevation formation efforts. Also the benefits of shape recognition were discussed, and edge detection, contour following and form recognition, which are some of these studies, were explained.

In the 4th Chapter, the methodology, the user manual of the program and the flow charts were explained. The methodology section points out the aim of the work and the operations to be performed. Where and how the program obtains the data was explained. The operations to be performed with the running of the program were explained using figures and dialogue boxes. The data required for the program to run and the default values that will be used in the absence of these were listed step by step. After that, the subroutines used in the program were listed in the order they were used in the program. In the section explaining the user manual of the program, the steps necessary to load the program from AutoCAD and the operations required to run the program were specified. The usage of each variable in the dialogue box was explained, and the data type to be entered to ensure correct operation of these were pointed out.

In the results chapter it was stated that the program will minimise the time and effort to be expanded by the designer for drawing during the preliminary sketch. Consequently, it is possible for the designer to utilise the time required for drawing instead for designing, to develop more alternative facades. Also, in this section it was stated that the structure of the program allows expansion, and additions would improve the program. It was explained that with the program all windows belonging to each area can be drawn in separate and differing forms, dimensions and shapes. The program can be used as an editor to make the necessary modifications to both the general contours of the facade and the window modules. It was explained that the program could be made to draw a roof in a facade using a roof plan of a project.

In the appendices the source code of the program and the text of the dialogue box used in the program were included.

BÖLÜM 1. GİRİŞ

BDT (Bilgisayar Destekli Tasarım) ile çizim zannedildiği gibi veriler ve istekler bilgisayarlara verildiğinde, tasarımın tümü isteklere göre bilgisayar tarafından yapılıp kağıda çizdirilmesi şeklinde değildir. Yine bütün iş Mimar'a ve onun becerisine kalmaktadır. Önemli olan çizimin kağıt, kalem ve T cetveli kullanmadan yapılmasıdır. Kağıt yerini ekrana, kalem yerini mouse'a veya klavye'ye bırakmakta, T cetveli ise tamamen kaldırılmaktadır. Aynı işi kağıt yerine ekrana çizmek ilk bakışta avantajlı gibi görünmeyebilir. Ancak çizim zorluğu aynı derecede bile olsa bilgisayar ortamında çizim yaparak, saklamanın getireceği bir çok avantajlar vardır. Sadece çizimi gerektiği zaman, tekrar istenilen ölçekte çizdirebilmek bile bilgisayar ve BDT programlarını kullanmak için yeterli bir nedendir. Bununla birlikte elde edilecek kazanç sadece bu değildir. Bilgisayarla çizim yapmak, hem daha kolay hem de kolaylığı dışında getireceği bazı avantajlar vardır. Örneğin biribirinin aynı olan bir elemanın sadece bir tanesi gerekli detayda çizilerek, diğerlerinin ilk çizilenden çok kolay bir şekilde üretilmesi sağlanabilir. Eğer elemanlar tamamen aynı değil, fakat aralarında benzerlik varsa, yine de yeni elemanı baştan çizmenin bir anlamı yoktur. Bunun için de yine bir öncekinden bir kopya alınarak yeni eleman için gerekli düzeltmeler yapılarak çizilebilir. Bunların dışında aynı elemandan çok sayıda var ise; çok kolay olarak bu elemanın gerekli yerlerde ve sayılarda kopyaları çıkarılabilir.

El-Göz yeteneği ve geleneksel grafik medyanın olmaması, bilgisayar grafikleri açısından bir sorun değildir. Esas amaç, kalem-kağıt ve geleneksel El-Göz çiziminde olduğu gibi grafiksel gösterim yapabilecek, tasarım işleminin başlangıç aşamalarında kullanılmak üzere tasarımcılar için bir araç geliştirmektir. Kullanıcıların klavye ile yazdıkları veya bir çeşit menüden seçtikleri gibi veya önce girip, daha sonra işletilebilecek uzun bir dizi oluşturulabilir. Burada popüler AutoCAD programlama dili AutoLISP tanıtılarak kullanılacaktır. Fakat ilgi sadece AutoLISP'in kuralları ve yetenekleri üzerine yoğunlaştırılmayacaktır. AutoLISP grafik programlamanın basit prensiplerini desteklemek amacıyla kullanılacaktır.

Buradaki amaç, şu anda normal bir plan çizer gibi çizilen cephelerin biraz daha otomatikleştirilerek, planı çizilmiş bir projenin otomatik cephesinin ve alternatiflerinin daha kolayca çizilmesini sağlamak ve otomatik cephe çizimine duyulan ihtiyacı gidermektir. Her şeyi ile tam olarak bitmiş bir cephe, düşünülen şekli ile tam olarak çizdirilemese de yine de parametrik olarak değerlerin kullanıcı tarafından verilmesi ve diğer bilgilerin de plandan alınması ile otomatik bir şekilde kısmi çizim yapılabilecektir.

BÖLÜM 2. BİLGİSAYAR SİSTEMİ

2.1. BİLGİSAYAR SİSTEMİNİN YAPISI

Bilgisayar Destekli Mimari tasarım, bugün teknolojik ve ekonomik olarak her büro için gerçekleştirilebilirlik düzeyine ulaşmış durumdadır. 1960'lı yılların başlarında, bu alandaki teori ve pratiklerin gelişimi öyle bir noktaya gelmiştir ki, 1980'li yıllar boyunca BDT tekniklerinin günlük kullanımı sayesinde mimarlık pratiği kökten değişime uğramaya başlamıştır. Birçok faktör bu gelişime yardımcı olmuştur. [30].

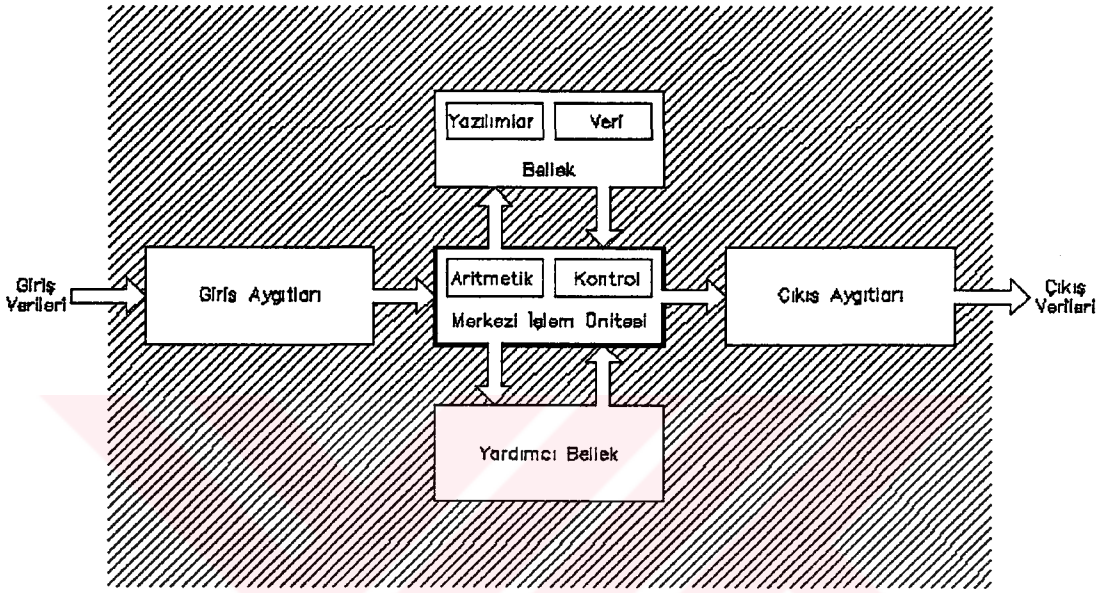
2.2. BİLGİSAYAR SİSTEMİNİN ELEMANLARI

Bilgisayarlar, iç belleklerinde bilgileri depolayan ve istenilen sonuçları üretmek için bu depolanan bilgilerden program olarak bilinen komut dizilerini kullanarak işletme işini yapabilme kabiliyeti olan elektronik aletlerdir. Bir bilgisayar sisteminin en esaslı elemanları donanım (Hardware) olarak bilinen araç grupları ve hesaplama yapan kısımları merkezi işlemcileri, işlenmek üzere bulunan veriler (Data) ve yazılım (Software) olarak bilinen programlardır. [30].

2.2.1. DONANIM (HARDWARE)

Tipik bir bilgisayar donanımı (Şekil 2.1) de gösterilmiştir. Sisteme bilgi girişi Giriş aygıtları denilen aletlerle yapılır. Bunlar delikli kart okuyucu veya elektrikli yazı makinası klavyesi olabilir. Giriş aygıtları bilgileri elektrik dalgaları halinde kodlar. Kodlanan bu bilgiler daha sonra hafızada saklanabilir. Bütün sistem konrol bölümü ile aritmetik ve mantıksal ünite tarafından yönetilen bir merkezi işletim birimi (CPU) ile kontrol edilmektedir. Kontrol ünitesi

hafızada bulunan programdaki komutları alır, yorumlar ve işler. Bu komutlar girdi veya çıktı verilerine ihtiyaç duyabilir, bellekte verileri saklayıp geri alabilir veya aritmetik ve mantıksal işlemleri veriler üzerinde uygulayabilir. Matematiksel ve mantıksal işlemler aritmetik ve mantık üniteleri tarafından uygulanır. [30].



Şekil 2.1 Bilgisayar Donanımının Yapısı

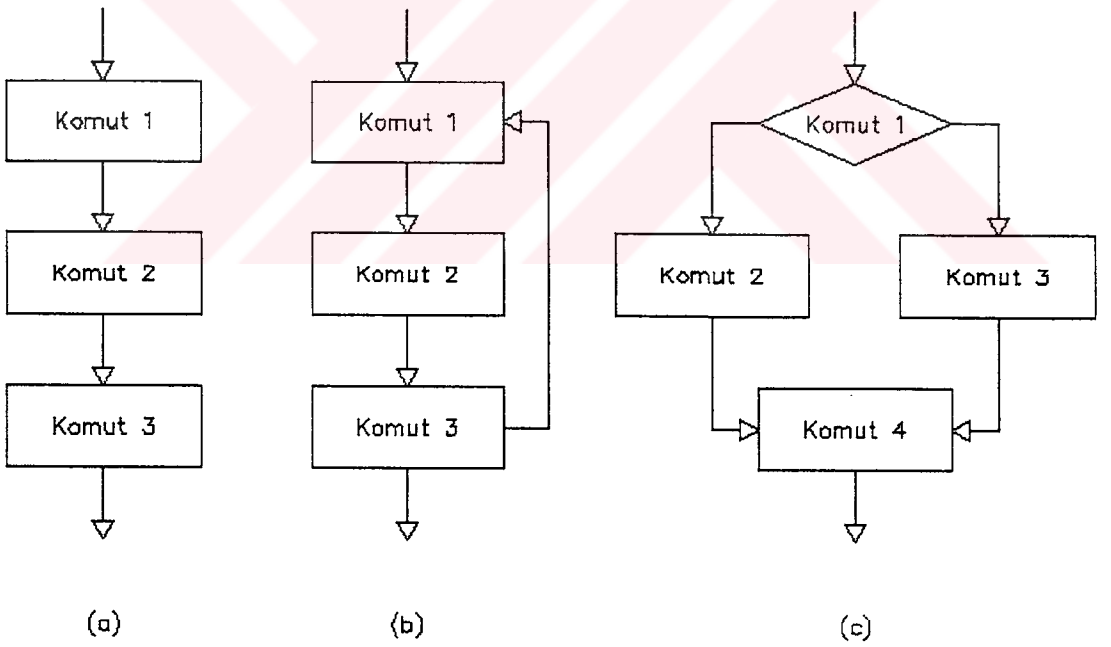
2.2.2. VERİ (DATA)

Herhangi türdeki bir veriyi elektrik sinyalleri haline dönüştüren her türlü araç; bir kart okuyucu, bir klavye, bir televizyon kamerası, bir mikrofon veya algılayıcı herhangi bir alet çok yaygın olarak bilgisayar girdi aygıtı olarak kullanılabilir. Benzer olarak elektrik sinyalleri ile kontrol edilen herhangi bir alet örneğin elektrikli yazı makinası, katod ışın tüplü ekran veya bir müzik sentez aleti de çıktı aygıtı olarak kullanılabilir. Çok değişik şekildeki veri tiplerini giriş için kodlayabilmesi ve çıkış sinyallerini de çok geniş cinsteki forma dönüştürebilmesi bilgisayarların çok esnek bir yapıya sahip olduklarını ve genel enformasyon işleyen aygıtlar olduklarını gösterir. Bunlar matematik makinaları, yazı işleme makinaları, grafik makinaları, konuşma ve dinleme makinaları, müzik makinaları, hareketli

otomatlar, hissetme, tadma veya koku alma makinaları ve başka makinaları kontrol eden makinalar olabilirler. [30].

2.2.3. PROGRAMLAR (PROGRAMS)

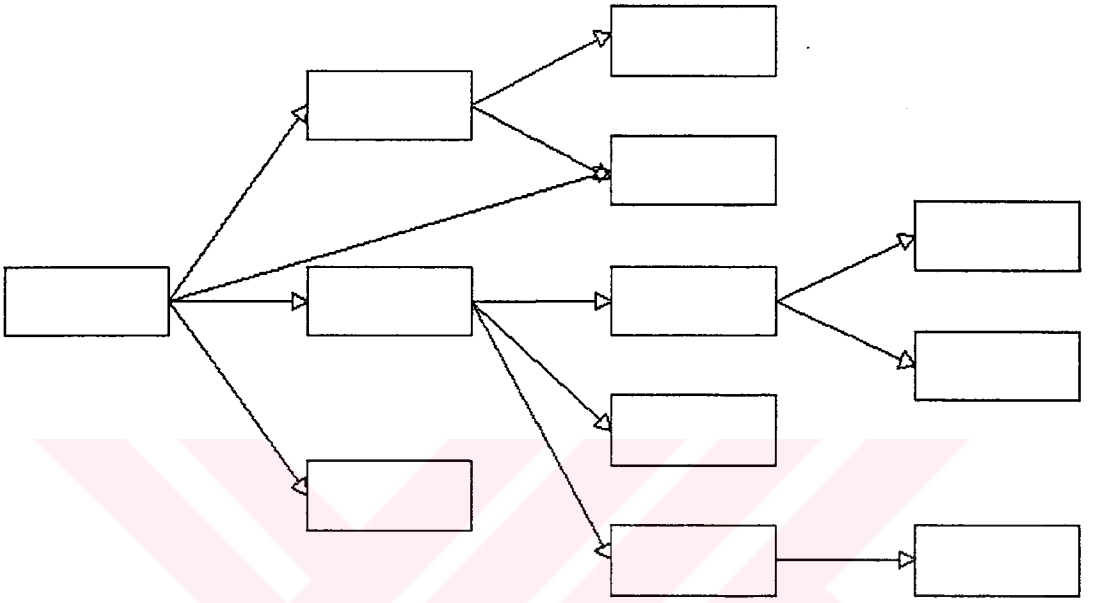
Merkezi işlem birimi tarafından veriler üzerinde işlemlerin yapılması program aracılığı ile olur ki, bu da veriler gibi hafızada saklanır. Bir program bilgisayar dili ile yazılmış bir dizi komutlardan oluşur. Basit bir programda bu komutlar oluşturulan sıra ile birer birer işlenerek çalıştırılır. (Şekil 2.2.a). Daha karmaşık programlarda döngü (Şekil 2.2.b) ve dallanmalar (Şekil 2.2.c) bulunur. Döngü, program içinde bulunan ve belirli sayıda tekrar edilen bir dizi komutlardan ibarettir. Dallanma komutu ise belirlenmiş operasyonlardan elde edilecek sonuçlara göre sonraki işlemin ne olacağını seçer. Döngü ve Dallanmalar uzun ve karmaşık komutların program içinde kısa ve daha anlaşılabilir olmasını sağlar.



Şekil 2.2.a,b,c Basit bir program, Döngü ve Dallanmalar

Bir program çok fazla uzun ise, genellikle alt rutinler ve fonksiyonlar olarak ayrı ayrı parçalar halinde yazılırlar. Bir alt rutin veya fonksiyon, bir komutla ana program içerisinden çağrıldığında çalışabilen, küçük program parçacıklarıdır. Alt rutin ve fonksiyonların

işlenmesi tamamlandıktan sonra ana program kaldığı yerden devam eder. Alt rutin ve fonksiyonların kendileri de bir başka alt rutin ve fonksiyonu çağırabilir. Bu tür tipik bir program yapısı (Şekil 2.3)'de gösterilmiştir. [30].



Şekil 2.3 Tipik bir program yapısı

2.3. TEKNOLOJİK GELİŞMENİN BİLGİSAYAR DESTEKLİ MİMARİ TASARIMIN YAPILABİLİRLİĞİ ÜZERİNDEKİ ETKİSİ

Bilgisayar Destekli Mimari Tasarımda yapılabilirlik çalışmalarının bilgisayar sistemlerindeki performansın artırılıp fiyatların düşürülmesinde 3 etkisi vardır. İlki, daha önce hiçbir şekilde yapılamayan yeni tip pratik işlemlerin hızındaki gelişmedir. İkincisi, hafıza kapasitesinin artması büyük miktarda veri saklanmasına ve işletilmesine olanak vermiştir. Bu iki neden çok önemlidir. Bilgisayar Destekli Mimari Tasarım teknikleri son yıllara kadar mümkün olabilen bilgisayar teknolojisinin sınırlarını sürekli olarak zorlamıştır. Üçüncüsü, uygulama ekonomisinin değişiminin tamamlanmasıdır. [30].

BÖLÜM 3. BİLGİSAYARLA BİNA CEPHELERİNİN TASARIMI KONUSUNDA LİTERATÜR ÇALIŞMALARI

3.1. BİNA CEPHELERİNİN TANIMLANMASINDA BİLGİSAYAR GRAFİKLERİ

BDT'deki bilgisayar grafiklerinin rolü tartışmalı bir konudur. Bazı insanlar, bilgisayar grafiklerini ve BDT'i eşanlamlı düşünürken, bazıları bilgisayar grafiklerinin tasarım için kullanılabilmesi için ileri ve geniş araştırmaların yapılmasının gerekli olduğunu belirtirler. Diğerleri, en basit bilgisayar grafik teknolojisinin dahi tasarım maliyetlerinde büyük tasarruflar sağladığını iddia etmektedirler. [18]. BDT sistemleri kendi içlerinde çok iyi organize edilmişlerdir. BDT sistemleri kullanıcı ve diğer programlarla ilişki halindedir. Bu sistemler kullanıcıyla iletişim kurmak üzere bir çeşit yönetim diline sahiptirler ve bu sistemler için uygulama programları yazmak mümkündür. BDT sistemleri kendi mimarilerine göre bir dosyalama sistemine ve uluslararası standartlara dayanan bir yapıya sahiptirler. Bu sistemler, mimarlar tarafından ortak kullanılan bir bilgi yapısına bağımlıdır. Bir sistemden diğerine bilgi transferi ve bu veri yapılarını kullanmak üzere uygulamaların eklenmesi olanağı vardır.

3.1.1 SÖZ DİZİMİ YAPILAR

Bilgisayar yardımı ile mimari kompozisyonun eğitimi konusunda çalışmalar vardır. Elemanlarının kurallarını grameri ile belirtildiği bir sözcük tarafından karakterize edilmiş bir mimari diller dizisi mekanda yerleştirilebilir. [16]. Mimari dilleri Fleming "Duvar mimarisi" şeklinde tanıtarak başlar. Sözlükteki bu eleman en basit eleman yüksekliğine ve uzunluğuna kıyasla derinliği az olan bir duvardır. Uygulamalı mevcut dilde duvarların birbirleri ile nasıl ilişkilendiğini, sezgi sonucu basit belirli kurallarla tanımlanır. Bu kurallarla örüntüler geliştirilir, değişik duvarlar yaratılır ve sonuç olarak mimarca binaların katalogları geliştirilebilir. Fleming kütle mimarlığı ve ondan oyulup çıkarılan bina kütlelerinden de söz

eder. İkinci tip duvarlar, duvarları bağımsız elemanlar olarak dikkate alan bir panel mimarisini oluşturur. Üçüncü tip duvar dizin dilinde kolonlar, kirişler gibi çerçeveleri oluşturan temel elemanlardan söz edilmektedir. Bu dilde mekanları çevreleyen dolgu elemanları ve bölmeler de dil içine alınmıştır.

3.1.2 CEPHE ÇALIŞMALARI

Mitchell'in [23] çalışmasında belirttiği gibi Pascal bilgisayar diliyle hazırlanan 3 basit tekrarlama modu geliştirmiştir. Çeşitli şekillerde tekrarlanabilir grafik kompozisyonları ve bunların döngü kullanılarak nasıl yapılabilecekleri açıklanmıştır. Mitchell, bir döngü içindeki sözlük elemanı parametresinin tekil pozisyonunun artırılmasıyla oluşan kompozisyonları incelemekle işe başlamıştır. Sonra iki ve üç pozisyon parametrelerini artırmayı düşünmüştür. Daha sonra ise bir sözlük elemanının tekil şekil parametresini artırmaya çalışmıştır. Sonuçta bir döngü içinde bir sözlük elemanın birden fazla parametresinin artışını incelemiş ve hepsini biraraya koymuştur. Çeşitli programlar geliştirerek nasıl tekrarlanan grafik ve mimari kompozisyonların yapılacağını açıklamaktadır. Bir duvardaki normal pencere açıklıkları, bir binayı tanımlayan zemin panel yükseklik parametreleri, duvar panelleri, merdivenlerin kesit çizimleri, kolon sıraları, basamaklı piramitler, friz kompozisyonsuzlukları, çatı makasları, çok katlı binaların kesitleri, parametrik programlar uygulanarak Pascal'la programlanmıştır.

3.1.3. BİÇİM TANIMA

Nagakura, bir prototip "shape-scripting" dili formundaki bir bilgisayar programını uygulamıştır. Bu program biçimlerin kategorilerini tanımlar. Bu, birbirleriyle ilintili parametrik kısıtlamalarla "successive" çizgilerin seti olarak tanımladığı biçimler kategorisiyle belirttiği "object-script" olarak adlandırdığı basit yapıları kullanır. Ortaya çıkan alt biçimlerin tanınmasını mümkün kılan bir mekanizma kullanır. Çizimlerdeki, tanımlanmış biçim kategorisinin üyelerini bulmak için biçim eşleşmesi kullanılır. Onun lisanı, biçim kategorilerinin kodlanması ve onların transformasyonları için bir hücre olarak çalışan, ve çizgi çizimlerinde biçim transformasyonu ve gelişen biçim tanımlanmasını sağlar. Mimarlar, bir aşamadan diğerine dizaynlarını geliştirmek için büyük miktarda bilgi kullanırlar. [25].

3.1.4. GÖRÜNTÜ KİTAPLIĞI

De Cola [12] Miller [32] ve Van Bakergem [31]'in belirttiği gibi çoğu zaman, eski şehirlerin ve mevcut binaların görünüşleri bilgisayarlarda saklanıp çeşitli nedenlerle sonradan kullanılmaktadır. Onlara göre kişisel hafıza, hayal gücünün tasarımda kullanılmasında önemli bir rol oynar. Fakat buna güvenilmez ve kontrolsüz olduğu söylenebilir. İkisi de tasarım üzerinde düşünce ve çizime yardımcı olmak üzere görüntü kitaplığı yapmışlardır. Böylece görüntülerin yüklenmesinde, değiştirilmesinde, çağrılmasında, yapılmasında elektronik yöntemleri kullanabilme kapasitesi, tasarımcıların, daha önce mümkün olmayan geniş görüntü kitaplıklarını etkili kullanması mümkün olmuştur. Üç kitaplık çeşidi tanımlanmıştır. [31]. Bunlar Genel, Toplumsal ve Kişisel'dir

Üç boyutlu geometrik modellerin kitaplıklarının önem kazanacağı görülmektedir. Bunlar Network (ki bu fazla önemli değildir) aracılığı ile Server'den CD ROM'a yazılabilir. Bir mimarın ihtiyaç duyabileceği modeller şunları kapsar: Şehir ortamlarının genel modelleri, önemli binaların benzer modelleri ve inşaat bileşenlerinin ve alt sistemlerinin kitaplıklarıdır [33]. MIT'de Miller [32] ve dünyadaki birçok mimarlık okulunda stüdyo eğitimi için bu görüntü kitaplıkları bugün mimari ölçekli modellerin yerini almıştır.

3.1.5 ÜÇ BOYUTLU GRAFİKLER

Öğrencilerin eğitimi, onların uzaydaki nesnelerin 2 boyuttaki temsili ile karıştırma yönündeki eğilimlerini azaltarak, binayı 3 boyutta düşünmelerini sağlamak amacına yönelmektedir [10]., [13], [17], [32]. Öğrenciler bilgisayarları 3 boyutlu çizimler için kullanmayı tercih ederler. Özellikle, öğrenciler 3 boyutlu uzay konusuna meraklıdırlar. 3 boyutlu grafikler ve çizimler öğrencilerin mimarlık maharetlerinin gelişmesine yardımcı olur. Binaların peyzajını görmek, öğrencilerin tasarımı daha fazla derinlikle incelenmesine fırsat vermektedir. Öğrenciler bu fikirleri modelleştirip deneyebilirler. Tasarım stüdyolarında bilgisayar kullanımı ile geliştirilimiş bazı problemler vardır. Ağırlıklı olarak Bilgisayar Destekli Tasarım eğitimi verme geleneği olmayan bir çok meslek okulunda bu tip problemlerle karşılaşmıştır.

Bilgisayar ortamında 3 boyutlu mevcut modellerin kullanımı ve bu modellerin herzaman öğrencilerin mimari tasarım stüdyolarında zihinlerinde benzer görüntüler oluşmalarına yararlı olmuştur.

3.2. BİLGİ-TABANLI BİNA CEPHELERİNİ BİÇİMLENDİRME ÇALIŞMALARI

Birçok mühendislik uygulaması bilgi-tabanlı sistemlerin sentezinin oluşturulması ile ilgilidir [28]. Örnek-tabanlı ve kural-tabanlı, mimarlık ve makina mühendisliği alanlarındaki çalışmalar öyle veya böyle biçimlerin sentezin üzerinde yoğunlaşmaya başlamıştır. Kural-tabanlı algoritmalar biçim özelliğinin karakteristiği olan bazı önceden belirlenmiş kurallara dayanarak biçim özelliği tanımlarlar. Kural tabanlı yaklaşımların dış avantajları şunlardır: a) kurallar biçim özellikleri ile doğrudan ilişkili değildir, b) algılanabilen her biçim özelliği için kurallar kullanılamaz, c) tanıma katı modelin tekrarlanan çok sayıda yorucu arama sürecini içerir.

BDT prensipleri tasarım otomasyonunun ihtiyaçlarını hedef alır. Otomasyon derecesi, tasarım metodolojisi konusundaki mevcut bilgi ile doğrudan ilgilidir ve yazılımın geneli kullanımının ters fonksiyonudur. Bu nedenle bir işlemin otomasyonu, metodolojinin tümü ile anlaşılmasına bağlıdır.

3.2.1. TANIMLAYICI TASARIM

Myers [24] akıllı tanımlayıcı tasarım otomasyon programı geliştirmiştir. Bu araştırma nesneye yönelik tanımlayıcı tasarım programlaması kullanarak "deniz platformu" konfigürasyonu için bir prototip geliştirilmesiyle ilgilidir. Bu prototipin amacı, ilk eleman büyüklükleri ile başlangıç konfigürasyonu geliştirmek ve bunları daha sonra var olan yapısal sistemleri kullanarak analiz etmektir. Her dış veri, diğer bölgelerle ilişkiye geçmeden önce, zemin sistemi ile ilişkiye geçmesi dolayısıyla mantıksal olarak tasarımın başlangıç noktası olarak düşünülmüştür. İlk başlangıç tasarımın tamamlanmasıyla, ile uygun sayıdaki elemanın modelinin geliştirilmesi gerçekleştirilebilir. Bu aşamayı takiben malzeme değişim modeli başarılabilir. Bu çeşit nesne-yönelimli tanımlayıcı tasarım programlaması, büyük

miktarda bilginin işlenmesine olanak vermekte ve tekrarlanan işlemleri yapabilmekte ve mühendisin çeşitli platform tasarımları yapmasına olanak vermektedir..

3.3. BİNA PLANLARI İÇİN ŞEKİL TANIMA ÇALIŞMALARININ YARARLARI

Şekil tanıma bilimsel ve teknolojik bilgisayar uygulama programlamasında önem kazanan bir konudur. Görüntülerden oluşan fiziksel nesnelerin anlamlı tanımlanmasıdır. Şekil tanıma'nın uygulandığı birçok alan vardır. Bunlar ; robot görme sistemleri, büyük resim koleksiyonlarının arşivlenmesi veya ekranda gösterilmesi, parmak izi alınması, bulut ve hava fotoğraflarının alınması, karakter tanınması, askeri hedeflerin keşfedilmesi, tomografik tarama gibi alanlardır. Şekil tanıma'da işlenmemiş, algılanmış bilgi genelde ayrıştırılmadan önce ön bir işleme tabi tutulur. Tüm "pixel" değerleri saklamak mümkündür. Tüm "pixel" 'ler işlendiğinde hesaplama için uzun süre ve özel amaçlı teknik donanıma ihtiyaç vardır. Görüntünün içeriğini sunabilmek için, bu "pixel" 'lerin kullanımı gereklidir. Böylece görüntü sahasındaki nesnelerin daha basit bir tanımı verilebilir. Bu işlemi çalıştırabilmek için genel metotlar yoktur. Fakat, değişik problemlerde kullanılmak üzere bazı teknikler geliştirilmiştir. Aşağıda sadece ilgili teknikler incelenmiştir.

3.3.1 KENAR KEŞİF (EDGE DETECTION) TEKNİĞİ

Kenar keşifi bölgesel bir kenarın bulunması, eşik (tresholding), kenar keşif işlemcileri, zincir kodlaması, çevre hesaplamaları, alan hesaplamaları, nesne genişliği hesapları, değişik problemlerde kullanılan tekniklerdir. Şekil tanıma'da belirli sistemler belirli uygulamalar için kullanılır.

3.3.2 SINIR İZLEME (CONTOUR FOLLOWING) TEKNİĞİ

Sınır izleme tekniği ile bir bölgedeki veya onun dışındaki bir test noktasının yerinin belirlenmesinde kullanılabilir. Başlangıçta bir kenar noktası aranır. Eğer mevcut nokta bölgenin içinde ise test "pixel" 'leri de bölgenin içindedir. Böylece arama işlemi bölge bitene

kadar sol taraftan devam eder. Eğer mevcut nokta bölgenin dışında ise arama işlemi bölgeye girene kadar sağ taraftan devam eder.

3.3.3 FORM TANIMA (FORM RECOGNITION) TEKNİĞİ

Biçim özelliği üzerine yapılan çalışmalar iki ana kategoride sınıflandırılabilir.

a) İnsan destekli özellik tanıma
b) Otomatik özellik tanıma

Birçok uygulamada insan destekli tanıma sistemleri kullanılır. Bu sistemlerde kullanıcı bir bilgisayar ekranından bir bölümün tel çerçeve görüntüsünden nesneler seçer. [9], [8], [19], [26]. Biçim özelliği tanıma için sinir-ağları (Neural networks) ilkelerinden hareket ederek yeni bir teknik geliştirilmiştir. Prabhakar ve Henderson [27] tarafından paralel olarak parçaların katı modellerini veri kabul eden bir sinir-ağı geliştirmişlerdir. Bu çalışmada cephenin cepheyle ilişkisine dayanan cephe tanımları ortaya koyan bir format geliştirilmiştir. Katı modellerin çoğu sıralanan üç yaklaşımdan birisine girer:

- kompozisyon modelleri
- yapım modelleri
- sınır tanımlama modelleri

Bu çalışma katı modellerin sınır tanımlamalarıyla ilgilidir. Bu çalışmada bir nesne önce katı model olarak tanımlanır. Sonra sinir ağına dayanan tekniği kullanan bir tanıma işlemi uygulanan algoritma başlar.

BÖLÜM 4. ÇALIŞMANIN METODU

Bu çalışma, bir planın kenarlarını tanıma ve tanınan kenar çizgilerine dayanarak kullanıcının belirleyeceği parametrelere göre binanın dört cephesinin çizdirilmesi için yapılmıştır. Bu çalışmada aşağıda sıralanan kabuller yapılmış ve program bu kabullere dayanarak geliştirilmiştir.

1) Cephede pencerelerin boşlukları ve doğrama kalınlıkları dışında bir süsleme olmadığı
2) Çizdirilecek cephenin 1/50, 1/100 veya 1/200 çizim ölçeğinde eskiz olduğu
3) Cephe kaplama malzemelerinin bu aşamada gösterilmemesi
4) Çatının düz çatı olarak kabul edilmesi
5) Pencere bölmelerinin şu aşamada yalnız düşey olması
6) Ekranın üst kısmının Kuzey cephesi olduğu

Bu kabullere göre duvar çizgileri ve pencere boşluları çizilecek ve pencere boşluklarının içine pencerelerin çizimi yaptırılacaktır.

4.1. METOT

Planı AutoCAD ortamında çizilmiş olan bir binanın dört cephesinin çizdirilmesi için bir çalışma yapılmaktadır. Bu yapılacak çalışmada:

Planı verilen bir binanın kullanıcının belirleyeceği cephe özelliklerine göre cephelerin otomatik olarak çizdirilmesine çalışılmaktadır. Programda cephenin çizdirilmesi için yaptırılması düşünülen işler düşey izdüşümlerin plandan alınması ve buna kullanıcı tarafından girilecek yatay hatlara gerekli bilgilerin eklenmesi ile cephenin çizdirilmesidir.

Burada kullanıcı tarafından girilecek değerlere:

- Cephe bilgileri
- Çizdirilecek katın adı
- Çatı durumu
- Ölçeği
- Kat adedi
- Kat yükseklikleri
- Parapet yükseklikleri
- Pencere yükseklikleri
- Normal kat alt kotu
- Çizdirilecek cephenin yönü
- Pencere tipi
- Doğrama kalınlığı
- Pencere bölme sayısı
- Pencere bölme aralıklarının durumu
- Pencere aralıkları farklı ise ölçüleri
- Açılan kanatlar
- Çatı parapet yüksekliği
- Saçak genişliği
- Çatı parapeti alt kotu

plandan alınan:

- Bina dış kontur hizaları
- Pencere ve kapı düşey izdüşüm yerleri ve genişlikleri

gibi değerlerin eklenmesiyle pencere boşlukları ve dış duvar çizgileri çizdirilecektir.

Programın başında herhangi bir hata oluştuğunda yapması gereken işlemler tanımlanmaktadır. Buna göre yapılacak işler "Hatalı bilgi" mesajı verdikten sonra tüm değişkenlerin sahip oldukları değerler boşaltılıp programın durdurulması şeklinde olacaktır. Bu işlem için bir alt fonksiyon **"PEN_ERR"** tanımlanarak gereken durumlarda bu

fonksiyonun ana program içerisinden çalışması sağlanmıştır. Sonraki aşamada programın ana bölümlerinden biri olan dialog kutusunun çalışması için gerekli olan alt program **"PEN_DIA"** çaılştırılacaktır. Dialog kutusunda programın ihtiyaç duyduğu ve kullanıcı tarafından verilmesi gerekli tüm bilgiler kullanıcıya sunulmaktadır. Kullanıcıdan alınacak bilgiler şu şekildedir.

a) Çizdirilecek cepheye ve projeye ait yazdırılması istenen yazı bilgileri

Projenin adı

Cephenin adı

Yazdırılacak yazının ayar bilgileri

Yazının stili

Yazının yüksekliği

Yazının açısı

b) Çizdirilmesi istenen katın adı

Zemin kat, Normal kat veya Çatı katı (Sadece bir tanesi seçilebilir.)

c) Çizdirilecek kat ile birlikte çatının da çizilip çizilmeyeceği

d) Çizdirilmesi düşünülen detay için gerekli ölçek

1/50, 1/100 veya 1/200 (Sadece bir tanesi seçilebilir.)

e) Çizdirilecek kata ait bilgiler

Kat adedi (Normal kat cephesi çizilecekse değer verilebilir.)

Kat yüksekliği

Parapet yüksekliği

Pencere yüksekliği

1. Normal katın alt kotu (Normal kat cephesi çizilecekse değer verilebilir.)

f) Çizdirilecek cephenin adı

Kuzey Cephesi, Güney Cephesi, Doğu Cephesi veya Batı Cephesi (Sadece bir tanesi seçilebilir.)

g) Cephede bulunan pencerelere ait bilgiler

Pencere tipi (Geliştirmeye yönelik olarak ilerisi için düşünülmüştür.)

Pencerelerin düşeydeki bölme sayısı

Pencerelerin bölme aralıkları; Eşit veya Farklı olabilir.

Açılan kanatların numaraları (soldan sağa doğru)

h) Çatı bilgileri

Çatı parapet yüksekliği

Saçak genişliği

Çatı parapeti alt kotu

"PEN_DIA"

Program çalıştığında ekranda bir dialog kutusu belirecek ve kullanıcıdan yukarıda söz edilen değişkenlere ait bilgilerin girilmesi istenecektir. Eğer kullanıcı hiçbir değişkene değer vermez ise program ön kabul değerlerini kullanarak cepheyi çizecektir. Ön kabul değerleri şu şekilde verilmiştir.

Kat adı	: Zemin kat
Çatı durumu	: Yok
Ölçek	: 1/50
Kat adedi	: 1 (Kat adı "Zemin kat" olduğu için)
Kat yüksekliği	: 270
Parapet yüksekliği	: 90
Pencere yüksekliği	: 120
1. Normal kat alt kotu	: 270 (sadece Kat adının "Normal kat" olması durumunda kullanılacaktır.)
Cephe adı	: Ön Cephe
Pencere tipi	: TIP1 (ileriki aşama için düşünülmüştür.)
Bölme sayısı	: 2
Bölme aralıkları	: Eşit
Açılan kanatlar	: Yok
Çatı parapet yüksekliği	: Pencere parapet yüksekliğine eşitlenmiştir.
Saçak genişliği	: 50
Çatı parapet alt kotu	: (Programın çalışması esnasında belirlenmektedir.)

Alınan bu bilgilere göre program işlemeye başlayacak ve adım adım yaptırılan işlemlerden sonra cephe çizimi yapılacaktır.

Dialog kutuları yardımı ile yukarıda sözedilen bilgilerin alınmasından sonra, plandan duvar yerlerinin ve pencere boşluklarının tesbiti için gerekli olan işlemler yapılacaktır. Bunun için yeni bir alt program **"SEC"** tanımlanmıştır.

"SEC"

Planda bulunan tüm duvar çizgileri AutoCAD programının Layer ve nesne tipi özelliklerinden faydalınalarak şeçtirilmiştir. Bunun için Layer adı "DUVAR" ve nesne tipi "LINE" olan elemanlar seçtirilerek 1 numaralı seçim seti (sec_duv) oluşturulmuştur. Bu aşamada tüm duvar çizgileri seçildiği için bir eleme yaptırılıp, çizilmesi istenen cephe adına göre yeni bir seçim seti oluşturulmalıdır. Buna göre çizdirilecek cephenin adı "Güney Cephesi" veya "Kuzey Cephesi" ise bir önceki seçim setinden, sadece yatay olan çizgiler, "Doğu Cephesi" veya "Batı Cephesi" ise sadece düşey olan çizgiler alınacaktır. Yeni oluşan 2 numaralı seçim seti (sec_duv2) ile birlikte bu sette bulunan çizgilerin başlangıç ve bitiş nokta koordinatlarının saklandığı 1 numaralı liste (lis_duv1) oluşturulmuştur. Benzer olarak bu listede de saklanan koordinatlar, çizdirilecek cephe adına göre belirlenmiştir. Bu aşamadaki amaç, seçilen yatay veya düşey çizgilerden oluşturulan seçim setinden, çizdirilmesi istenilen cephe adına göre bir eleme daha yaptırılarak, çizdirilecek cepheye ait olan duvar çizgilerinin bulunduğu 3 numaralı seçim setini (sec_duv3) oluşturmaktır. Sözedilen liste için, elde bulunan 2 numaralı seçim seti (sec_duv2) nesnelerin başlangıç ve bitiş koordinatlarının saklanması gerekecektir. Yukarıda da sözedildiği gibi bu işlem çizdirilecek cephe adına göre olacaktır. Eğer çizdirilecek cephenin adı "Güney Cephesi" veya "Kuzey Cephesi" ise seçim setinde bulunan nesnelerin başlangıç ve bitiş noktalarının "y" koordinatları, "Doğu Cephesi" veya "Batı Cephesi" ise "x" koordinatları listeye ilave edilecektir. Bu işlem 2 numaralı seçim setine ilave edilecek her nesne için yapılacaktır. **"SEC"** isimli bu alt programın işlemesi bittikten sonra ana programa dönülerek **"SEC2"** adındaki bir başka alt program çalıştırılacaktır.

"SEC2"

"SEC" alt programında adı geçen 3 numaralı seçim seti bu aşamada oluşturulmaktadır. Bunun için 1 numaralı listede bulunan elemanlar içinde değeri en büyük ve en küçük olanlar belirlenerek birer değişkene atanacaktır. 3 numaralı seçim setine dahil edilecek olan nesnelerin kontrolu, çizilecek cephe adına göre listeden elde edilen en büyük (max1) veya

en küçük (min1) değere göre yapılacaktır. Kontrol işlemi çizilecek cephenin adı "Güney Cephesi" veya "Batı Cephesi" ise en küçük değere, "Kuzey Cephesi" veya "Doğu Cephesi" ise en büyük değere göre yapılacak ve kontrolden geçebilen nesnelerin oluşturacağı seçim seti 3 numaralı set (sec_duv3) olacaktır. Burada yapılan kontrol işlemi şu şekilde çalışmaktadır: Yukarıda açıklandığı gibi 1 numaralı listeden alınan en büyük ve en küçük değerler (max1) ve (min1) adındaki değişkenlere atanmıştır. 2 numaralı seçim setinde bulunan her elemandan çizilmesi istenen cephe adı "Güney Cephesi" veya "Batı Cephesi" ise başlangıç ve bitiş koordinatı (min1)'e eşit olanlar, "Kuzey Cephesi" veya "Doğu Cephesi" ise (max1)'e eşit olanlar elemeden geçecek ve bunlar 3 numaralı seçim setini (sec_duv3) oluşturacaktır. Bu seti oluşturan elemanların başlangıç ve bitiş noktalarının koordinatları da (lis_duv2) adındaki listeyi oluşturacaktır. Elemeden geçemeyen nesneler ise 2.1 numaralı seçim seti (sec_duv21) diye adlandıracağımız seti, bu nesnelere ait başlangıç ve bitiş nokta koordinatları da tekrar (lis_duv1) adındaki listeyi oluşturmuşlardır. Elemeden geçemeyenler de bir seçim setinde toplanmıştır, çünkü, 3 numaralı seçim seti ile elde edilen nesneler cephenin en önündeki duvar çizgilerine ait olacaktır. Cephede geriye çekilmelerin de olabileceği düşünülerek, 2.1 numaralı set oluşturulmuş ve bu sayede geriye çekilmeler de çizdirilebilmiştir. 2 numaralı seçim setinde bulunan tüm nesneler kontrolden geçrilip 2.1 ve 3 numaralı set oluşturulduktan sonra 2 numaralı seçim seti iptal edilerek 2.1 numaralı sete ait tüm nesneler 2 numaralı sete aktarılacaktır.

Buradan sonra tekrar ana programa dönülerek çizilmesi istenen kat adı "Zemin kat" ise, **"ZEMIN"** adındaki alt programın çalışması sağlanacaktır.

"ZEMIN"

Bu alt programın çalıştırılmasının amacı, zemin kat cephesinde çizilmesi gerekli olacak kapıların olup olmadığını tesbit etmektir. Bunun için planda bulunan nesnelerden Layer adı "KAPI" ve nesne tipi "BLOCK" olanlar seçilerek bir set (sec_kap) oluşturulmuştur. İlk aşamada elde edilen sette iç ve dış kapıların tümü olacağı için bir eleme yapılarak sadece dış, ve çizilmesi istenen cepheye ait kapıların seçtirilmesi gerekecektir. "Güney Cephesi" ve "Kuzey Cephesi" için açısı "0" veya "180" derece, "Doğu Cephesi" ve "Batı Cephesi" için açısı "90" veya "270" derece olanlar ve kapının yerleştirildiği yerin koordinatı bir önceki aşamada elde edilen listede (lis_duv1) olanlar seçilerek (sec_kap1) adlı kapı seçim seti oluşturulmuştur.

Ana program aşamasında çizilecek katın adı "Zemin kat" değilse **"ZEMIN"** adlı alt program ihmal edilecektir. Sonraki aşamada (lis_duv2) adındaki listede eleman olup olmadığı kontrol edilmiştir. Eğer listede eleman var ise, bu çizilecek duvar ve pencere çizgileri olduğunu göstermektedir. Eleman var ise sırası ile **"SIRALA"**, **"MAXMIN2"**, **"CIZ"** ve **"MAXMINK"** adlı alt programlar çalışacaktır.

"SIRALA"

Bu alt program ile daha önceki alt programlardan **"SEC2"** ile elde edilen (lis_duv2) adlı listeye ait değerlerin sıralanması yaptırılacaktır. Sıralanan değerler en küçükten en büyüğe doğru olacak şekilde yapılacak ve yine (lis_duv2) adlı listede toplanacaktır.

"MAXMIN2"

Programın çalışması ile (lis_duv2)'ye ait değerlerden en büyüğü (max2), en küçüğü (min2) değişkenine atanacaktır.

"CIZ"

Çizilecek kat adı "Zemin kat" ise, bir altında herhangi bir kat olmadığı için katın alt kot değeri (kat_yuko) 0 (Sıfır) olarak atanmıştır. Çizilecek kat adı "Zemin kat" değil ise, katın alt kot değeri daha önceden diyalog kutuları aşamasında belirlenmişti. Duvar çizgilerinin alt kotu bu şekilde belirlendikten sonra, bu değere diyalog kutusundan alınan kat yüksekliği değeri ilave edilerek, duvarın üst kotu elde edilmiştir. **"SIRALAMA"** programı ile sıralanan listeden elde edilen en küçük değer (min2) sol dış duvar ve en büyük değer (max2) sağ dış duvar hizalarını göstermektedir. Bu iki değerin arasında kalanlar ise, pencere boşluklarının hizasını göstermektedir. Öncelikle sol dış duvar, elde edilen alt kot değeri ile sol dış duvar hizası ve üst kot değeri ile sol dış duvar hizası değerlerine göre, sağ dış duvar, elde edilen alt kot değeri ile sağ dış duvar hizası ve üst kot değeri ile sağ dış duvar hizası değerlerine göre kat adı "Çatı" dan farklı olmak şartı ile çizdirilmiştir. Bu seçim setinde bulunan nesnelere ait dış duvarlar çizildikten sonra, çizdirilen kat ile birlikte çatının çizdirilip çizdirilmeyeceği kontrol edilmiştir. Çizdirilmesi gerekiyor ise, **"CATI_CIZ"** alt programı

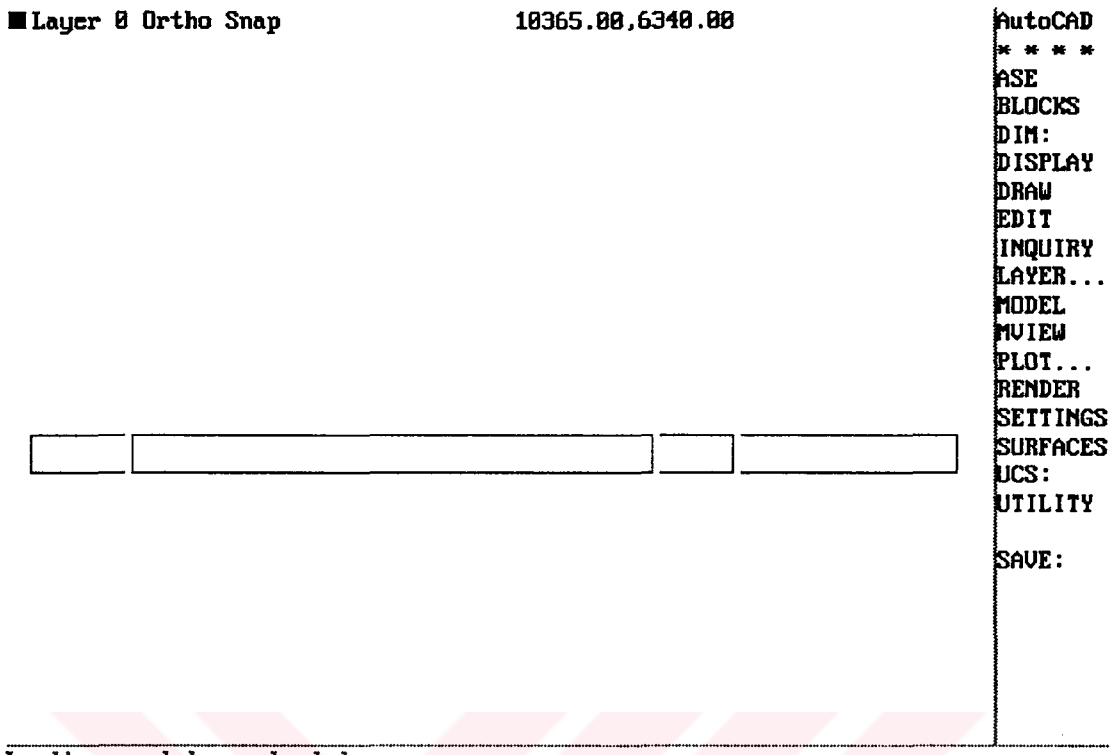
çalıştırılmış, gerekmiyor ise programa devam edilmiştir. Yukarıda da sözedildiği gibi, listede (lis_duv2) bulunan min2 ve max2 değerleri dışında kalanlar, pencere boşluklarının düşey izdüşüm koordinatları olarak kullanılacaktır. Düşey izdüşümleri bu şekilde alınan pencerelerin, yatay hizaları da dialog kutusundan alınan parapet yüksekliği ve pencere yüksekliği değerlerine göre elde edilecektir. Çizdirme aşamasına gelmeden önce, bu boşluklarda kapı olup olmadığının kontrolü yapılmıştır. Eğer boşluklara denk gelen düşey izdüşümlerinden herhangi birisi kapı listesinde (kap_lis2) bulunan değere eşit ve çizilecek kat adı "Zemin kat" ise, burada bir kapı olduğu anlamı çıkmaktadır ve bunun için kapının alt kotu duvar çizgilerinin alt kotu olarak belirlenerek boşluk parapetsiz olarak çizilecektir. Kapı yok ise, normal pencere boşluğu açılacaktır. Açılan boşluklara çizdirilmesi gereken kapı veya pencere kanatları da bir başka alt program ile yapılmaktadır. **"KAPI"** ve **"PEN"** adı ile çalıştırılan bu alt programların işleyişleri daha sonra açıklanacaktır. Çizdirme işleminin son aşamasında son çizilen kat adı "Zemin kat" ise, o kata ait üst kot bir değişkene (kat_yuko) atanarak çizdirilmesi düşünülebilecek üst katlar için alt kot belirlenmiş olacaktır.

"CATI_CIZ"

Çatı çizgilerinin çizdirilebilmesi için çatı alt kotunun belirlenmesi gerekmektedir. Çatı çizimi Normal kat çiziminden sonra yapılacaksa, çatı alt kotu Normal kat sayısı ve kat yüksekliklerinin çarpılması ile elde edilecektir. Zemin kat çiziminden sonra ise çatı alt kotu kat yüksekliğinin kendisi olacaktır. Her dış duvar çiziminden sonra çatı çizimi yapılacağından dış duvar hizalarına saçak genişlikleri ilave edilerek çatı düşey izdüşümleri belirlenmiştir. İlk çatı çizimi en öndeki çatı için olacağından kapalı bir dikdörtgen çizdirilmiştir. Cephe hareketlerindeki geriye çekilmelerden dolayı oluşacak çatı çekilmeleri ise geride kaldıkları hissini verebilmesi için önde kalan çatı hizasından kopartılmış ve bu şekilde çizdirilmiştir. (Şekil 4.1)

"KAPI"

Bu program ile kapı boşluğu içine çizilecek çizgi adedi belirlenmiştir. Bu işlem ölçeğe bağlı olarak yapılmıştır. Ölçek 1/50 ise 3 çizgi ile, 1/100 ise 2 çizgi ile gösterilmesi sağlanmıştır. Ölçek 1/200 ise tek çizgi olarak bırakılarak çizgi ilavesi yapılmamıştır.



Şekil 4.1 Çatı çizimi

"PEN"

Bu program ile pencere boşluklarının yeri tekrar tesbit edilmekte ve dialog kutusundan alınan pencere kanat sayılarına göre kanat genişliklerinin tesbiti bir başka alt program **"KAN_GEN"** ile yapılmaktadır.

"KAN_GEN"

Pencere bölme sayısına göre kanat genişlikleri bu aşamada belirlenmektedir. Bölme aralıkları eşit olduğu durumlarda, pencere genişliğinden doğrama kalınlıkları düşülerek elde edilen değer, pencere bölme sayısına bölünmüş ve bu sayede her bir kanadın genişliği bulunmuştur. Burada, her bir kanat eşit olduğu için, bir kanat çizilmiş, diğerleri ilk çizilen kanadın kopyalanması ile oluşturulmuştur. Eşit olmadığı durumda ise, o anda çizimi yapılan pencereye ait her bir kanadın genişliği kullanıcıya tek tek sorularak alınan cevaba göre kanatlar çizilmiştir. Her iki durumda da bu programın içerisinden çalıştırılan **"KAN_CIZ"** alt programı kullanılmıştır.

"KAN_CIZ"

Ölçek 1/200 ise bölmeler sadece tek çizgi olarak, 1/100 ve 1/50 olduğunda 2 çizgi olarak gösterilmiştir. 1/100 ve 1/50 ölçek için kanatlar pencere boşluklarından doğrama kalınlığı kadar geri çekilerek oluşturulmuştur. Diyalog kutularından alınan bilgilerde açılan kanat numaraları verildi ise, bu numaralara denk gelen kanatların içlerine ölçeğin sadece 1/50 olması durumunda birer çizgi daha ilave edilmiştir. Kanat numaraları verilmiş bile olsa, ölçek 1/100 veya 1/200 ise açılır kanatlar ihmal edilerek çizdirilmemiştir.

"MAXMINK"

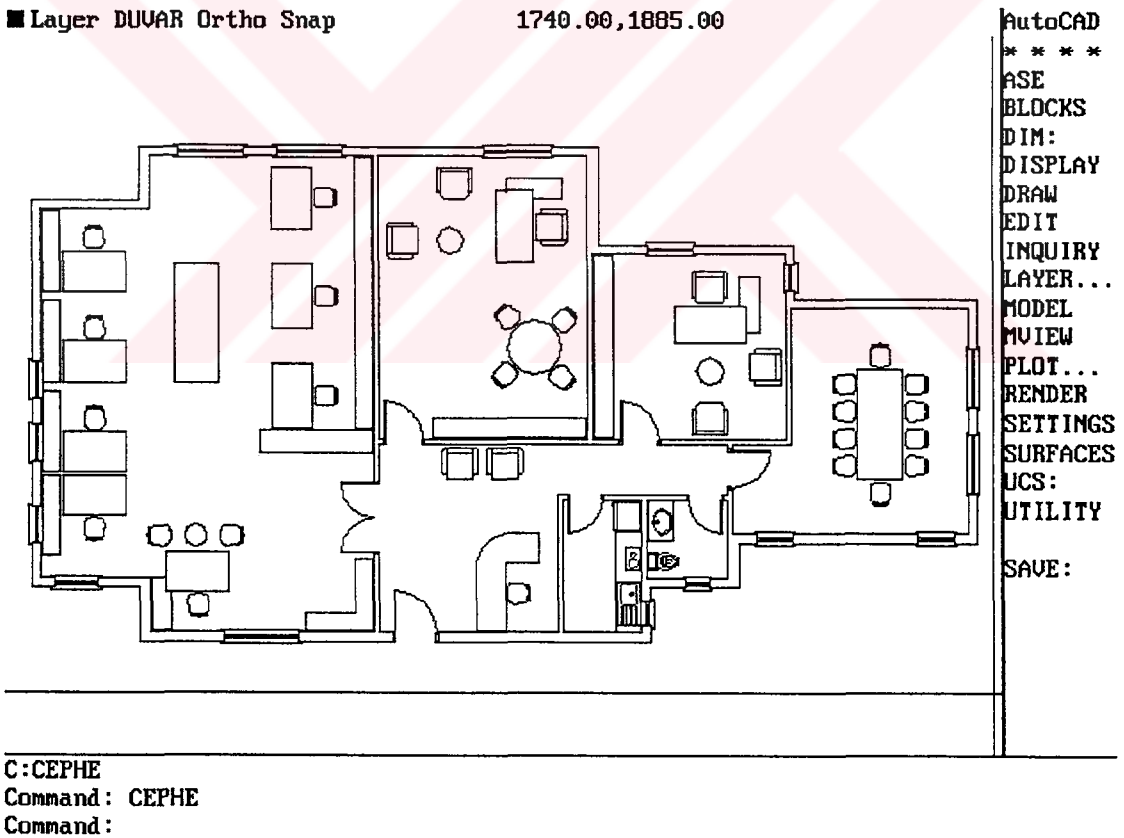
Sonraki işlemlerde kontrol amacı ile kullanılabilmesi için önceki aşamalarda elde edilen (max2) değişkeni (maxk)'ya, (min2) değişkeni de (mink)'ya atanmışlardır.

Daha önceden nedeniyle açıklanarak oluşturulan (lis_duv1) listesinde eleman olduğu sürece, önce yeni seçim setlerinin oluşturulması **"SEC2"** alt programı ile yapılmakta ve bu programa göre oluşturulan (lis_duv2) adlı listenin mevcudiyetine göre yukarıda da açıklanan **"SIRALA"**, **"MAXMIN2"**, **"CIZ"** ve **"MAXMINK"** alt programları (lis_duv1) listesinde hiç eleman kalmayana kadar çalıştırılmaktadır. Bu da çizdirilmesi düşünülen cepheye ait tüm duvar ve pencere çizgilerinin çizilmesini sağlayacaktır. Bu işlem (lis_duv1) listesinde hiç eleman kalmayıncaya kadar tekrarlanacaktır.

Ana programdaki bir sonraki adım zemin çizgisinin çizdirilmesidir. Eğer, çizdirilen cephe adı "Zemin kat" ise, bir de zemin çizgisi ilave edilecektir. Daha sonra çizdirilen cephenin "Kuzey Cephesi" veya "Batı Cephesi" olup olmadıkları kontrol edilmektedir. Eğer "Kuzey Cephesi" veya "Batı Cephesi" ise, bu cepheler ilk aşamada içeriden bakılmış gibi çizdirildiğinden simetriği alınarak, esas cephe resimleri oluşturulmuştur. Çizdirilen cephe adı "Normal kat" ise ve "Kat adedi" 1'den fazla ise, çizdirilen ilk Normal kat çizimi kat adedi kadar kopyalanarak cephe çizimi tamamlanmıştır. Son olarak projenin ve cephenin adları çizimin altına yazdırılarak program bitirilmiştir.

4.2. PROGRAMIN KULLANMA KILAVUZU

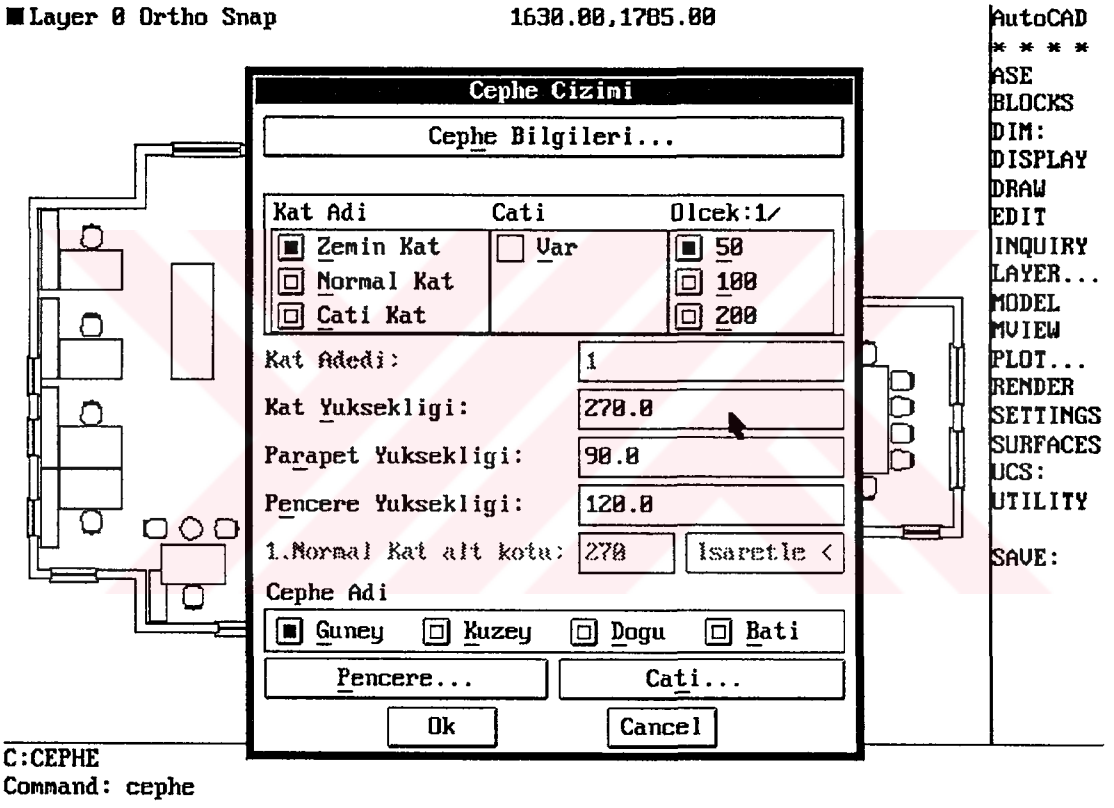
Program AutoCAD Release 12 için hazırlanıp, AutoCAD ile planı çizilen bir projenin otomatik cephe çiziminin yapılabilmesi amacı ile yazılmıştır. Programın yazımı Yapay zeka (Artificial Intelligence) işlemleri için geliştirilen LISP (List Processing) programlama dilinin içine AutoCAD komutlarının ilavesi ile geliştirilen AutoLISP programlama dili kullanılarak gerçekleştirilmiştir. Program kullanıcı ile AutoCAD'in imkanlarının birleştirilip AutoCAD ortamında planı çizilen projeden faydalanarak otomatik cephe çizimine duyulan ihtiyacı gidermek amacı ile yazılmıştır. Program, daha önceden çizilmiş plandan (Şekil 4.2) cephenin düşey hatlarının alınmasından ve diyalog kutuları yardımı ile kullanıcı tarafından da yatay hatların belirlenmesinden sonra bunların birleştirilip kapı ve pencere boşluklarının açılması esasına dayanmaktadır.



Şekil 4.2 Program ile 4 cephesi çizdirilen örnek plan

Programın kullanılabilmesi için AutoCAD Release 12 çalıştırılıp, cephesi çizilmesi istenen plan açılarak içine girilmesi gerekmektedir. AutoCAD içinde AutoLISP programlarının

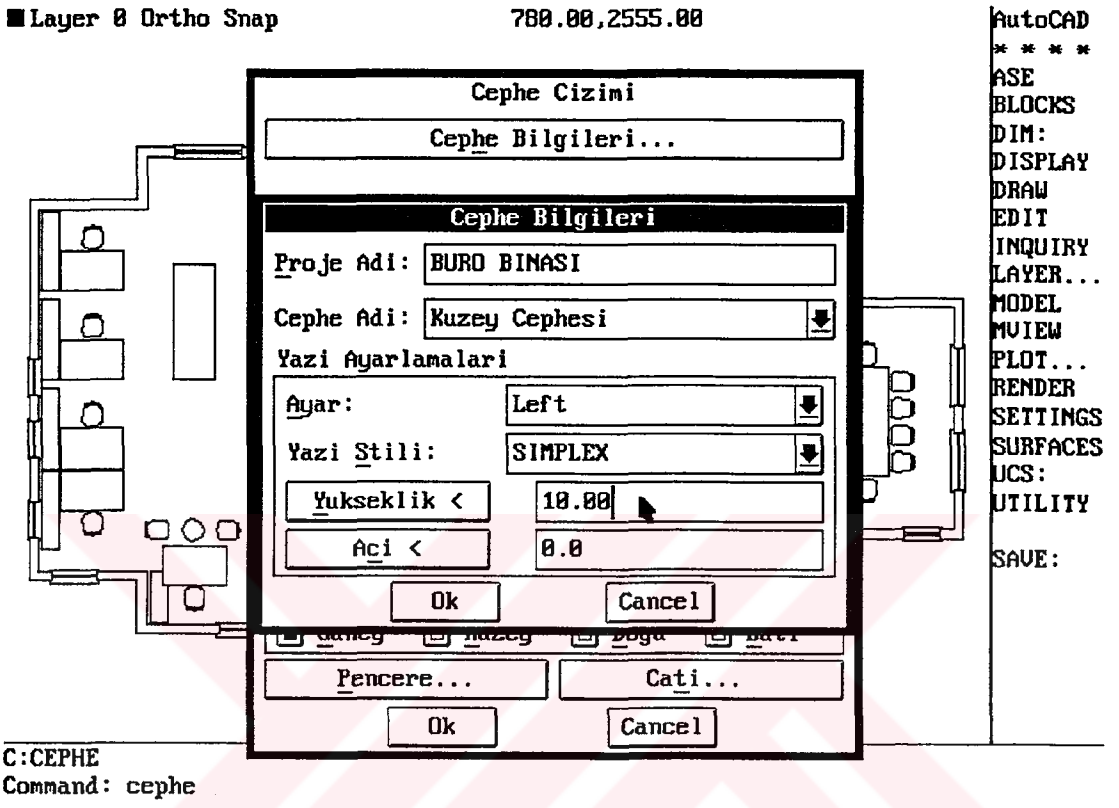
yüklenebilmesi için gerekli komut **"LOAD"** kullanılarak cephe çizimi için hazırlanan **"CEPHE.LSP"** yüklenir. Bunun için AutoCAD'de "Command:" durumunda iken ***(LOAD "CEPHE")*** yazılarak, program yüklenip kullanıma hazır duruma getirilir. Bunun sonucunda AutoCAD içine yeni bir komut ilavesi yapılmış olur ve normal AutoCAD komutu kullanır gibi kullanıcının kullanımına olanak verilmiş olur. Programın çalıştırılması için ***"CEPHE"*** yazılıp ***"ENTER"*** tuşuna basılarak çizim ekranına bir diyalog kutusunun (Şekil 4.3) gelmesi sağlanır. Buradaki diyalog kutusunda aşağıda açıklanan değişkenlere verilen değerlere bağlı olarak cephe çizimi yapılır.



Şekil 4.3 "Cephe Çizimi" isimli diyalog kutusu

Cephe Bilgileri: Çizilen cephenin altına açıklama yazdırılmak isteniyor ise bu bölümün içinden yazdırılabilir. Bu durumda ekrana (Şekil 4.4)' deki "Cephe Bilgileri" isimli diyalog kutusu gelecektir. Burada kullanılabilen "Proje adı", "Cephe adı", yazdırılacak olan yazının "Ayarı", "Stili", "Yüksekliği" ve "Açısı" parametreleridir. Bu parametreler üzerinde kullanıcı istediği değerleri vererek parametrik olarak cepheyi çizdirebilir. Bu tür çizim yaptırmaya parametrik tasarım adı verilmektedir. Burada yalnız yatay cephe

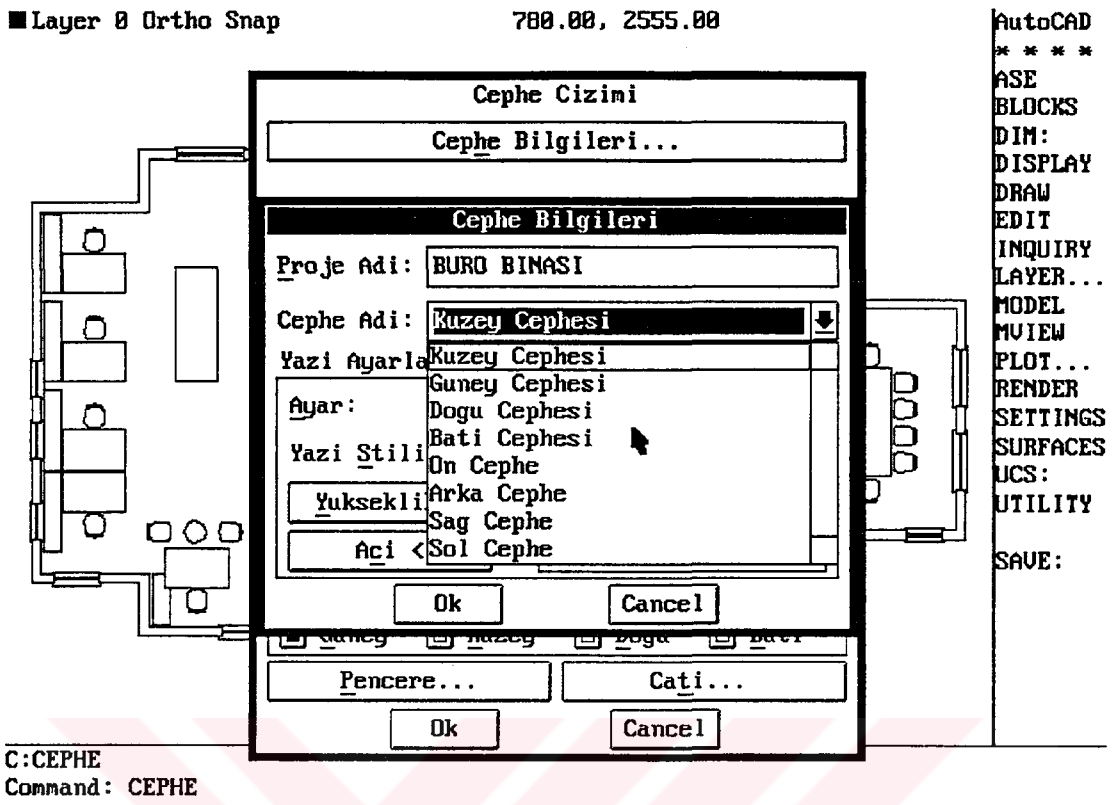
çizgilerinin ve pencerelerin bölme parametreleri değiştirilebilmektedir.



Şekil 4.4 "Cephe Bilgileri" isimli diyalog kutusu

Proje adı: Cephesi çizilen projenin ne projesi, veya kime ait alduğu bu bölüme yazılabilir. Buraya yazılan değerler cephenin alt sağına yazılacaktır.

Cephe adı: Çizilmesi istenecek olan cephenin adı bu bölümde belirecektir. Bunun için kullanıcının yeniden değer girmesi gerekmemektedir. "Cephe Çizimi" adlı diyalog kutusundan seçilen "Cephe adı" ne ise o isim aynen burada belirecektir. Ancak kullanıcı "Kuzey Cephesi", "Guney Cephesi", "Dogu Cephesi", "Batı Cephesi" yerine "Arka Cephe", "On Cephe", "Sag Cephe", "Sol Cephe" olarak yön isimlerini tercih edecek ise bu durumda seçimini "Pop-up" list adı verilen listeden yapmak durumandadır. (Şekil 4.5).

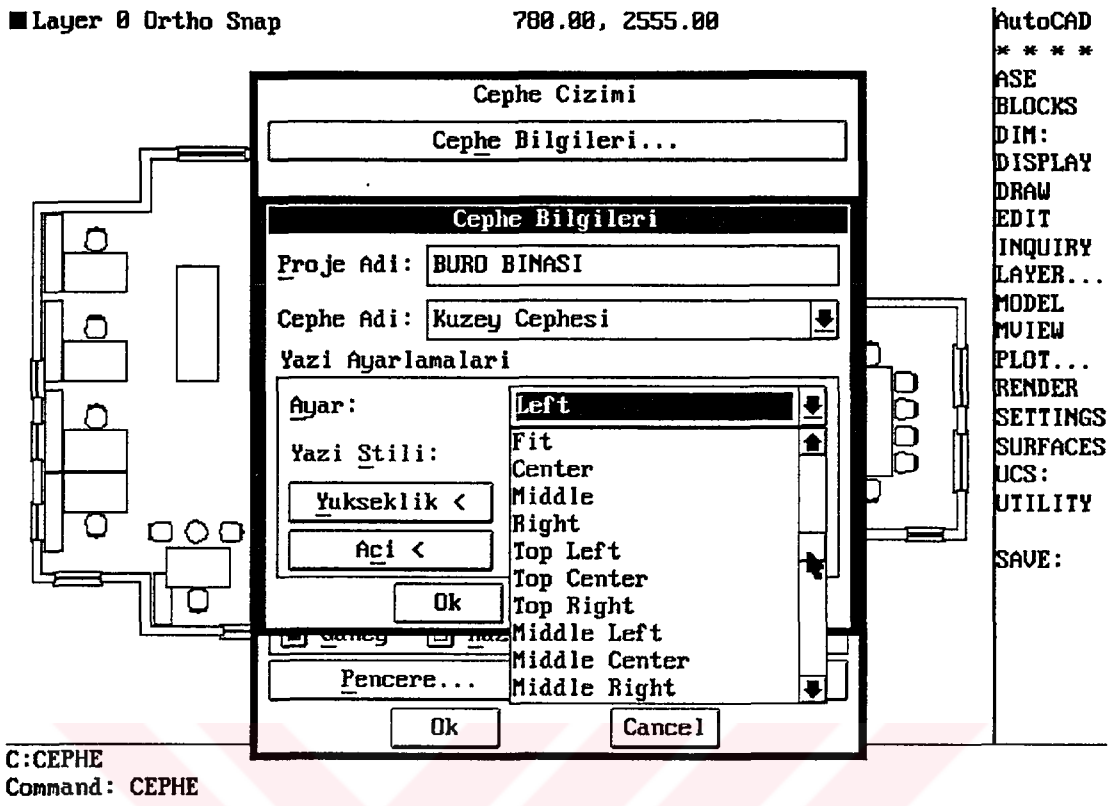


Şekil 4.5 "Cephe Adı" isimli diyalog kutusu

Ayar: Yazıların düzenini belirlemek için kullanılacaktır. "Pop-up" list şeklinde tanımlanmıştır. Kullanılabilecek değerler sadece listede çıkanlardır. Bu durumda yazı sağa, sola dayalı olarak, ortalatılarak, iki nokta arasına sıkıştırılarak veya daha başka şekillerde yazdırılabilir. (Şekil 4.6).

Yazı stili: Yazdırılacak yazının stilini belirlemek için kullanılacaktır. Daha önceden yazı stili AutoCAD'in uygun komutları ile tanımlanmalıdır. Tanımlanmış olan yazı stilleri buradan şeçilerek aktif hale gelmeleri sağlanacaktır.

Yukseklik <: Yazının yükseklik değeri buradan verilebilecektir. Sağdaki bölüme doğrudan yazılabileceği gibi "Yukseklik <" kutusu seçilerek çalışma ekranına dönülüp buradan iki nokta işaretlemek sureti ile de yükseklik değeri verilebilir.

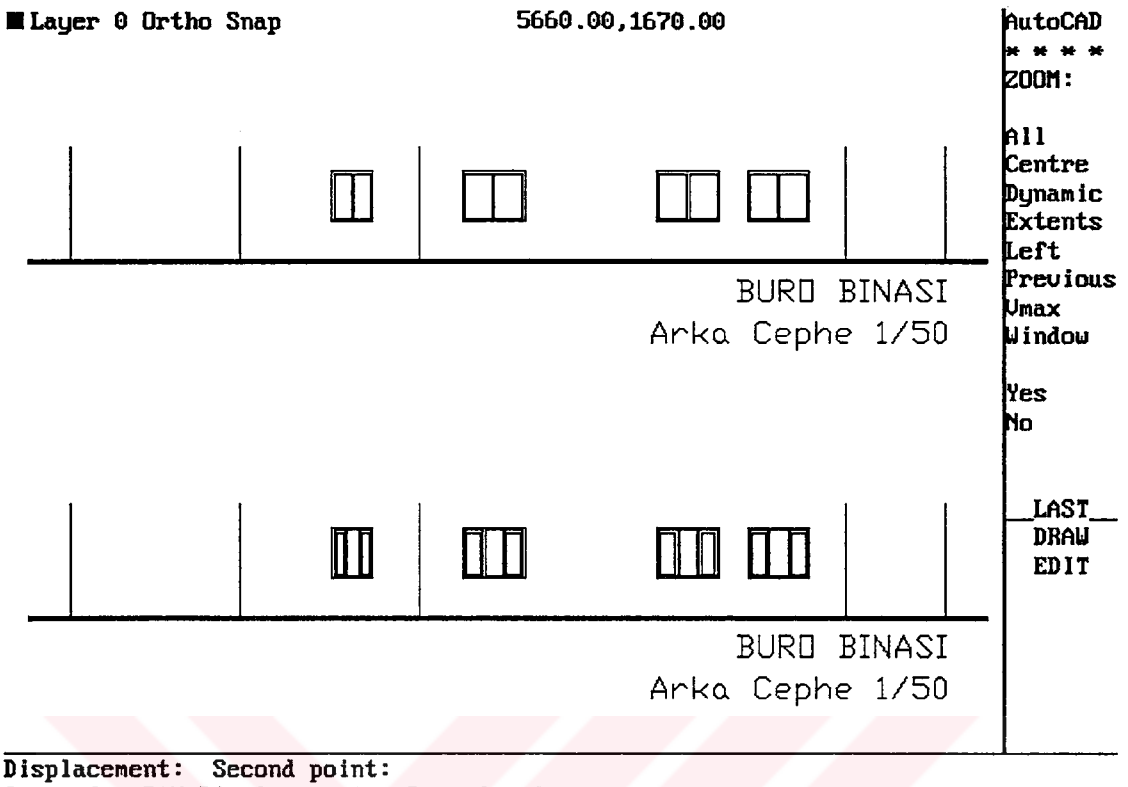


Şekil 4.6 "Ayar" isimli diyalog kutusu

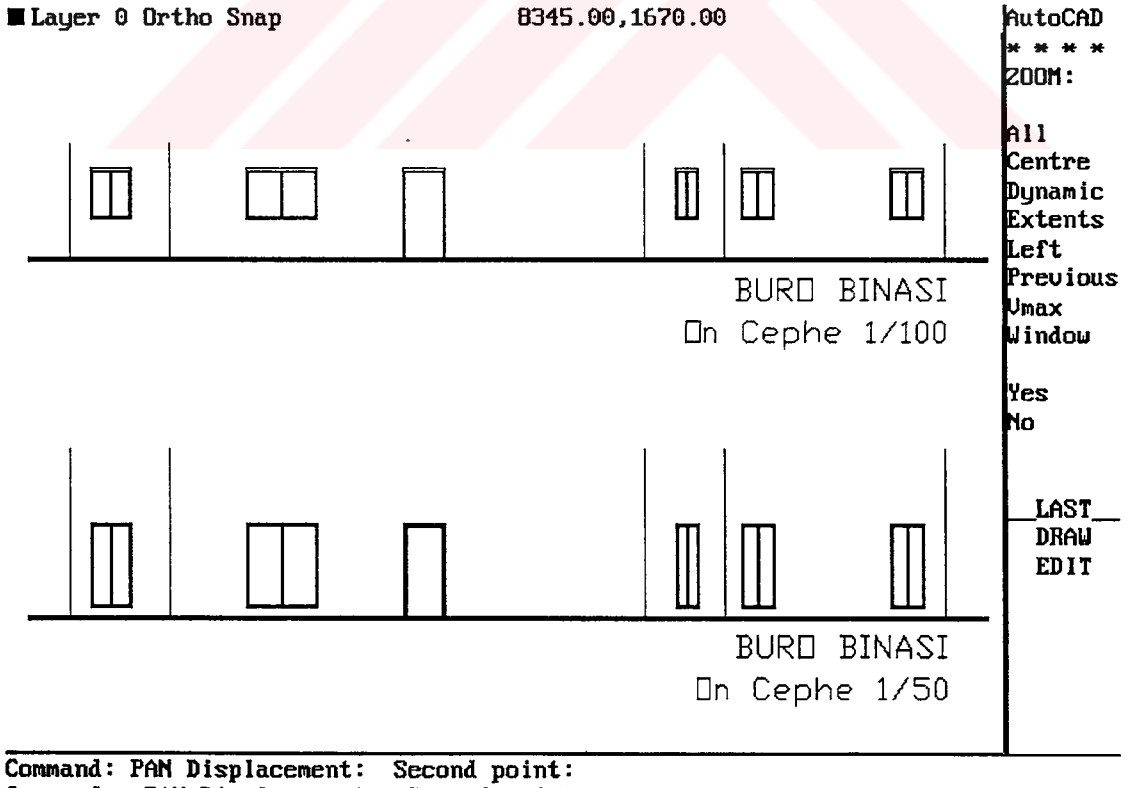
Acı <: Yazının açısı girilecektir. "Yukseklik <" parametresine benzer olarak çalışmaktadır.

"Cephe Bilgileri" bölümündeki verilerin tamamlanmasından sonra "Ok" kutucuğu seçilerek "Cephe Cizimi" adlı diyalog kutusuna dönülüp işlemlere devam edilebilir.

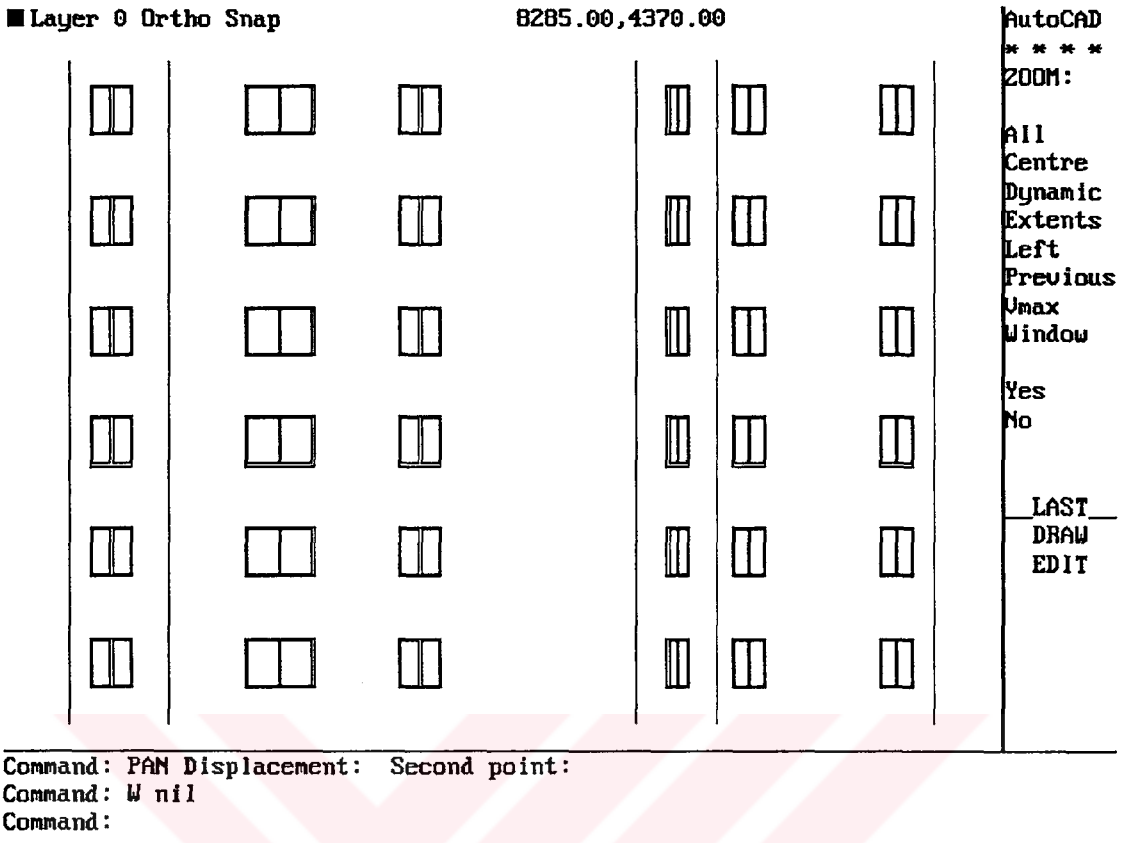
Kat adı: Bu değişkene 3 değer girilebilmektedir. Bunlar "Zemin kat", "Normal kat" ve "Catı kat" değerleridir. Bu değişken için, kullanılabilecek değerlerin seçimi, AutoCAD Release 12'nin imkanları ile sağlanan, radyo düğmesi ile yapılmıştır. Herhangi bir seçenek seçildiğinde bir diğerinin seçim dışı kaldığı görülecektir. Burada cephesi çizilmesi istenen katın yanındaki düğme seçilip çizim yapılacak kat aktif hale getirilir.



Şekil 4.7 Pencere bölme sayısı için iki örnek



Şekil 4.8 Kat yükseklikleri, Parapet yükseklikleri ve çizim ölçekleri farklı iki örnek



Şekil 4.9 Normal kat cephe çizimi örneği

Catı: Çizilmesi düşünülen katla birlikte çatının da çizilmesi isteniyorsa buradaki kutucuk işaretlenerek aktif hale getirilir. Katın üzerinde çatı çizimi istenmiyorsa bu kutucuk boş bırakılır.

Olcek: Pencerelerin çizim detayını belirlemek için ölçek seçimi yapmak gerekmektedir. Bunun için düşünülen ölçek radyo düğmesi yöntemine göre seçilmelidir. Seçilebilecek ölçekler 1/50, 1/100 ve 1/200 'dir.

Kat adedi: Sadece **Kat adı** değişkeni "Normal kat" iken kullanılabilir. Cephesi çizilmek istenen projeye ait normal kat sayısı bu değişkene atanmalıdır.

Kat yuksekligi: Çizdirilmek istenen katın yüksekliği verilmelidir.

Parapet yuksekligi: Yine çizdirilmek istenen kata ait olan pencerelerin parapet yükseklikleri verilmelidir.

Pencere yuksekligi: Çizimi yapılacak cepheye ait olan katdaki pencerelerin yükseklikleri verilmelidir.

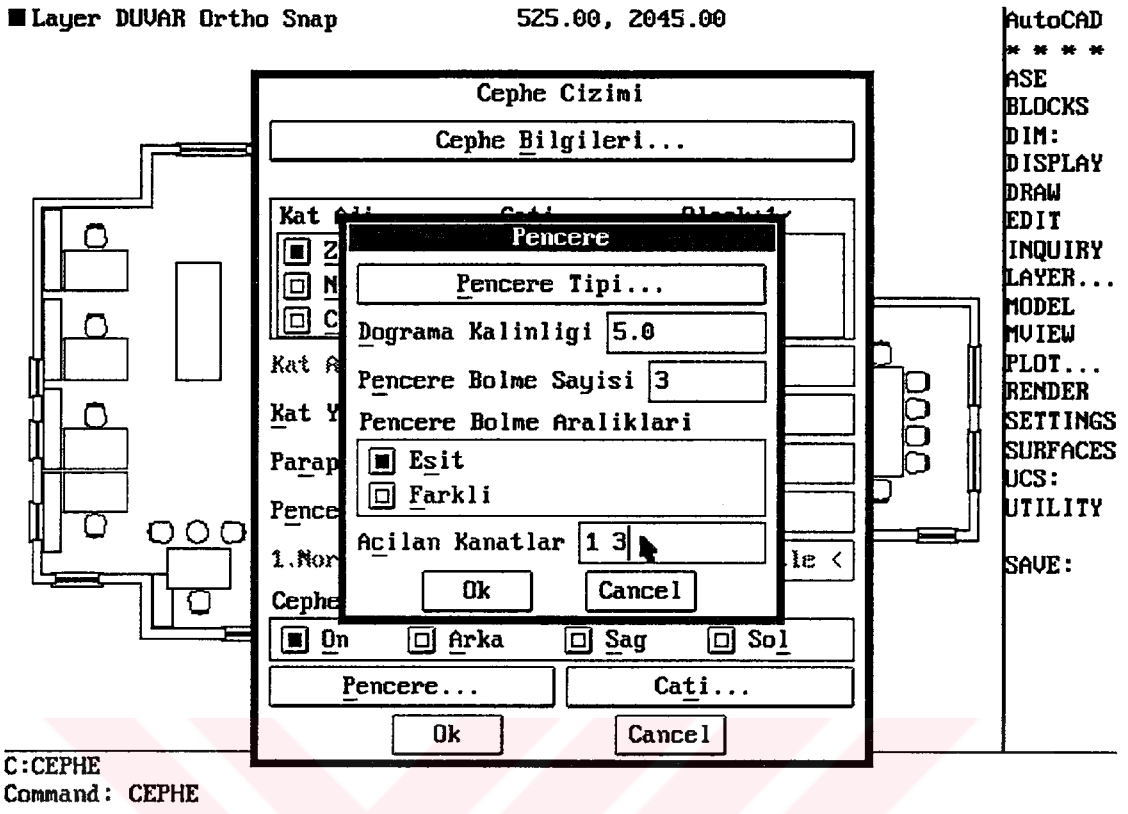
1.Normal kat alt kotu: Normal katların birincisinin başlangıç kotunun bulunabilmesi için bu değişkene 1. Normal kat alt kotu bir başka deyişle Zemin kat tavan kotunu vermek gerekmektedir.

Cephe adı: **Kat adı** değişkenine benzer olarak bu değişken için de Radyo düğmesi tanımlaması kullanılmıştır. Yani **Cephe adı** değişkeni için kullanılabilecek 4 değişik seçenek vardır. Bunlar: "Kuzey Cephesi", "Guney Cephesi", "Dogu Cephesi", "Batı Cephesi" seçenekleridir. Yine bunlardan sadece bir tanesi aktif hale getirilip seçilebilmekte, seçilenin dışındakiler seçim dışı kalmaktadır.

Pencere: Eğer pencere tip ve boyutları ile ilgili herhangi bir ayarlama yapılmak istenirse bu bölüm seçilerek "Pencere" adlı bir diyalog kutusunun (Şekil 4.10) ekrana gelmesi sağlanır.

Pencere tipi: İleriki aşama için düşünülmüş olan bu bölüm seçildiğinde Pencere tipi ile ilgili seçim yapma imkanı olacaktır. Düğme seçildiğinde ekrana bir slayt kutusu gelecektir. Burada bulunan pencere tiplerinden herhangi birisi seçilerek, **"Pencere"** başlığı altında toplanan diyalog kutusuna tekrar dönülecektir.

Doğrama kalınlığı: Pencerelere veya kapılara ait kasaların doğrama kalınlıkları bu bölüme yazılarak verilecektir.



Şekil 4.10 "Pencere" isimli diyalog kutusu

Pen. bölme sayısı: Bu bölüme yazılacak sayı ile cephede çizilmesi istenen pencerelerin kanat sayıları belirlenebilmektedir. Sadece düşey bölmeler yapılabilmektedir.

Pen. bölme aralık.: **Pencere bölme sayısı** değişkenine verilen değer 1'den farklı ise kullanılabilen bu değişken yine Radyo düğmesi özelliği ile kullanılmaktadır. Seçilebilecek Radyo düğmelerinden biri **"Eşit"** diğeri de **"Farklı"** seçenekleridir. **"Eşit"** seçeneği seçildiğinde programın plandan almış olduğu pencere genişliklerini diyalog kutusu yardımı ile **"Pencere bölme sayısı"** değişkenine verilen sayıya eşit olarak bölüp pencerelerin bölme genişliklerini bulacaktır. **"Farklı"** seçeneği seçildiği durumlarda program çalışmaya başladığında o anda çizilen pencere için genişlikler bölme sayısı kadar sorularak genişliklerin girilmesi istenecektir.

Açılan kanatlar: Pencere bölmelerinden herhangi bir tanesi açılır kanat ise; bu kutuya soldan sağa doğru olacak şekilde açılması istenen bölmenin numarası yazılarak o bölmenin açılır kanat olması sağlanmış olacaktır. Birden fazla açılır kanat varsa bunlar aralarında birer boşluk bırakılarak belirtilmelidir. Ölçek değişkenine verilen değer de burada önemlidir. 1/50 ölçek dışındaki seçimlerde açılır kanatlar belirtilse bile bunlar ihmal edilecektir.

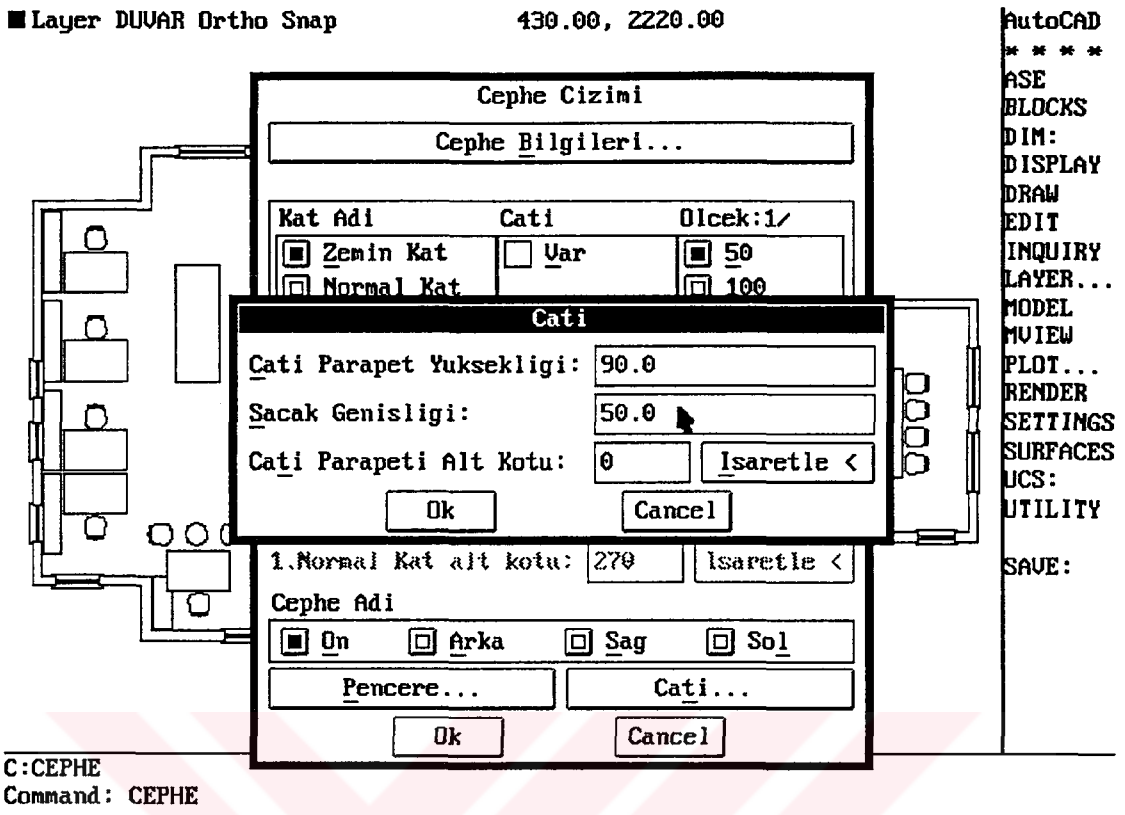
"Pencere" başlığı altında toplanan tüm bu ayarlamaların yapılmasından sonra **"Ok"** tuşu seçilerek bir önceki **"Cephe Çizimi"** adlı diyalog kutusuna dönülür.

Çatı: Çatı Parapet Yüksekliği, Saçak genişliği ve Çatı parapet alt kotu değişkenlerinden herhangi biri değiştirilmek istenirse bu kutu seçilerek ekrana **"Çatı"** adlı diyalog kutusunun gelmesi sağlanır. (Şekil 4.11). Değiştirilmek istenmiyorsa bu bölüme girmeye gerek olmayacaktır. Bu durumda ön kabul değerleri kullanılacaktır.

Catı parapet yuk.: Çatı parapetinin yükseklik değeri **"Parapet yuksekligi"** değişkenine eşitlenmiştir. Bu durumda **"Parapet yuksekligi"** değerine verilecek her değer **"Catı parapet yuksekligi"** değişkenine de atanmış sayılacaktır. Ancak değiştirmek istenirse bu imkan da mevcuttur.

Sacak genisligi: Çatı saçak genişliği buraya girilmelidir. Değiştirilmediği sürece ön kabul değeri olan "50 cm." kullanılacaktır.

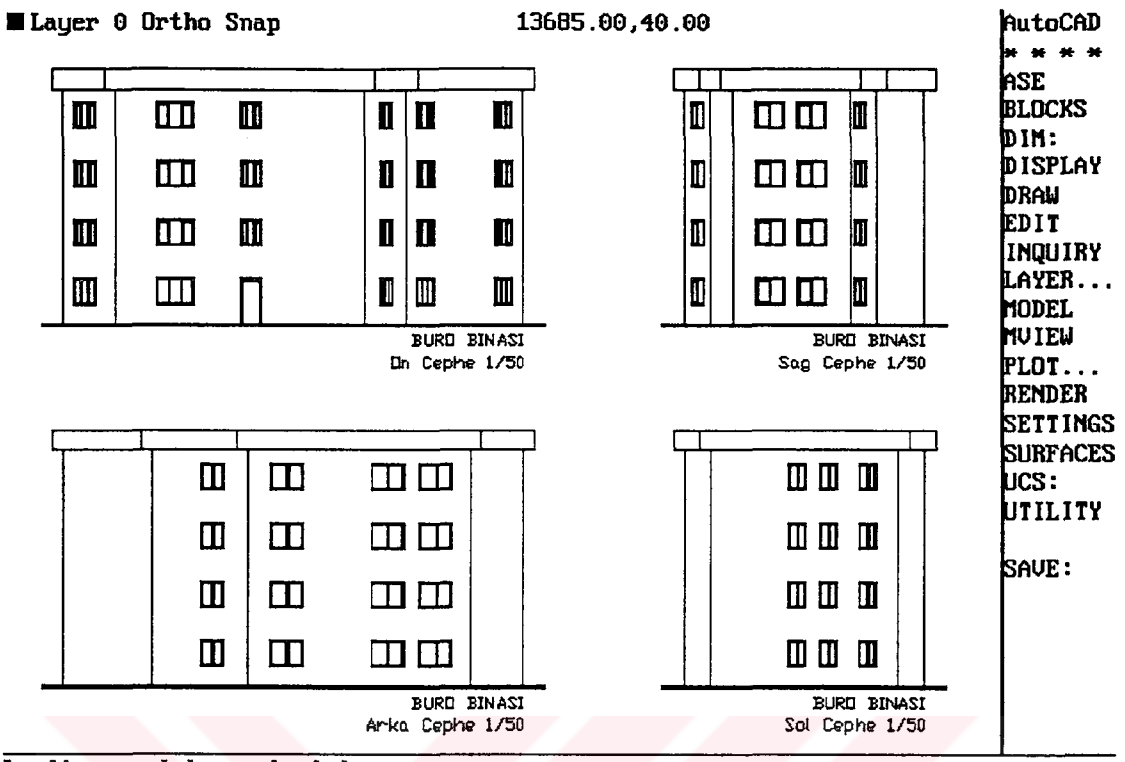
Çatı par. alt kotu: Çizilmesi istenen çatının alt kotunun ne olacağı buraya yazılmalıdır. Eğer çizilen cephe nomal kat cephesi ise ve kat ile birlikte çatı çizimi de istenmiş ise buraya herhangi bir değer vermeye gerek yoktur. Program çalışırken çatı parapeti alt kotunu normal kat adedine göre hesaplayacaktır.



Şekil 4.11 "Çatı" isimli diyalog kutusu

Tüm bu ayarlamalardan sonra **"Ok"** düğmesi seçildiğinde **"Cephe Cizimi"** adlı diyalog kutusuna dönülecektir.

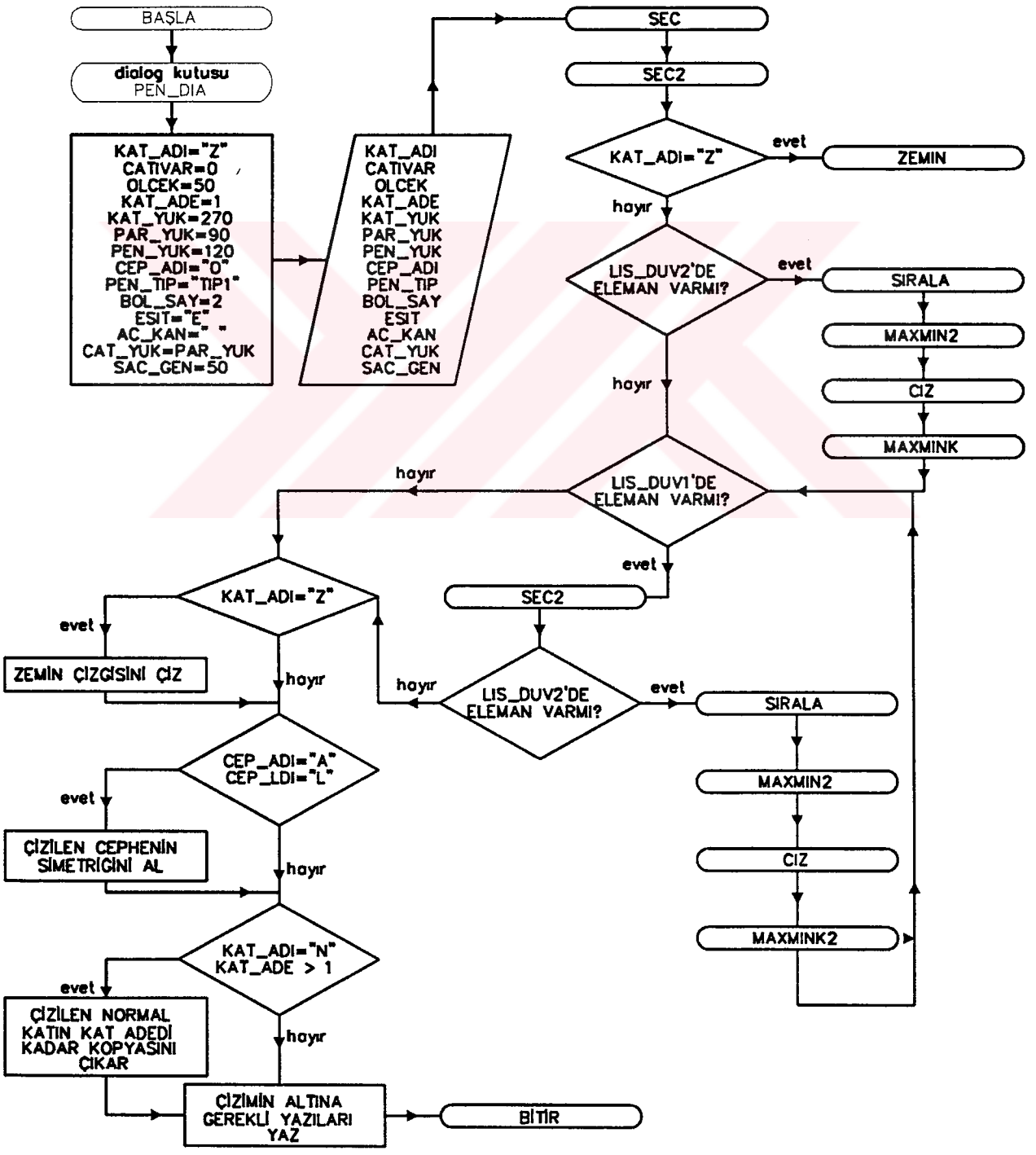
Bu bölümde de herhangi bir aksaklık yok ise buradan da **"Ok"** düğmesi seçilerek program başlatılmış olur. Bu şekilde yapılmış bir çalışma sonucu elde edilen cephe çizimleri (Şekil 4.12)' de gösterilmiştir.

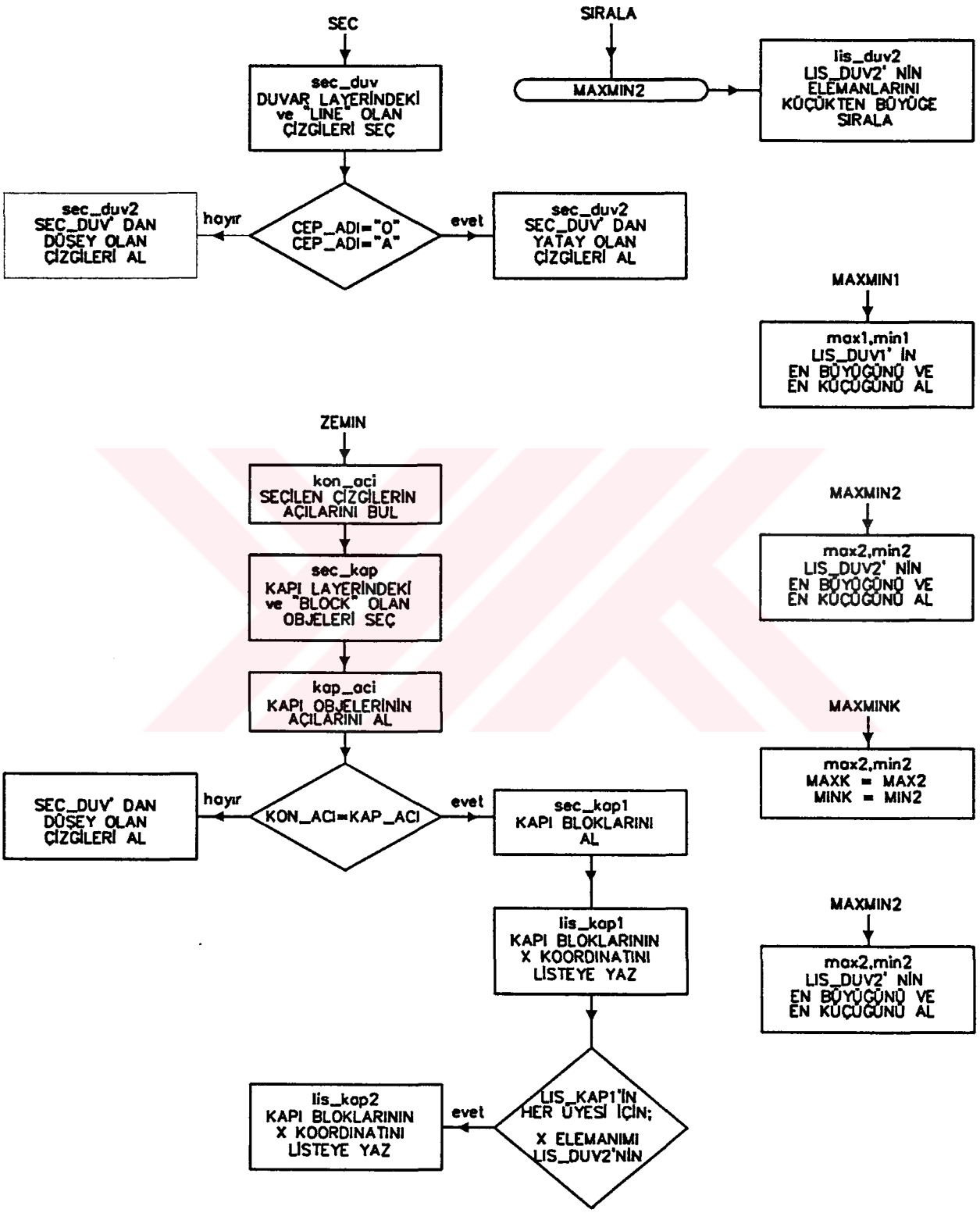


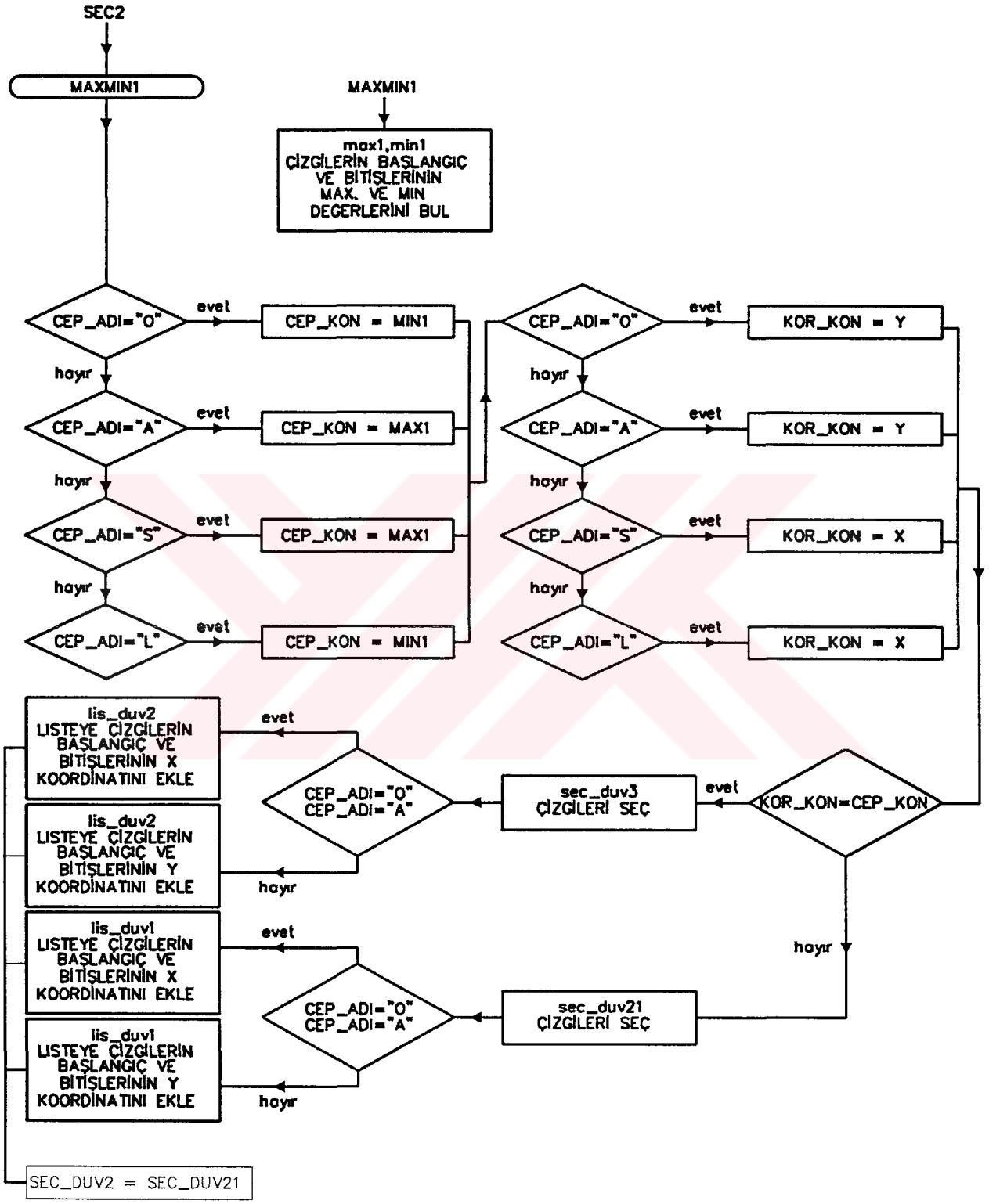
Şekil 4.12 Örnek Cephe Çizimi

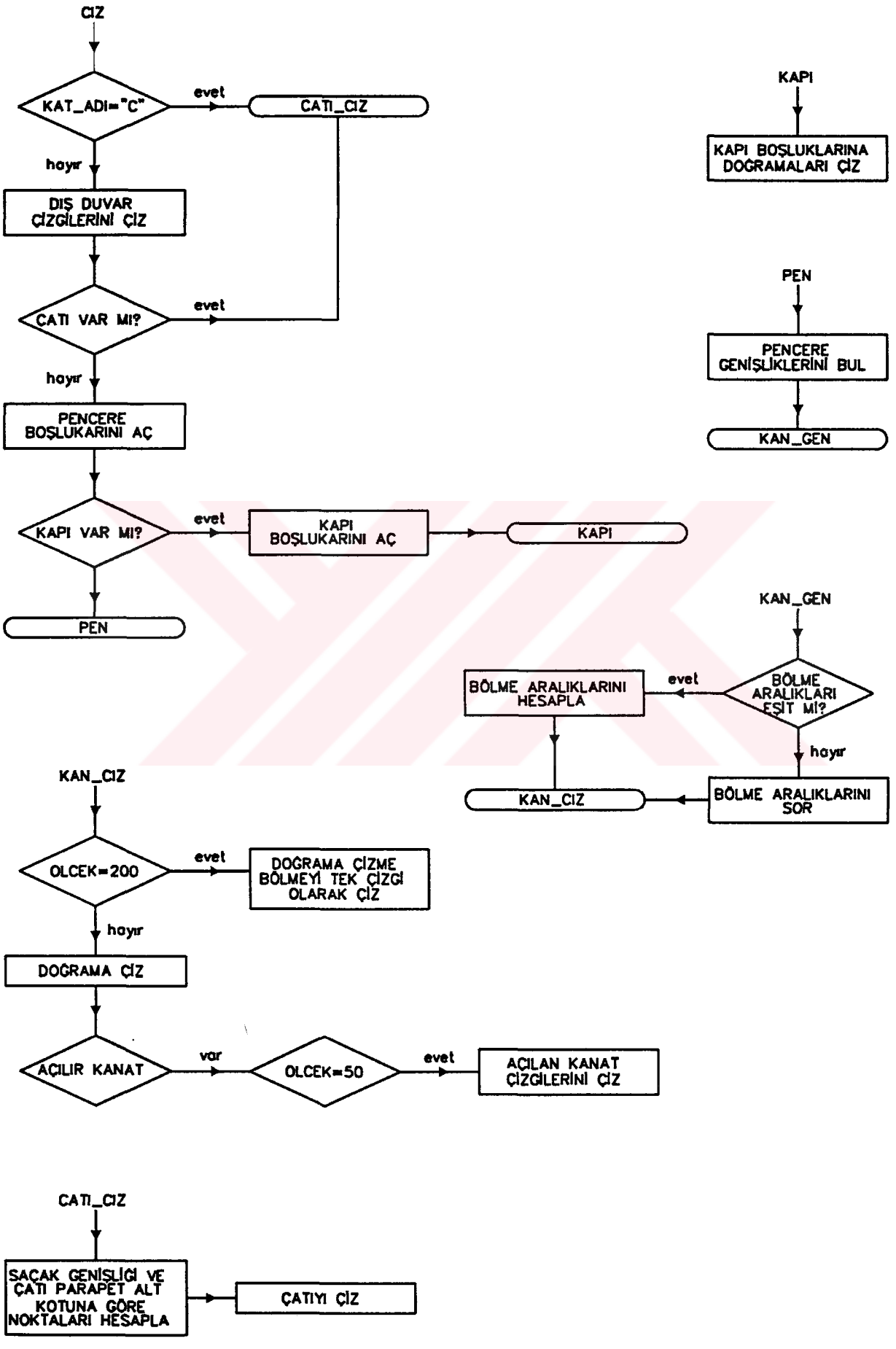
4.3. AKIŞ DİYAGRAMI

Aşağıda geliştirilen programın akış diyagramı verilmektedir. Akış diyagramından bir önceki alt bölümde açıklanan adımları takib etme olanağı bulunmaktadır. Akış diyagramında yer alan değişken adları ve sırası bir önceki bölümde ayrıntılı olarak ele alınmıştır,




SEC
sec_duv
DUVAR LAYERİNDEKİ ve "LINE" OLAN ÇİZGİLERİ SEÇ
CEP_ADI="0" CEP_ADI="A"
hayır
sec_duv2
SEÇ_DUV' DAN DÜŞEY OLAN ÇİZGİLERİ AL
evet
sec_duv2
SEÇ_DUV' DAN YATAY OLAN ÇİZGİLERİ AL
SIRALA
MAXMIN2
lis_duv2
LIS_DUV2' NİN ELEMANLARINI KÜÇÜKTEN BÜYÜĞE SIRALA
MAXMIN1
max1,min1
LIS_DUV1' İN EN BÜYÜĞÜNÜ VE EN KÜÇÜĞÜNÜ AL
ZEMIN
kon_aci
SEÇİLEN ÇİZGİLERİN AÇILARINI BUL
sec_kap
KAPI LAYERİNDEKİ ve "BLOCK" OLAN OBJELERİ SEÇ
kap_aci
KAPI OBJELERİNİN AÇILARINI AL
KON_ACI=KAP_ACI
hayır
SEÇ_DUV' DAN DÜŞEY OLAN ÇİZGİLERİ AL
evet
sec_kap1
KAPI BLOKLARINI AL
lis_kap1
KAPI BLOKLARININ X KOORDINATINI LİSTEYE YAZ
LIS_KAP1'İN HER ÜYESİ İÇİN;
X ELEMANIMI LIS_DUV2'NİN
evet
lis_kap2
KAPI BLOKLARININ X KOORDINATINI LİSTEYE YAZ
MAXMIN2
max2,min2
LIS_DUV2' NİN EN BÜYÜĞÜNÜ VE EN KÜÇÜĞÜNÜ AL
MAXMINK
max2,min2
MAXK = MAX2
MINK = MIN2
MAXMIN2
max2,min2
LIS_DUV2' NİN EN BÜYÜĞÜNÜ VE EN KÜÇÜĞÜNÜ AL


SEC2
MAXMIN1
MAXMIN1
max1,min1
ÇİZGİLERİN BAŞLANGIÇ
VE BİTİŞLERİNİN
MAX. VE MIN
DEĞERLERİNİ BUL
CEP_ADI="O"
evet
CEP_KON = MIN1
hayır
CEP_ADI="A"
evet
CEP_KON = MAX1
hayır
CEP_ADI="S"
evet
CEP_KON = MAX1
hayır
CEP_ADI="L"
evet
CEP_KON = MIN1
CEP_ADI="O"
evet
KOR_KON = Y
hayır
CEP_ADI="A"
evet
KOR_KON = Y
hayır
CEP_ADI="S"
evet
KOR_KON = X
hayır
CEP_ADI="L"
evet
KOR_KON = X
lis_duv2
LİSTEYE ÇİZGİLERİN
BAŞLANGIÇ VE
BİTİŞLERİNİN X
KOORDİNATINI EKLE
evet
lis_duv2
LİSTEYE ÇİZGİLERİN
BAŞLANGIÇ VE
BİTİŞLERİNİN Y
KOORDİNATINI EKLE
hayır
CEP_ADI="O"
CEP_ADI="A"
sec_duv3
ÇİZGİLERİ SEÇ
evet
KOR_KON=CEP_KON
hayır
lis_duv1
LİSTEYE ÇİZGİLERİN
BAŞLANGIÇ VE
BİTİŞLERİNİN X
KOORDİNATINI EKLE
evet
lis_duv1
LİSTEYE ÇİZGİLERİN
BAŞLANGIÇ VE
BİTİŞLERİNİN Y
KOORDİNATINI EKLE
hayır
CEP_ADI="O"
CEP_ADI="A"
sec_duv21
ÇİZGİLERİ SEÇ
SEC_DUV2 = SEC_DUV21


CIZ
KAT_ADI="C"
evet
CATI_CIZ
hayır
DIŞ DUVAR ÇİZGİLERİNİ ÇİZ
CATI VAR MI?
evet
hayır
PENCERE BOŞLUKARINI AÇ
KAPI VAR MI?
evet
KAPI BOŞLUKARINI AÇ
KAPI
PEN
KAPI
KAPI BOŞLUKLARINA DOĞRAMALARI ÇİZ
PEN
PENCERE GENİŞLİKLERİNİ BUL
KAN_GEN
KAN_GEN
BÖLME ARALIKLARINI HESAPLA
evet
BÖLME ARALIKLARI EŞİT Mİ?
hayır
KAN_CIZ
BÖLME ARALIKLARINI SOR
KAN_CIZ
OLCEK=200
evet
DOĞRAMA ÇİZME BÖLMEYİ TEK ÇİZGİ OLARAK ÇİZ
hayır
DOĞRAMA ÇİZ
AÇILIR KANAT
var
OLCEK=50
evet
AÇILAN KANAT ÇİZGİLERİNİ ÇİZ
CATI_CIZ
SAÇAK GENİŞLİĞİ VE ÇATI PARAPET ALT KOTUNA GÖRE NOKTALARI HESAPLA
ÇATIYI ÇİZ

4.5. CEPHE ÇİZİMİ YAPTIRILACAK PLANIN ÖZELLİKLERİ

Cephesi çizdirilecek planın sahip olması gerekli bazı çizim ilkeleri vardır. Bu ilkelere sahip olmayan planların henüz programın bu haliyle cephelerinin çizilmesine olanak yoktur. Program geliştikçe bu ilkelerde artıp eksilebilmektedir.

1.- Plan çizimlerinin dik açılı olması gerekmektedir. Program eğik bir planın cephesini çizememektedir. Çizim işlemi program içinden çizdilecek yöne ait duvar nesnelerinin seçimi ile başlamaktadır. Kullanılan yönteme göre seçilecek duvar nesnelerinin tesbiti Kuzey cephesi için ekranın en üstünde kalan ve aynı yatay hizada bulunan duvar nesneleridir. Aynı şekilde Güney cephesi için en altta kalanlar, Doğu cephesi için en sağda kalanlar ve Batı cephesi için de en solda kalanlar seçilmektedir. Eğik olan duvar nesnelerinde duvar hizalarının belirlenmesi ve onların içinde kapı ve pencere boşluklarının açılması zor bir işlem olduğu için şu aşamada yaptırılmamıştır.

2.- Plan çizgileri "DUVAR" Layer'inde ve "LINE" olarak çizilmelidir. Programın seçilecek nesneleri tanıyabilmesi için plandaki çizgilerin bu şekilde çizilmeleri gerekmektedir.

3.- Kapı çizimlerinin "KAPI" Layer'inde ve "BLOCK" olmaları gerekmektedir. Yukarıdakine benzer olarak açılan boşlukların kapı boşluğu mu yoksa pencere boşluğu mu kararının verilebilmesi için planda kapı olan yerlerdeki çizimlerin bu özelliği taşıması gerekmektedir.

4.- Program çalıştığında planın tümünün ekranda görünüyor olması gerekmektedir. Nesne seçimi sırasında program sadece o anda ekranda görebildiği nesnelerin seçimini yapmaktadır. Dolayısı ile ekranda görünmeyen nesneler seçilmediği takdirde programın işleyişi doğru olamayacağı için oluşan cephe çizimi yanlış olacaktır.

5.- Bina da içeriye doğru geri çekilmeler (cep) olmamalıdır. Bu şekilde yapılmış bir uygulamada program geriye çekilmeleri pencere boşluğu olarak algılayacaktır. Bu durumda duvar olması gereken yere pencere çizecek ve yine yanlış bir cephe çizimi elde edilecektir.

SONUÇLAR

Program bir CAD yazılımı olan AutoCAD Release 12 ile 2 boyutlu mimari çizim yapanlara yönelik olarak hazırlanmıştır. Program, tasarımcının eskiz sırasında, çizim için harcayacağı zaman ve emeği en aza indirmekte ve doğru ölçülerde cephe etüdlerinin yapılmasını sağlamaktadır. Bu sayede tasarımcı çizim için harcaması gereken süre içinde daha fazla cephe alternatifi geliştirebilmek için tasarıma faha fazla zaman ayırabilecektir. Bir başka deyişle tasarımcı kendi işi olan tasarım işini yapacak onun için gerekli olan çizimleri programa bırakacaktır. Çizim için harcanacak süre ve emek çok aza indiği için tasarımcı cephe alternatif sayısını artırabilecektir. Veya yapmış olduğu cephede değiştirmeyi düşündüğü her türlü kararın sonucunu hızlı bir şekilde görme imkanı olacaktır. Programın kullanımı Diyalog kutularının yardımı ile olduğu için çok kolaydır. Öneriler bölümünde ki açıklamalar dikkate alınarak kullanıldığında, program planı çizilmiş bir projenin 4 cephesini ana hatları ile çizmektedir. Bu da Mimara çok hızlı bir şekilde cephe etüdleri yapabilme imkanı verecektir. Elde edilecek cephe 1/200 - 1/100 düzeyinde olacağı için programın, cephenin sadece ön etüdlerini yapması ilk amaç olmuştur. Buradan elde edilen cephe alternatiflerinin altlık gibi kullanılması ile mimarın cephe üzerinde gerekli detay ve ayrıntıları işlemesi ve geliştirmesi sağlanabilir. Projenin planlarında yapılacak her türlü düzeltme, program kullanılarak çok kolay ve hızlı bir şekilde cepheye yansıtılabilecek ve değişikliğin cepheye getireceği değişiklikler çok kısa sürede görülebilecektir. Bu sayede çok daha fazla sayıda cephe alternatifleri üzerinde çalışma imkanı doğacaktır. Yapılacak değişiklikler sadece plandaki ile kalmamaktadır. Binanın Kat yüksekliği, Kat Adedi, Parapet yüksekliği, Pencere yüksekliği, Çatı saçak genişliği ve Çatı parapet yüksekliği de isteğe göre değiştirilip hızlı bir şekilde cephe üzerindeki etkileri görülebilecektir.

Program geliştirilmeye açıktır. Program ile cephede bulunan her bir mekana ait pencereler ayrı ayrı ve değişik form, boyut ve şekilde çizdirilebilinir. Program bir editör gibi kullanılarak hem cephenin genel hatlarında hem de pencere modüllerinde gerekli düzeltmeler yaptırılabilinir. Bir projenin çatı planından faydalanarak programın cephede çatı çizmesi sağlanabilinir.

KAYNAKLAR

[1] Autodesk, AutoLISP Release 12 Programmer's Reference, AUTODESK BV, June 1992

[2] Autodesk, AutoCAD Release 12 Reference Manual, AUTODESK BV, June 1992

[3] Autodesk, AutoCAD Release 12 Customization Manual, AUTODESK BV, June 1992

[4] Autodesk, AutoCAD Release 12 Extras Manual, AUTODESK BV, June 1992

[5] AutoCAD 2.5&2.6, ELEKTRONİK İLETİŞİM AJANSI YAYINLARI, ANKARA 1988

[6] Ballard, D.H., Brown, C.M., Computer Vision, Prentice-Hall, 1982.

[7] Batchelor, B.G., Pattern Recognition, Plenum Press, 1978.

[8] Brown, P.R., and McLean, C.R., Interactive Process Planning in the AMRF, National Institute of Standards and Technology, USA, 1986.

[9] Chank, T.C., and Wysk, R.A., Integrating Cad and CAM through automated Process Planning, International Journal of Production Research, Vol. 22, No. 5, 1984. pp. 877-894.

[10] Cpriani, R., Lagomarsino, A.D., Stagnaro, A., Valenti, E., Tziana, S., Some Years' Experience Teaching CAAD, içinde (Derleyen) M. McCullough, W.J. Mitchell, P. Purcell, Electronic Design Studio, The MIT Press, Cambridge Mass. 1990. pp. 347-361.

[11] Critchlow, A.J., Introduction to Robotics, MacMillan Publishing Company, 1985.

[12] De Cola, S., De Cola, B., Pentasuglia, F., Messina 1908: The Invisible City, içinde (Derleyen) M. McCullough, W.J. Mitchell, P. Purcell, Electronic Design Studio, The MIT Press, Cambridge Mass. 1990. pp. 239-246

[13] Danahy, J.W., Irises in a Landscape: An Experiment in Dynamic Interaction and Teaching Design Studio, içinde (Derleyen) M. McCullough, W.J. Mitchell, P. Purcell, Electronic Design Studio, The MIT Press, Cambridge Mass. 1990. pp. 363-374.

[14] Fairhurst, M.C., Computer Vision for Robotics Systems, Prentice-Hall, 1982.

[15] Fargas, J., Papazian P., Metaphors in Design: An Experiment with a Frame, Two Lines and Two Rectangles, içinde (Derleyen), K.M. Kensek, D. Noble, Computer Supported Design in Architecture: Mission Method and Madness, The Association for Computer Aided Design in Architecture, 1992. pp. 13-22.

[16] Flemming, U., Syntactic Structures I. Architecture: Teaching Composition with Computer Assistance, içinde (Derleyen) M. McCullough, W.J. Mitchell, P. Purcell, Electronic Design Studio, The MIT Press, Cambridge Mass. 1990. pp. 31-48.

[17] Fox, C.W., Integrating Computing into an Architectural Undergraduate Program, içinde (Derleyen) M. McCullough, W.J. Mitchell, P. Purcell, Electronic Design Studio, The MIT Press, Cambridge Mass. 1990. pp. 377-391.

[18] Hatvany, J., Newman, W.M., Sabin, M.A., World Survey of Computer-Aided Design, Computer Aided Design, Vol. 25, No. 2, December 1993. pp. 776-798.

[19] Hummel, K.E., and Brooks, S.L., Symbolic Representation of Manufacturing Features for an Automated Process Planning System, Proceedings of ASME Winter Annual Meeting, Anaheim, California, USA, 1986.

[20] Klafter, R.D., Chmielewski, T.A., Robotics Engineering: An Integrated Approach, Prentice-Hall, 1989.

[21] Marshall, T.B., Computers as a Graphic Medium in Conceptual design, içinde (Derleyen), K.M. Kensek, D. Noble, Computer Supported Design in Architecture: Mission Method and Madness, The Association for Computer Aided Design in Architecture, 1992. pp. 39-47.

[22] McCullough, M., Low-Threshold Modeling, içinde (Derleyen) M. McCullough, W.J. Mitchell, P. Purcell, Electronic Design Studio, The MIT Press, Cambridge Mass. 1990. pp. 413-426.

[23] Mitchell, W.J., Liggett, R.S., Kvan, T., The Art of Computer Graphics Programming: A Structured Introduction for Architects and Designers, New York: Van Nostrand Reinhold Company, 1987. pp. 201-250.

[24] Myers, W.L., Barnes II, J.T., Hoerner, M.R., Using Object Oriented Descriptive design Programming in the Design and Configuration of Offshore Platforms, Wisdom Systems a McDermott Company, Technical Paper: WTP 87-1, 10/20/87. pp. 1-15.

[25] Nagakura, T., Shape Recognition and Transformation: A Script-Based Approach, içinde (Derleyen) M. McCullough, W.J. Mitchell, P. Purcell, Electronic Design Studio, The MIT Press, Cambridge Mass. 1990. pp. 149-185.

[26] Nau, D., and Gray, M., SIPS: An Application of Hierarchical Knowledge Clustering to Process Planning, Symposium on Integrated and Intelligent Manufacturing, ASME Winter Annual Meeting, Anaheim, California, USA, 1986.

[27] Prabhakar, S., Henderson, M.R., Automatic Form-Feature Recognition Using Neural-Network-Based Techniques on Boundary Representations of Solid Models, Computer Aided Design, Vol. 24, No. 7, July 1992. pp. 381-393.

[28] Rodriguez, J., Seirag, A.A., Algorithmic Rule-Based Methodology for Shape Synthesis: 2D Cases, Computer-Aided Design, Vol. 24, No. 8, August 1992. pp. 411-424.

[29] K.M. Kensek, D. Noble, (Derleyen), Computer Supported Design in Architecture: Mission Method and Madness, The Association for Computer Aided Design in Architecture, 1992.

[30] Mitchell, W.G., Computer-Aided Architectural Design, Petrocelli / Charter New York 1977. pp. 3-6, 21

[31] Van Bakergem, D., Image Collection in the desgn Studio, içinde (Derleyen) M. McCullough, W.J. Mitchell, P. Purcell, Electronic Design Studio, The MIT Press, Cambridge Mass. 1990. pp. 261-271.

[32] Miller, F.C., Form Processing Workshop: Architectural Design and Solid Modeling at MIT, içinde (Derleyen) M. McCullough, W.J. Mitchell, P. Purcell, Electronic Design Studio, The MIT Press, Cambridge Mass. 1990. pp. 441-455.

[33] Mitchell, W.J., Liggett, R.S., Tan, M., Top-down Knowledge-based design, içinde (Derleyen) M. McCullough, W.J. Mitchell, P. Purcell, Electronic Design Studio, The MIT Press, Cambridge Mass. 1990. pp. 137-148.

EKLER

Ek A. PROGRAM METNİ

Geliştirilen AutoLISP programının metni aşağıda verilmektedir.

```
(defun kan_gen()
 (if (= esit "e") (progn
                  (setq kon_deg 0)
                  (repeat bol_say
                   (setq kon_deg (1+ kon_deg))
                   (if (= olcek 200)
                    (setq kan_g (/ pen_g (1+ bol_say))
                          x (+ x kan_g))
                    (setq x (+ x dog_kal)
                          kan_g (/ (- pen_g (* (+ bol_say 1) dog_kal)) bol_say))
                   )
                   (kan_ciz)
                   (setq x (+ x kan_g))
                  )
                 )
                 (progn
                  (if (< deg2 bol_say)
                   (progn
                    (princ (strcat (itoa pen_num) ". pencere, "))
                    (princ deg2)
                    (if (/= olcek 200) (setq x (+ x kan_g dog_kal)))
                    (setq kan_g (getreal ".kanat genisligi: "))
                    (if (= olcek 200) (setq x (+ x kan_g)))
```

```
                (setq kan_top (+ kan_g kan_top))
                (setq kon_deg deg2)
                (setq deg1 bol_say)
                (setq deg2 (1+ deg2))
                (kan_ciz)
               )
               (progn
                (setq x (+ x kan_g dog_kal))
                (setq kan_g (- pen_g (+ kan_top (* (+ bol_say 1) dog_kal))))
                (setq deg3 1)
                (setq kon_deg bol_say)
                (kan_ciz)
               )
              )
             )
  )
)

(defun kan_ciz()
  (if (= olcek 200)
    (progn
      (command "line" (list x y) (list x (+ y pen_y)) "")
      (setq kan_g 0)
    )
    (command "pline" (list x (+ y dog_kal))
                     (list x (+ (- pen_y dog_kal) y))
                     (list (+ x kan_g) (+ (- pen_y dog_kal) y))
                     (list (+ x kan_g) (+ y dog_kal)) "c")
  )
  (if (> (strlen ac_kan) 0)
    (progn
      (setq don_kan 1 kon_ac_kan2 "")
      (repeat (strlen ac_kan)
        (setq kon_ac_kan (substr ac_kan don_kan 1))
        (setq kon_ac_kan2 (strcat kon_ac_kan2 kon_ac_kan))
        (if (= kon_ac_kan " ")
          (progn
            (setq kon_ac_kan3 (append kon_ac_kan3 (list (read kon_ac_kan2)))
```

```
            kon_ac_kan2 "")
        )
       )
       (setq don_kan (1+ don_kan))
     )
     (if (/= kon_ac_kan " ")
        (setq kon_ac_kan3 (append kon_ac_kan3 (list (read kon_ac_kan))))
     )
     (if (and (= olcek 50) (member kon_deg kon_ac_kan3))
        (command "offset" dog_kal (list x (+ y dog_kal))
                          (list (+ x dog_kal) (+ y dog_kal)) ""
        )
     )
     (setq kon_ac_kan3 nil)
   )
  )

  (if (= esit "e")
    (progn
    )
    (if (< deg2 bol_say) (kan_gen) (if (/= deg3 1) (kan_gen)))
  )
)

(defun pen()
  (setvar "cmdecho" 0)
  (setq deg1 1 deg2 1 deg3 0 kan_g 0 kan_top 0)
  (setq pen_tip 1)
  (setq bas_n p1)
  (setq pen_g (- (car p2) (car p1)))
  (setq pen_y pen_yuk)
  (setq x (car bas_n) y (cadr bas_n))
  (kan_gen)
)

(defun kapi()
  (setq pen_g (- (car p2) (car p1)))
  (setq pen_y (+ par_yuk pen_yuk))
```

```
 (setq x (car bas_n) y (cadr bas_n))
 (if (or (= olcek 50) (= olcek 100))
  (progn
   (command "offset" dog_kal p1 (list (+ dog_kal (car p1)) (cadr p1)) "")
   (if (= olcek 50)
    (command "offset" dog_kal (list (+ dog_kal (car p1)) (cadr p1))
                    (list (+ (* 2 dog_kal) (car p1)) (cadr p1)) "")
   )
  )
 )
)

(defun zemin()
 (setq pt1 (getvar "extmin") pt2 (getvar "extmax"))
 (setq sec_kap (ssget "c" pt1 pt2 '((-4 . "<AND") (8 . "kapi") (0 . "INSERT")
                    (-4 . "AND>"))))
 (setq don 0 don2 0 lis_kap1 nil lis_kap2 nil)
 (setq kon_aci (cond ((= cep_adi "G") 0)
                ((= cep_adi "K") 180)
                ((= cep_adi "D") 90)
                ((= cep_adi "B") 270))
 )
 (setq sec_kap1 (ssadd))
 (while (< don (sslength sec_kap))
  (setq w (entget (ssname sec_kap don)))
  (setq kap_x (cadr (assoc 10 w)) kap_y (caddr (assoc 10 w)))
  (setq kap_aci (cdr (assoc 50 w)))
  (if (= kap_aci kon_aci)
   (progn
    (setq sec_kap1 (ssadd (ssname sec_kap don) sec_kap1))
    (setq lis_kap1 (append lis_kap1 (list kap_x)))
   )
  )
  (while (< don2 (length lis_kap1))
   (if (member (nth don2 lis_kap1) lis_duv2) (setq lis_kap2 (append lis_kap2 (list kap_x))))
   (setq don2 (1+ don))
  )
  (setq don (1+ don))
```

```
 )
)

(defun sec()
 (setq sec_duv (ssget "x" '((-4 . "<AND") (8 . "duvar") (0 . "LINE")
                        (-4 . "AND>"))))
 (setq don 0 lis_duv1 nil)
 (setq sec_duv2 (ssadd))
 (while (< don (sslength sec_duv))
  (setq w (entget (ssname sec_duv don)))
  (setq duv_x1 (cadr (assoc 10 w)) duv_y1 (caddr (assoc 10 w))
      duv_x2 (cadr (assoc 11 w)) duv_y2 (caddr (assoc 11 w)))
  (if (or (= cep_adi "G") (= cep_adi "K")) (setq duv_kon1 duv_y1 duv_kon2 duv_y2)
                                  (setq duv_kon1 duv_x1 duv_kon2 duv_x2))
  (if (= duv_kon1 duv_kon2)
    (progn
     (ssadd (ssname sec_duv don) sec_duv2)
     (setq lis_duv1 (append lis_duv1 (list duv_kon1)))
    )
  )
  (setq don (1+ don))
 )
)

(defun sec2()
 (maxmin1)
 (setq don 0 lis_duv2 nil lis_duv1 nil)
 (setq sec_duv3 (ssadd) sec_duv21 (ssadd))
 (setq len (sslength sec_duv2))
 (while (< don len)
  (setq ent_name (ssname sec_duv2 don))
  (setq w (entget ent_name))
  (setq duv_x1 (cadr (assoc 10 w)) duv_y1 (caddr (assoc 10 w))
      duv_x2 (cadr (assoc 11 w)) duv_y2 (caddr (assoc 11 w)))
  (setq cep_kon (cond ((= cep_adi "G") min1)
                ((= cep_adi "K") max1)
                ((= cep_adi "D") max1)
                ((= cep_adi "B") min1)))
```

```
  (setq kor_kon (cond ((= cep_adi "G") duv_y1)
                      ((= cep_adi "K") duv_y1)
                      ((= cep_adi "D") duv_x1)
                      ((= cep_adi "B") duv_x1)))
  (if (or (= cep_adi "G") (= cep_adi "K")) (setq kor_kon1 duv_x1
                                                 kor_kon2 duv_x2)
                                           (setq kor_kon1 duv_y1
                                                 kor_kon2 duv_y2)
  )
  (if (= kor_kon cep_kon)
   (if (or (and (>= kor_kon1 maxk) (>= kor_kon2 maxk))
           (and (<= kor_kon1 mink) (<= kor_kon2 mink)))
     (progn
      (ssadd ent_name sec_duv3)
      (if (or (= cep_adi "G") (= cep_adi "K")) (setq lis_duv2 (append lis_duv2 (list duv_x1
duv_x2)))
                                               (setq lis_duv2 (append lis_duv2 (list duv_y1 duv_y2))))
     )
    )
    (progn
     (ssadd ent_name sec_duv21)
     (if (or (= cep_adi "G") (= cep_adi "K")) (setq lis_duv1 (append lis_duv1 (list duv_y1
duv_y2)))
                                              (setq lis_duv1 (append lis_duv1 (list duv_x1 duv_x2))))
    )
  )
  (setq don (1+ don))
 )
 (setq sec_duv2 sec_duv21)
)

(defun ciz()
 (setvar "blipmode" 0)
 (if (= kat_adi "Z") (setq kat_yuko 0) (setq kat_yuko nor_alt don 1))
 (setq dis_p1 (list min2 kat_yuko) dis_p2 (list min2 (+ kat_yuk kat_yuko))
     dis_p3 (list max2 kat_yuko) dis_p4 (list max2 (+ kat_yuk kat_yuko)))
 (if (/= kat_adi "C")
  (progn
```

```
(command "line" dis_p1 dis_p2 "")
(command "line" dis_p3 dis_p4 "")
(if (= 1 cativar) (cati_ciz))
(setq num 1)
(repeat (/ (- (length lis_duv2) 2) 2)
 (setq p1 (list (nth num lis_duv2) (+ kat_yuko par_yuk))
       p2 (list (nth (1+ num) lis_duv2) (+ kat_yuko par_yuk))
       p3 (list (car p2) (+ (cadr p2) pen_yuk))
       p4 (list (car p1) (+ (cadr p1) pen_yuk)))
 (if (and (or (= (car lis_kap2) (nth num lis_duv2))
              (= (car lis_kap2) (nth (1+ num) lis_duv2))
          )
          (= kat_adi "Z")
     )
     (setq p1 (list (car p1) (- (cadr p1) par_yuk))
           p2 (list (car p2) (- (cadr p2) par_yuk))
           kapi_kon 1)
 )
 (setq p1_kon (cond ((= cep_adi "G") (car p1))
                    ((= cep_adi "K") (car p1))
                    ((= cep_adi "D") (car p1))
                    ((= cep_adi "B") (car p1))))
 (setq p2_kon (cond ((= cep_adi "G") (car p2))
                    ((= cep_adi "K") (car p2))
                    ((= cep_adi "D") (car p2))
                    ((= cep_adi "B") (car p2))))
 (if (or (< p1_kon mink) (< p2_kon mink) (> p1_kon maxk) (> p2_kon maxk))
  (progn
   (command "pline" p1 p4 p3 p2 (if (= kapi_kon 1) "" "c"))
   (if (= kapi_kon 1) (kapi) (pen))
  )
 )
 (setq num (+ 2 num))
 (setq kapi_kon 0)
 (setq pen_num (1+ pen_num))
)
(if (= kat_adi "Z") (setq kat_yuko kat_yuk))
)
```

```
    (cati_ciz)
  )
)

(defun cati_ciz()
  (if (= kat_adi "N") (setq cat_alt (+ nor_alt (* kat_ade kat_yuk))))
  (if (= kat_adi "Z") (setq cat_alt kat_yuk))
  (setq sac_alt cat_alt)
  (setq c_p1 (list (+ max2 sac_gen) sac_alt)
        c_p2 (list (- min2 sac_gen) sac_alt)
        c_p3 (list (car c_p2) (+ (cadr c_p2) cat_yuk))
        c_p4 (list (car c_p1) (+ (cadr c_p1) cat_yuk))
  )
  (if (= mink nil)
    (command "pline" c_p1 "w" 0 0 c_p2 c_p3 c_p4 "c")
    (progn
      (if (< min2 mink) (progn
                          (setq c_p1 (list (- mink sac_gen 5) (cadr c_p1))
                              c_p4 (list (- mink sac_gen 5) (cadr c_p4)))
                          (command "pline" c_p1 "w" 0 0 c_p2 c_p3 c_p4 "")
                        )
      )
      (if (> max2 maxk) (progn
                          (setq c_p1 (list (+ max2 sac_gen) sac_alt)
                              c_p2 (list (+ maxk sac_gen 5) (cadr c_p1))
                              c_p3 (list (+ maxk sac_gen 5) (cadr c_p4))
                              c_p4 (list (car c_p1) (+ (cadr c_p1) cat_yuk)))
                          (command "pline" c_p2 "w" 0 0 c_p1 c_p4 c_p3 "")
                        )
      )
    )
  )
)

(defun maxmin1 ()
  (setq max1 (apply 'max lis_duv1) min1 (apply 'min lis_duv1))
)
```

```
(defun maxmin2 ()
  (setq max2 (apply 'max lis_duv2) min2 (apply 'min lis_duv2))
)

(defun maxmink ()
  (setq maxk max2 mink min2)
)

(defun maxmink2 ()
  (if (> max2 maxk) (setq maxk max2))
  (if (< min2 mink) (setq mink min2))
)

(defun sirala ()
   (setq kon_lis_duv2 nil sir_lis_duv2 nil)
   (setq num 0)
   (maxmin2)
   (while (> (length lis_duv2) 0)
     (repeat (length lis_duv2)
        (setq kontrol (nth num lis_duv2))
        (if (/= min2 kontrol) (setq kon_lis_duv2 (append kon_lis_duv2 (list kontrol)))
                              (setq sir_lis_duv2 (append sir_lis_duv2 (list kontrol)))
        )
        (setq num (1+ num))
     )
     (setq num 0 lis_duv2 kon_lis_duv2 kon_lis_duv2 nil)
     (maxmin2)
   )
   (setq lis_duv2 sir_lis_duv2)
)

(defun pen_dia ()
  (setq
       cephe_adi "G"
       c_justif "0"
       t_style "0"
       t_yuksek 50.0
```

```
    t_aci 0.0
    kat_adi "Z"
    olcek 50
    kat_yuk 270
    kat_ade 1
    par_yuk 90.0
    pen_yuk 120.0
    cep_adi "G"
    pen_tip "TIP1"
    dog_kal 5
    bol_say 2
    esit "e"
    cat_yuk par_yuk
    sac_gen 50
    ac_kan ""
)
(if (not nor_alt) (setq nor_alt kat_yuk))
(if (/= kat_yuko nil) (setq nor_alt kat_yuko))
(setq dia_cep 5)
(setq id (load_dialog "cephe.dcl"))
(while (< 1 dia_cep)
  (new_dialog "cephe" id)
  (set_tile "ddkat_yuk" "270.0")
  (set_tile "ddpar_yuk" "90.0")
  (set_tile "ddpen_yuk" "120.0")
  (set_tile "ddnor_alt" (rtos nor_alt 2 2))
  (set_tile "ddkat_ade" "1")
  (mode_tile "ddkat_ade" 1)
  (mode_tile "ddnor_alt" 1)
  (mode_tile "dd_isaret" 1)
  (action_tile "dd_cephe" "(cephe_bilgi)")
  (action_tile "ddkat_yuk"
    "(setq kat_yuk (atoi (get_tile \"ddkat_yuk\")))")
  (action_tile "ddpar_yuk"
    "(setq par_yuk (atoi (get_tile \"ddpar_yuk\")))")
  (action_tile "ddpen_yuk"
    "(setq pen_yuk (atoi (get_tile \"ddpen_yuk\")))")
  (action_tile "ddkat_zem"
```

```
  "(setq kat_adi \"Z\") (mode_tile \"ddkat_ade\" 1) (mode_tile \"ddnor_alt\" 1)
   (mode_tile \"dd_isaret\" 1) ")
(action_tile "ddkat_nor"
  "(setq kat_adi \"N\") (mode_tile \"ddkat_ade\" 0) (mode_tile \"ddnor_alt\" 0)
   (mode_tile \"dd_isaret\" 0) (setq kat_yuko nor_alt)")
(action_tile "ddkat_cat" "(setq kat_adi \"C\" cativar 1) (cati_dia)")
(action_tile "ddcat_var"
   "(setq cativar (atoi (get_tile \"ddcat_var\")))")
(action_tile "ddolcek_50" "(setq olcek 50)")
(action_tile "ddolcek_100" "(setq olcek 100)")
(action_tile "ddolcek_200" "(setq olcek 200)")
(action_tile "ddkat_ade"
   "(setq kat_ade (atoi (get_tile \"ddkat_ade\")))")
(action_tile "ddnor_alt"
   "(setq nor_alt (atoi (get_tile \"ddnor_alt\")))")
(action_tile "dd_isaret" "(done_dialog 2)")
(action_tile "ddcep_guney"
  "(setq cep_adi \"G\")")
(action_tile "ddcep_kuzey"
  "(setq cep_adi \"K\")")
(action_tile "ddcep_dogu"
  "(setq cep_adi \"D\")")
(action_tile "ddcep_bati"
  "(setq cep_adi \"B\")")
(action_tile "dd_pencere" "(blok_dia)")
(action_tile "dd_cati" "(cati_dia)")
(action_tile "cancel" "(done_dialog) (setq err \"\") (exit)")
(action_tile "accept"
  (strcat
    "(progn (setq kat_yuk (atof (get_tile \"ddkat_yuk\")))"
    "(setq par_yuk (atof (get_tile \"ddpar_yuk\")))"
    "(setq pen_yuk (atof (get_tile \"ddpen_yuk\")))"
    "(done_dialog))"
  )
)
(setq dia_cep (start_dialog))
(cond
  ((= dia_cep 2)
```

```
      (initget 1)
      (setq nor_alt (cadr (getpoint "1. Normal Kat Alt Kotu: ")))
    )
   )
  )
)

(defun cephe_bilgi ()
  (if (not (new_dialog "c_bilgi" id)) (exit))
  (set_ulist)
  (set_ulist2)
  (set_ulist3)
  (setq cephe_adi (cond ((= cep_adi "G") "0")
                        ((= cep_adi "K") "1")
                        ((= cep_adi "D") "2")
                        ((= cep_adi "B") "3")
                  )
  )
  (num_style)
  (setq t_style c_stylenum)
  (yuk_text)
  (if (/= h_style 0) (mode_tile "t_yuksek" 1) (mode_tile "t_yuksek" 0))
  (if proje_adi (set_tile "ddproje_adi" proje_adi))
  (set_tile "ddcephe_adi" cephe_adi)
  (set_tile "tstyle" c_stylenum)
  (set_tile "t_yuksek" "50.00")
  (set_tile "t_aci" "0.00")
  (action_tile "ddcephe_adi" "(setq cephe_adi $value)")
  (action_tile "tstyle" "(setq t_style $value) (yuk_text)
                         (if (/= h_style 0) (mode_tile \"t_yuksek\" 1) (mode_tile \"t_yuksek\" 0))")
  (action_tile "cjustif" "(setq c_justif $value)")
  (action_tile "t_yuksek" "(setq t_yuksek (atof $value))")
  (action_tile "t_aci" "(setq t_aci (atof $value))")
  (action_tile "accept"
    (strcat
      "(progn (setq proje_adi (get_tile \"ddproje_adi\"))"
      "(done_dialog 0))"
    )
```

```
 )
 (start_dialog)
)

(defun set_ulist()
 (setq ulist (list "Guney Cephesi"  "Kuzey Cephesi"    "Dogu Cephesi"
                   "Bati Cephesi"   "On Cephe"         "Arka Cephe"
                   "Sag Cephe"      "Sol Cephe"
             )
 )
 (start_list "ddcephe_adi")
 (mapcar 'add_list ulist)
 (end_list)
 (set_tile "ddcephe_adi" cephe_adi)
)

(defun set_ulist2()
 (setq ulist2 (list "Left"     "Align"     "Fit"       "Center"
                    "Middle"   "Right"
                    "Top Left"      "Top Center"    "Top Right"
                    "Middle Left"   "Middle Center" "Middle Right"
                    "Bottom Left"   "Bottom Center" "Bottom Right"
              )
 )
 (start_list "cjustif")
 (mapcar 'add_list ulist2)
 (end_list)
 (set_tile "cjustif" c_justif)
)

(defun set_ulist3(/ n_style h_style)
 (setq n_style (cdr (assoc 2 (tblnext "style" t))))
 (while n_style
  (setq lis_style (cons n_style lis_style))
  (setq n_style (cdr (assoc 2 (tblnext "style"))))
 )
 (start_list "tstyle")
 (mapcar 'add_list lis_style)
```

```
  (end_list)
  (set_tile "tstyle" t_style)
)

(defun yuk_text ()
  (if lis_style (setvar "textstyle" (nth (atoi t_style) lis_style)))
  (setq style_name (tblnext "style" t))
  (while style_name
    (if (= (cdr (assoc 2 style_name)) (getvar "textstyle"))
      (setq h_style (cdr (assoc 40 style_name)))
    )
    (setq style_name (tblnext "style"))
  )
)

(defun num_style ()
  (setq don_style 0)
  (repeat (length lis_style)
    (if (= (nth don_style lis_style) (getvar "textstyle"))
      (setq c_stylenum (itoa don_style))
    )
    (setq don_style (1+ don_style))
  )
)

(defun blok_dia ()
  (if (not (new_dialog "blok" id)) (exit))
  (set_tile "ddpen_tip" "TIP1")
  (set_tile "dddog_kal" "5.0")
  (set_tile "ddbol_say" "2")
  (action_tile "ddesit" "(setq esit \"e\")")
  (action_tile "ddfarkli" "(setq esit \"f\")")
  (action_tile "cancel" "(done_dialog 0) (setq err \"\")")
  (action_tile "accept"
    (strcat
      "(progn (setq dog_kal (atof (get_tile \"dddog_kal\")))"
      "(setq bol_say (atoi (get_tile \"ddbol_say\")))"
      "(setq ac_kan (get_tile \"ddac_kan\"))"
```

```
    "(done_dialog 0))"
   )
  )
  (start_dialog)
)

(defun cati_dia ()
  (setq cat_yuk par_yuk)
  (if (= kat_adio "Z") (setq nor_alt 0 kat_ade 1 kat_yuk kat_yuko2))
  (if (= kat_adio "N") (setq nor_alt (atoi (get_tile "ddnor_alt"))
                             kat_ade kat_adeo kat_yuk kat_yuko2))
  (if (not (new_dialog "cati" id)) (exit))
  (set_tile "ddcat_yuk" (rtos par_yuk 2 2))
  (set_tile "ddsac_gen" "50.0")
  (set_tile "ddcat_alt" (rtos (+ nor_alt (* kat_ade kat_yuk)) 2 2))
  (action_tile "ddcat_yuk"
    "(setq cat_yuk (atoi (get_tile \"ddcat_yuk\")))")
  (action_tile "ddsac_gen"
    "(setq sac_gen (atof (get_tile \"ddsac_gen\")))")
  (action_tile "ddcat_alt"
    "(setq cat_alt (atof (get_tile \"ddcat_alt\")))")
  (action_tile "cancel" "(done_dialog 0) (setq err \"\")")
  (action_tile "accept"
    (strcat
     "(progn (setq cat_alt (atof (get_tile \"ddcat_alt\")))"
     "(done_dialog 0))"
    )
  )
  (start_dialog)
)

(defun pen_err (msg)
 (setq *error* olderr)
  (if (not err)
    (princ (strcat "\nHatali bilgi: " msg))
    (princ err)
  )
  (if sblip (setvar "blipmode" sblip))
```

```
 (if scmde (setvar "cmdecho" scmde))
 (princ)
)

(Defun c:cephe(/ lis_style lis_styleh)
 (setq olderr *error*
       *error* pen_err
       sblip nil
       scmde nil
       err nil
 )
 (setq pen_num 1)
 (pen_dia)
 (if (= olcek 200) (setq bol_say (- bol_say 1)))
 (setq cmd_ech (getvar "cmdecho"))
 (setq blp_mod (getvar "blipmode"))
 (setvar "cmdecho" 0)
 (setvar "blipmode" 1)
 (sec)
 (sec2)
 (if (= kat_adi "Z") (zemin))
 (if lis_duv2 (progn
              (sirala)
              (maxmin2)
              (ciz)
              (maxmink)
             )
 )
 (while lis_duv1
   (sec2)
   (if lis_duv2 (progn
                (sirala)
                (maxmin2)
                (ciz)
                (maxmink2)
               )
   )
 )
```

```
(if (= kat_adi "Z")
 (progn
  (command "pline" (list (- mink 100) -5) "w" 10 10
                   (list (+ maxk 100) -5) "w" 0 0 "")
 )
)
(if (and (or (= cep_adi "K") (= cep_adi "B"))
         (or (= kat_adi "Z") (= kat_adi "N")))
  (command "mirror" "w" (list mink (cadr dis_p1)) (list maxk (cadr dis_p4)) ""
           "mid" (list (- mink 100) -5) "@100<90" "y")
)
(if (and (or (= cep_adi "K") (= cep_adi "B")) (= 1 cativar))
    (command "mirror" "w" (list (- mink sac_gen) (cadr c_p1))
                          (list (+ maxk sac_gen) (cadr c_p4)) ""
             "mid" (list (- mink 100) -5) "@100<90" "y")
)
(setvar "blipmode" 0)
(if (and (= kat_adi "N") (> kat_ade 1))
   (command "array" "w" (list mink (cadr dis_p1)) (list maxk (cadr dis_p4))
            "" "r" kat_ade 1 kat_yuk))
(if (= kat_adi "Z")
 (progn
  (yuk_text)
   (setq cephe_adi (cond ((= (atoi cephe_adi) 4) "On Cephe")
                         ((= (atoi cephe_adi) 5) "Arka Cephe")
                         ((= (atoi cephe_adi) 6) "Sag Cephe")
                         ((= (atoi cephe_adi) 7) "Sol Cephe")
                   )
   )
  (if (not cephe_adi)
   (setq cephe_adi (cond ((= cep_adi "G") "Guney Cephesi")
                         ((= cep_adi "K") "Kuzey Cephesi")
                         ((= cep_adi "D") "Dogu Cephesi")
                         ((= cep_adi "B") "Bati Cephesi")
                   )
   )
  )
```

```
(command "text")
(cond ((= (atoi c_justif) 0) (command (list maxk -100)))
    ((= (atoi c_justif) 1) (setq tp1 (getpoint "\nFirst text line point: ")
                tp2 (getpoint tp1 "\nSecond text line point: "))
            (command "_j" "_a" tp1 tp2)
    )
    ((= (atoi c_justif) 2) (setq tp1 (getpoint "\nFirst text line point: ")
                tp2 (getpoint tp1 "\nSecond text line point: "))
            (command "_j" "_f" tp1 tp2)
    )
    ((= (atoi c_justif) 3) (command "_j" "_c" (list maxk -100)))
    ((= (atoi c_justif) 4) (command "_j" "_m" (list maxk -100)))
    ((= (atoi c_justif) 5) (command "_j" "_r" (list maxk -100)))
    ((= (atoi c_justif) 6) (command "_j" "_tl" (list maxk -100)))
    ((= (atoi c_justif) 7) (command "_j" "_tc" (list maxk -100)))
    ((= (atoi c_justif) 8) (command "_j" "_tr" (list maxk -100)))
    ((= (atoi c_justif) 9) (command "_j" "_ml" (list maxk -100)))
    ((= (atoi c_justif) 10) (command "_j" "_mc" (list maxk -100)))
    ((= (atoi c_justif) 11) (command "_j" "_mr" (list maxk -100)))
    ((= (atoi c_justif) 12) (command "_j" "_bl" (list maxk -100)))
    ((= (atoi c_justif) 13) (command "_j" "_bc" (list maxk -100)))
    ((= (atoi c_justif) 14) (command "_j" "_br" (list maxk -100)))
)
(if (= (atoi c_justif) 1)
    (command proje_adi)
    (if (= (atoi c_justif) 2)
        (if proje_adi (if (= h_style 0)
                    (command t_yuksek proje_adi)
                    (command proje_adi)
                )
                (if (= h_style 0)
                    (command "" "")
                    (command "")
                )
        )
        (if proje_adi (if (= h_style 0)
                    (command t_yuksek t_aci proje_adi)
                    (command t_aci proje_adi)
```

```
                    )
                    (if (= h_style 0)
                       (command "" "" "")
                       (command "" "")
                    )
                )
            )
        )
        (command "text" "" cephe_adi)
    )
)
(setvar "cmdecho" cmd_ech)
(setvar "blipmode" blp_mod)
(setq max2 nil min2 nil maxk nil mink nil ;cephe_adi nil
    cativar nil kat_adio kat_adi kat_yuko2 kat_yuk kat_adeo kat_ade)
)
```

Ek B. PROGRAMLANABİLİR DİALOG KUTUSU (DCL) METNİ

Geliştirilen AutoLISP programı ile birlikte kullanılan diyalog kutusu metni aşağıda verilmektedir.

```
cephe:dialog {
 label = "Cephe Cizimi";
 :row {
  :button {
   label ="Cephe Bilgileri...";
   key = "dd_cephe";
   mnemonic = "h";
   width=2;
  }
 }
 : boxed_radio_row {
  :column {
   label = "Kat adi    ";
    : radio_button {
      key = "ddkat_zem";
      label = "Zemin kat";
      mnemonic = "Z";
      width=12;
      value=1;
    }
    : radio_button {
      key= "ddkat_nor";
      label = "Normal kat";
      mnemonic = "N";
      width=10;
    }
     : radio_button {
      key = "ddkat_cat";
      label = "Cati kat";
      mnemonic = "C";
      width=10;
```

```
    }
  }
  :column {
    label = "Cati";
    : toggle {
      label = "Var";
      key= "ddcat_var";
      mnemonic = "V";
      width=10;
    }
  }
  :column {
    label = "Olcek:1/ ";
    :radio_button {
      key = "ddolcek_50";
      label = "50";
      mnemonic = "5";
      value=1;
    }
    :radio_button {
      key = "ddolcek_100";
      label = "100";
      mnemonic = "1";
    }
    :radio_button {
      key = "ddolcek_200";
      label = "200";
      mnemonic = "2";
    }
  }
}
: edit_box {
  key = "ddkat_ade";
  label = "Kat adedi:";
  mnemonic = "a";
  edit_width = 18.2;
}
```

```
: edit_box {
   key = "ddkat_yuk";
   label = "Kat yuksekligi:";
   mnemonic = "y";
   edit_width = 18.2;
}
: edit_box {
   key = "ddpar_yuk";
   label = "Parapet yuksekligi:";
   mnemonic = "r";
   edit_width = 18.2;
}
: edit_box {
   key = "ddpen_yuk";
   label = "Pencere yuksekligi:";
   mnemonic = "e";
   edit_width = 18.2;
}
:row {
  : edit_box {
     key = "ddnor_alt";
     label = "1.Normal kat alt kotu:";
     mnemonic = "t";
     edit_width = 6;
  }
  : button {
      label ="Isaretle <";
      key = "dd_isaret";
      mnemonic = "s";
  }
}

: boxed_radio_row {
  label = "Cephe adi";
  : radio_button {
     label = "Guney";
     key = "ddcep_guney";
     mnemonic = "G";
```

```
    value=1;
  }
  : radio_button {
    label = "Kuzey";
    key = "ddcep_kuzey";
    mnemonic = "K";
  }
  : radio_button {
    label = "Dogu";
    key = "ddcep_dogu";
    mnemonic = "D";
  }
  : radio_button {
    label = "Bati";
    key = "ddcep_bati" ;
    mnemonic = "B";
  }
}
:row {
:row {
  :button {
    label ="Pencere...";
    key = "dd_pencere";
    mnemonic = "P";
    edit_width = 10;
  }
}
:row {
  :button {
    label =" Cati...  ";
    key = "dd_cati";
    mnemonic = "i";
  }
}
}
: row {
  :spacer { width=1;}
:button {
```

```
  label ="Ok";
     key = "accept";
     width=8;
     fixed_width=true;
   }
   :button {
     label ="Cancel";
     is_cancel =true;
     key = "cancel";
     width=8;
     fixed_width=true;
   }
   :spacer { width=1;}
  }
}
blok:dialog {
  label = "Pencere";
  : button {
     key = "ddpen_tip";
     label = "Pencere tipi...";
     mnemonic = "P";
  }
  spacer_0;
  :column {
    : edit_box {
      key= "dddog_kal";
      label = "Dograma kalinligi";
      mnemonic = "D";
    }
    : edit_box {
      key= "ddbol_say";
      label = "Pencere bolme sayisi";
      mnemonic = "e";
    }
  }
  spacer_0;
  : boxed_radio_column {
      label = "Pencere bolme araliklari";
```

```
      :radio_button {
      key= "ddesit";
      label = "Esit";
      mnemonic = "s";
      value = 1;
    }
    :radio_button {
      key= "ddfarkli";
      label = "Farkli";
      mnemonic = "F";
    }
  }
  : row {
    : edit_box {
      key = "ddac_kan";
      label = "Acilan kanatlar";
      mnemonic = "c";
    }
  }
// ok_cancel;
  : row {
    :spacer { width=1;}
  :button {
  label ="Ok";
      key = "accept";
      width=8;
      fixed_width=true;
    }
    :button {
      label ="Cancel";
      is_cancel =true;
      key = "cancel";
      width=8;
      fixed_width=true;
    }
    :spacer { width=1;}
  }
}
```

```
cati:dialog {
 label = "Cati";
  : edit_box {
    key= "ddcat_yuk";
    label = "Cati parapet yuksekligi:";
    mnemonic = "C";
    edit_width = 19.2;
  }
  : edit_box {
    key= "ddsac_gen";
    label = "Sacak genisligi:";
    mnemonic = "a";
    edit_width = 19.2;
  }
 :row {
  : edit_box {
     key = "ddcat_alt";
     label = "Cati parapeti alt kotu:";
     mnemonic = "t";
     edit_width = 6;
  }
  : button {
     label ="Isaretle <";
     key = "dd_isaret2";
     mnemonic = "s";
  }
 }
// ok_cancel;
 : row {
  :spacer { width=1;}
 :button {
 label ="Ok";
    key = "accept";
    width=8;
    fixed_width=true;
  }
```

```
    :button {
      label ="Cancel";
      is_cancel =true;
      key = "cancel";
      width=8;
      fixed_width=true;
    }
    :spacer { width=1;}
  }
}
c_bilgi:dialog {
  label = "Cephe Bilgileri";
  :column {
    : edit_box {
      key= "ddproje_adi";
      label = "Proje adi:";
      mnemonic = "P";
    }
    spacer;
    : popup_list {
      label = "Cephe adi:";
      key= "ddcephe_adi";
      mnemonic = "C";
      width=20;
    }
  }
  : boxed_column {
    label = /*MSG27*/"Yazi ayarlamalari";
    fixed_width = true;
    width = 40;
    : column {
       : popup_list {
          label = /*MSG28*/"Ayar:";
          key = "cjustif";
          edit_width = 20;
          mnemonic = /*MSG29*/"A";
       }
```

```
        : popup_list {
            label = /*MSG30*/"Yazi stili:";
            key = "tstyle";
            edit_width = 20;
            mnemonic = /*MSG31*/"s";
        }
    }
    : row {
        : button {
            label = /*MSG32*/"Yukseklik <";
            key = "p_yuksek";
            mnemonic = /*MSG33*/"Y";
            width = 15;
            fixed_width = true;
        }
        : edit_box {
            key = "t_yuksek";
            edit_width = 20;
            fixed_width = true;
        }
    }
    : row {
        : button {
            label = /*MSG34*/"Aci <";
            key = "p_aci";
            mnemonic = /*MSG35*/"i";
            width = 15;
            fixed_width = true;
        }
        : edit_box {
            key = "t_aci";
            edit_width = 20;
            fixed_width = true;
        }
    }
  }
// ok_cancel;
```

```
ow {
spacer { width=1;}
button {
 label ="Ok";
 key = "accept";
 width=8;
 fixed_width=true;
}
button {
 label ="Cancel";
 is_cancel =true;
 key = "cancel";
 width=8;
 fixed_width=true;
}
spacer { width=1;}

}
```

ÖZGEÇMİŞ

1969 yılında Gaziantep'de doğdu. İlk öğrenimini Almanyada tamamladı. Lise eğitimini 1985 yılında İstanbul Mecidiyeköy Lisesinde bitirdi. Aynı yıl İ.T.Ü. Mimarlık Fakültesi Mimarlık Bölümüne girdi. Mimarlık eğitiminin 3. yılı 2. döneminden itibaren Bilgisayar konusundaki çalışmalarını hızlandırarak mimari proje ödevlerini Bilgisayar Destekli Tasarım konusundaki yazılımlardan AutoCAD 2.6 ile hazırladı. Bundan sonraki tüm mimari proje ödevleri ile bitirme ödevini AutoCAD ile yaparak 1990-91 öğretim dönemi kış yarıyılında mimar olarak mezun oldu. Bu dönemler içerisinde AutoCAD 9,10 ve 11'i kullandı. Bunların dışında Dbase III Plus, Lotus 123 gibi paket programların eğitimlerini aldı. Yüksek Lisans eğitimini 1995 yılı Şubat ayında tamamlayarak mezun oldu. 1989 yılından beri İstanbul Gayrettepede İLTAY Mimarlık Dekorasyon ve İnşaat Limited Şirketi'nde çalışmaktadır. 1990 yılında T.M.M.O.B. Mimarlar Odası Büyükkent Şubesinde AutoCAD eğitmeni olarak başladığı göreve halen devam etmektedir.