

PREFACE

Developing an intelligent design supporter that generates alternatives to the furnishing process was a marvelous discovery for me because I found a viable use for design process which could also be immediately beneficial to space planning in architectural design. It would be further beneficial to different applications of interior architecture generating reasonable alternatives for the other spaces used for different facilities. Moreover, the setup of this furnishing agent could be easily introduced into architectural and furnishing firms as it does not require expensive and space taking hardware devices. Also, it would be easily applied to different products of these firms since it is not difficult to replace their products with the ones currently used in the system, with some adding to the codes written in C++ Builder environment.

Now that the potential for using computerized tools and the other computer aided design approaches in different architectural applications is inevitable, it becomes more important to verify that this tool successfully represents living space and eliminates the time consuming phases of furnishing. Moreover, perception of furnishing space is the same in both real and simulated environments when the correct parameters are given to the system within the concept of the developed software. The task of the program is to find and demonstrate convenient solutions for the application in order to prevent the wrong applications of furnishing and also time consuming phases of space planning. The hope is that the findings of this study will guide the both expert and non-expert users during the design activity and accelerate the furnishing process.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my committee members, Assoc. Prof. Sinan Mert Şener, Chairman, Prof.Dr. Hasan Şener, Dean of Architectural Faculty and Prof.Dr. Muhittin Gökmen, Dean of Electric and Electronic Faculty. I would also like to extend my special thanks to Computer Science Engineer Mustafa Aydın, for his support in design and implementation of the software and and Joseph Slanina for his superb editing. And finally, I want to thank the two people who were the most supportive and patient during this process, Hikmet Okçu and Ayşe Okçu.

May 2003

Selen Okçu – Architect

TABLE OF CONTENTS

	<u>Page</u>
LIST OF FIGURES	vi
LIST OF TABLES	viii
SUMMARY	ix
ÖZET	xi
1. INTRODUCTION	1
1.1. Goals	2
1.2. Scope of Issues on Furnishing Agent	3
1.3. Methods	4
1.4. Conclusion	5
2. DEFINITION OF DESIGN PROCESS	7
2.1 The Nature of Architectural Design and Interior Design Process	7
2.2 A Design Process Leading from Home Planning to Furnishing	8
2.3 The Need for Computational Tools for Design Processes	9
3. COMPUTER AIDED DESIGN	10
3.1 Overview of Computerized Design Aids and CAD	10
3.2 Evolution of Computer Aided Design Approach	11
3.3 Evaluation of CAD, CAAD, AI	12
3.4 Expert systems	13
3.5 Conclusions	15
4. ARTIFICIAL INTELLIGENCE IN DESIGN	16
4.1 Understanding of How AI Works	16
4.2 Possibilities of AI	17
4.3 Various Programming Languages Used in AI	18
4.4 Evaluation of the Programming Languages	19
4.5 Introduction to C++ Builder	21
4.6 Proposed System Architecture	28
4.7 Conclusions	30
5. PROJECT DESCRIPTIONS	31
5.1 Implementation	31
5.2 Limitations	66
5.3 Consequences of This Alternative Generating Approach	67
5.4 Results, Applications and Impacts	68

6. GENERAL RESULTS	69
REFERENCES	71
APPENDIX-A	75
WRONG APPLICATIONS OF FURNISHING	
APPENDIX-B	80
FLOW CHART	
CURRICULUM VIATE	82

LIST OF FIGURES

	<u>Page</u>
Figure 4.1. An overall architecture of the system codes written in C++ Builder environment	22
Figure 4.2. Component Resource Menu Options	23
Figure 4.3. The components on the various pages of the Component Palette	23
Figure 4.4. An overall structure of the system objects.	24
Figure 4.5. (a) Project Menu, (b) Run Menu	25
Figure 4.6. (a) View Menu, (b) Tools Menu	26
Figure 4.7. An example of a design composition	28
Figure 5.1. ‘Select shape of the room’ tab	33
Figure 5.2. (a) Edit box used to edit sizes of a rectangle room (b) Edit box used to edit the size of a square room	33
Figure 5.3. (a) Combo box used to choose furniture (b), (c) Explanations of the edited numbers of the chosen furniture	34
Figure 5.4. (a) ‘Choose furniture’ combo box showing the selection of sofa(3) (b), (c) Definition of edited numbers of sofa(2) and sofa(1).....	34
Figure 5.5. ‘Choose furniture’ combo box showing the selection of desktop.....	35
Figure 5.6. (a) ‘Choose furniture’ box showing the selection of eating table	35
(b) ‘Select the type of the eating table’ tab (c) The list of the selected movables	
Figure 5.7. (a), (b), (c) Error tabs	36
Figure 5.8. ‘Add door’ button	36
Figure 5.9. (a), (b), (c) Explanations of needed data for door selection	36
Figure 5.10. (a) Explanation of needed data for window selection (b) Demonstration of a window generated with the given parameters ...	37
Figure 5.11. (a) Explanation of needed data for serving window (b) Demonstration of a serving window	38
Figure 5.12. ‘Arrange’ button	38
Figure 5.13. ‘Calculate solutions’ button	39

Figure 5.14.	(a) Data of a rectangle room, (b) Data of a square room.....	39
Figure 5.15.	(a) Showing the two tabs used in defining the frequency of alternatives (b) Selectable distance options between the selected objects (c) Selectable distance options used by ‘generator module’	40
Figure 5.16.	(a), (b), (c), (d), (e) Examples of furnishing design compositions	42
Figure 5.17.	(a), (b), (c), (d), (e) Rotated alternatives of compositions	42
Figure 5.18.	(a) Image of not-rotated eating table for six people (b) Rotated alternative of eating table	42
Figure 5.19.	(a) ‘View alternatives’ button (b) Two possible icons used in demonstrating alternatives (c) Edit box, used in demonstrating the edited alternatives	43
Figure 5.20.	(a) Combo box showing the satisfying results according to the user (b) Furnishing composition produced while exploring the design space (c) Number of generated alternatives	44
Figure 5.21.	(a) Tab containing all buttons used while viewing and saving alternatives (b) ‘Save file’ window	45
Figure 5.22.	‘Load from file’ button.....	45
Figure 5.23.	‘Send to Auto- CAD’ button	46
Figure 5.24.	Simulation of an alternative in CAD environment.....	46
Figure 5.25.	Data of all alternatives generated, in text format.....	47
Figure 5.26.	‘New Room’ button	48
Figure 5.27.	Demonstration of all selectable movables within the program	49
Figure 5.28.	Various alternatives of mask compositions.....	49
Figure 5.29.	(a), (b) ‘Open’ window used to apply the selected mask	50
Figure 5.30.	Flow chart	52
Figure 5.31.	(a), (b) Showing the possible horizontal locations for eating table (c), (d) Showing the possible vertical locations for eating table.....	60

LIST OF TABLES

	<u>Page</u>
Table 5.1. Data of an alternative in a text file.....	46
Table 5.2. Sample from the codes calculating the area of the sitting group.....	53
Table 5.3. Sample from the codes; run when eating table is selected	54
Table 5.4. Sample from the codes, run in the control of meeting points.....	55
Table 5.5. Sample from the codes, run after the ‘Approve’ button.....	56
Table 5.6. Sample from the codes; run while calculating the empty space	57
Table 5.7. Sample from the codes, run to explore suitable locations on the edge AB for sitting group	58
Table 5.8. Sample from the codes, run to explore suitable locations for the desktop	59
Table 5.9. Definition of the horizontal edges of the room	60
Table 5.10. Definition of the vertical edges of the room	61
Table 5.11. Sample from the codes, used while searching for different location alternatives for eating table	61
Table 5.12. Sample from the codes, run while viewing the results.....	63
Table 5.13. Sample from the codes; run while classifying data of each object	64
Table 5.14. Sample from the codes; run while saving data of each object	65

SOFTWARE DEVELOPED FOR LIVING ROOM FURNISHING SUMMARY

In this study, a soft-ware was developed for furnishing process of architectural design. The living room was chosen as a pilot space and the system is programmed to generate reasonable compositions within the principals of spatial orientation.

In the first part, the subject of the thesis, objectives, scope of the issues analyzed within the project, method of the study and the need for computer aided design soft-wares are explained.

In the second part, design problems, main concerns of architectural design and interior design; furnishing in terms of space organization are clarified. The importance of computer-based systems in design applications is mentioned.

The structure of DDSS (design decision support systems) is explained in details in the third part. Computer usage that is inevitable in architectural design in recent years is examined and the definitions of computer aided architectural design, artificial intelligence (AI), expert systems (ES) are made.

In the forth part, contents of AI, possibilities offered by AI and various programming languages used in AI applications are explained. In addition evaluations of some of these programs are made in order to identify their traits. Characteristic features and contents of C++ Builder environment are described; especially the elements that form graphical user interfaces and its object-oriented fashion are explained. A detailed explanation of the importance of the software developed for users and the components of the system structure conclude the fourth part.

In the fifth part, all the icons on the user interface of the software and the steps that the software engine follows while generating alternatives are explained in detail. All the characteristics of the program and feasibilities offered by the program are defined. Explanations of the different application areas and the aim of the software

are mentioned according to their importance to this process. An evaluation of the results of this software application concludes the fifth part.

In the sixth and final part, general results of this master thesis are explained. Future objectives, possible additions to the software and different application areas based on these software modifications are discussed.

OTURMA ODASI TEFRİŞİ İÇİN GELİŞTİRİLEN BİR YAZILIM

ÖZET

Bu çalışmada mimari tasarımın tefriş aşaması için bir yazılım geliştirilmiştir. Oturma odası kılavuz alan olarak seçilmiş ve sistem mekansal yönelişin prensipleri çerçevesinde, uygun çözümler üretmek için programlanmıştır.

Birinci bölümde tezin konusu, hedefler, proje ile sorgulanan konuların kapsamı, çalışmada izlenen metot, bilgisayar destekli tasarım programlarına olan ihtiyaç anlatılmıştır.

İkinci bölümde tasarım problemi, mimari tasarım ve iç mekan tasarımının başlıca meselelerinin açıklamaları; tefriş aşamasının, mekansal organizasyon bağlamında tanımı yapılmıştır.

Üçüncü bölümde tasarım kararlarını destekleyen sistemlerin yapısı detaylı olarak ele alınmıştır. Son yıllarda mimari tasarımda kaçınılmaz olan bilgisayar kullanımı irdelenmiş ve bilgisayar destekli tasarım, yapay zeka, uzman sistemlerin tanımlamaları yapılmıştır.

Dördüncü bölümde yapay zekanın içeriği, yapay zeka ile sunulan olanaklar ve yapay zeka uygulamalarında kullanılan çeşitli yazılımlar açıklanmıştır. Bunlara ek olarak, bu yazılımların niteliklerinin saptanması için bazılarının değerlendirilmeleri yapılmıştır. C++ Builder ortamının karakteristik özellikleri ve içeriği anlatılmıştır; özellikle grafiksel kullanıcı arayüzünü oluşturan elemanları ve nesneye yönelik yapısı açıklanmıştır. Geliştirilen yazılımın kullanıcılar açısından önemi ve sistem yapısının bileşenleri detaylı olarak belirtilmiştir.

Beşinci bölümde yazılımın kullanıcı arayüzünde bulunan bütün ikonlar ve alternatiflerin üretiminde yazılım motorunun izlediği aşamalar detaylı olarak anlatılmıştır.. Programın bütün özellikleri ve programın sunduğu olanaklar belirtilmiştir. Bunlarla bağlantılı olarak, yazılımın amacı ve uygulama alanlarından

bahsedilmiştir. Beşinci bölümün sonunda bu proje ile elde edilen sonuçlar açıklanmıştır ve bu sonuçların değerlendirilmesi yapılmıştır.

Altıncı ve son bölümde, bu yüksek lisans tezi ile elde edilen genel sonuçlar açıklanmıştır. Gelecekteki hedefler, yazılımın olarak verdiği eklentiler ve bu eklentilere bağlı olarak yazılımın kolayca uyarlanabileceği değişik uygulama alanları ele alınmıştır.

1. INTRODUCTION

The process of going from conceptual design to detailed design involves solving complex dynamic problems. In addition, architectural design is more concerned with concepts, ideas, judgments and experiences than it is with numerical descriptions. The design process lends itself creating solutions that are fluid and emergent entities, generated by dynamic and situated designing activities. In designing, neither the stated goal nor the solution space is completely predetermined. The ability to provide useful designing support at the conceptual stages of designing to accommodate the situated and fluid nature of early schematic designing is essential. Such support creates the opportunity for conventional CAD systems to assist designers in exploring the solutions for space. It provides designers with useful and applicable design knowledge during the generation of design concepts and maintains the integrity of these concepts during different stages of designing.

An idea of building up a “furnishing agent”, which allows for interactive, visual simulation and experimentation of space organization during the process of architectural design, offers designers various alternate locations of selected objects in living room. Furnishing is mainly concerned with locating the equipment of user activities in a space so that it is a complement to the other movable and immovable equipment of the room. This is one of the most time consuming phases of architectural design and important for the design of qualified environments for tenants. Accordingly, furnishing focuses on the complex interactions between the equipment of the facilities and the careful analysis of their relations in order to produce reasonable solutions. As a result of these facts, use of an alternative generating system for furnishing is useful for designers to explore and perceive their designs differently.

1.1. Goals

The aim of this study is to explore a computerized tool for furnishing and develop an understanding of the computational processes that support design decision-making. This project focuses on the use of computers as design decision support systems which can be used to aid designers. It highlights the idea of information represented by symbols with processes which operate on that information in order to support decisions. Moreover, it draws its material from mathematically based structures of alternative generating system and optimization.

Another important objective of the system is to provide collaboration between the designer (expert and non-expert user) and the computer during the process of furnishing. This is accomplished by exploring the living room for various alternatives of furnishing compositions, preserving this collected data of convenient results in the memory and translating the data of satisfactory design solutions to CAD environment. This helps to demonstrate to the user the organization of the living space as it relates to user's interactions with the interface. This process determines which of the selected archetypal settings can be located in the room as well as taking into consideration both the physical values of the room and general principles of furnishing. Generating alternatives automatically with the given parameters, the system offers the user to choice between the most convenient ones among the generated alternatives. Besides integration with a conventional CAD system provides interactive support to expert users in furnishing.

Consequently, the home is a special environment in human's life and satisfactory design solutions should be created to promote their user's well being. In the process of furnishing the home environment, equipment should be professionally chosen as they may not fit into the room. The size of the room should be compared with the sizes of the furnishing equipment when deciding which equipment is the right choice. Different placement options of equipment in the room and different functional possibilities in the room should be thoroughly investigated beforehand in order to eliminate some of the chosen equipment. Another important issue is the layout of settings and the layout is defined by its spatial content, its form, its organization and its circulation. In other words, the effective use of space and the quality of

environment can be proved by satisfactory spatial orientation. In order to exemplify the compositions of selected merchandise within the requirements of spatial orientation, furnishing of a living room is chosen as a pilot project.

1.2 Scope of issues on furnishing agent

Merchandising is an important aspect for space organization and equipment needs to be carefully located and designed in order to provide the user expectations; furthermore, the equipped space must be appropriate for the purpose. The space where the human live should be designed with the interpretation of human expertise in order to prevent unexpected complications. This is especially true in a home environment where often time's users need to be guided in a different direction than the one they wish to follow. Otherwise, they become just the selectors of their merchandise without having any information about the layout of settings and space organization. Therefore, the concept of spatial problem solving should be illustrated by design projects or alternative generating systems supporting the process of planning a layout.

In a home environment, the living room can include a variety of facilities to meet different needs such as dining, working and sitting. If the physical values of the room, the area for circulation are considered appropriate through reasonable solutions, all of these functions can be located in a living room in order to allow residents to function more comfortably. In view of that, the most complex relations between the equipment in the home environment can be analyzed in the living room. Living room has the advantage of obtaining different knowledge related to different functions for this study. In addition, the software developed, as an outcome of this study, compiles the complex interactions between function and furnishing equipment, and analyzes these relations, and produces reasonable solutions for living room. This software presents various furnishing methods in exploring the convenient locations of equipment in a design composition. It allows designers to have a variety of representations of what can be designed within the given parameters. This possibility may lead them to different reasonable solutions to those they may otherwise have pursued. Finally, the system offers each designer, related to his interest from the generated design compositions, transferring data of satisfactory

design solutions to CAD environment. The designer can continue the detailed furnishing process in that environment.

1.3 Methods

The method used in this study was the development of a mathematically modeled design decision support system which is considered as a supporter while furnishing the living room. User expectations of comfort have been steadily rising due to changes in their living conditions. They expect their products to be adequately furnished within the reasonable standards of space organization. With the help of an intelligent alternative generating application, which provided them with a way to furnish their own conditions through acceptable design solutions for the chosen functions for the living room, they would be able to choose the most convenient results for them between the generated alternatives. Because the most important aspect of the user requirements of the indoor environment is to maintain comfort, it is important to analyze carefully the living room environment. In addition, it is necessary to understand the relationship between the chosen components. A questionnaire needs to be filled out concerning the user requirements before the system is structured. Finally, the user shouldn't be disappointed by the wrong size of the furniture bought from the catalogue.

For these reasons, in order to enhance the living conditions, maintain comfort of the room within the reasonable standards of furnishing and avoid the time consuming and challenging sides of furnishing/ such as consideration of the placement of functions and correlation between chosen component, etc., and the need for some controls within knowledge-based analysis like enough space for the chosen components, required circulation area/ this intelligent furnishing system is developed. This alternative generating system is also important for other required controls needed for the comfort analysis of furnishing, such as analyzing the relationship between the components and immovable equipment of the room like serving window, door, etc. By compiling the given data for each specific user, this alternative generating system attempts to increase the usefulness of the internal environment of living rooms to meet the user needs by offering many alternatives in real time. In addition to this, the visualization method and the searching engine

developed in this system make it possible to place furniture in a plan lay-out and see whether or not they fit into the room.

Consequently, to prevent the wrong applications of furnishing and support the reasonable furnishing applications within the parameters mentioned above; a new furnishing method working as an alternative generator has been developed. These wrong applications can be found in Appendix-A. Since the capacity of this system runs more efficiently in situations where the processing of movable and immovable equipment of the room is complex, the living room was chosen as the subject of the project. The system is programmed to search for the most reasonable locations of the selected components.

1.4 Conclusion

Computer technology has allowed home computer users to assume the creative role of professionals due to the potential of integrated software. Another important advantage of computer applications in design is the maintenance of the creative gap between the professional and the amateurs while it is empowering home users to become more involved in the creative process. Before the design activity begins, every user has an idea about what the space should look like and what equipment is desired. This process is very exciting and the creation of a unique room vision is very satisfying. Unfortunately, the next phase when they are trying to adapt their vision to reality can be very frustrating and end with unacceptable results. In other words, the desired planned layout in the mind, most times does not match up with acceptable design solutions. However, efficient, reasonable and spatially attractive space configurations can be designed through an effective partnership with the amateur and professional made possible by computer technology. Only after principals of spatial orientation have been evaluated professionally and that knowledge has been translated into applicable form can an effective design strategy be developed. Applicable form addresses the behavior of dynamic components. Moreover, this form can be simulated before the design activity with the use of computational tools.

As a final point, since the user's expectations will be simulated before furnishing by the help this application, the user's satisfaction with the proposed design will also be guaranteed. In other words, being able to use the general principles of that specific field with the tool called "alternative generating system" allows the users to review equipment in a simulation and make their own design choices. The use of this system ensures that furnishing applications are not only convenient but also can be concluded with satisfactory solutions in a short amount of time.

2. DEFINITION OF DESIGN PROCESSES

2.1 The Nature of Architectural Design and Interior Design Process

Architectural design is more concerned with concepts, ideas, judgment and experience much more than numerical descriptions. Each design component has implications for many other components. This process consists of problem definition, analysis, searching for reasonable alternatives, estimating and decision making. The architectural design field, like many other multidisciplinary areas is full of ill-structured problems in which social, political, economic and technical considerations are involved. First of all, one of the main concerns of this field is user satisfaction. The main requirement of occupant satisfaction is to create and maintain optimum life conditions. The most important aspects of these conditions are spatial and programmatic issues as well as the evaluation of the comfort sensation with the recommended existing standards. In different phases of the design process, the design parameters of the different disciplines mentioned above can be applied to the project by generation of spaces. This can be in terms of space relations according to general principles of field organization, analysis on spatial configurations, and physical measurements of variables involved in each aspect of the environment. Secondly, criteria like costs, structural and thermal behavior and many other aspects should be taken into account, preferably in the early stage of design and not just during analysis later in the process.

All these mentioned concerns also form the main issues of interior design as it is a branch of architectural design. If we summarize, the organization of multi spaces, establishment of hierarchy, development of space planning and furniture layout to establish programmatic and functional relationships; manipulation of space through the application of design elements, especially light, color, texture and materials, to enhance the desired ambiance and concept; consideration and design of all the components of the interiors, such as furniture, accessories, luminaries, etc. In addition to the architectural elements of the space, selection of appropriate interior

materials, finishes and products for the project and formation of a material board are the main issues of interior design. The main objective is to create an environment using the criteria mentioned above in order to provide its human occupants, an acceptable indoor living situation. The quality of indoor environment is of great importance for our lives and well being. Moreover, this optimum living situation can be defined as the condition of a mind in complete satisfaction with the environment in which it finds itself.

2.2 A Design Process Leading from Home Planning to Furnishing

In recent years special consideration has been given to space orientation of home planning. A home can be named basically as a shelter which means a place where a human meets the main needs of his life. In order to highlight the importance of user satisfaction and the general requirements of the user satisfaction in a home environment, several themes have been researched. Further approaches and ‘alternative’ design methods have been explored to provide the user needs in home environments. One of the main themes of this research was the organization of space including furnishing process.

The furnishing process is one of the most time consuming phases of design and contains complex interactions between all components of the selected room. The user should not be dissatisfied by the wrong size of the furniture bought in the furnishing space. It is also critical that the merchandising plan must be attractive, create awareness, interest, and measure up to expectations. This includes but is not limited to providing a suitable level of sophistication, strengthen the confidence in the standards of the premises and leave a memorable impression, related to furnishing space. The most time consuming part in this process is the organization of the components. This consists of generating and comparing different alternatives and ends when the user concerned reaches the desired destination through the reasonable alternatives in furnishing. By means of, lots of combinations for the location of components should be exemplified and the relations between each of them should be analyzed.

2.3 The Need for Computational Tools for Design Processes

In different phases of design, human beings may be capable of drawing conclusions, finding solutions and taking actions based on available information during the related processes of problem-solving and decision-making. However, the need for proper design supporting tools to handle complex problems is inevitable. For this reason, a computer-based system that behaves in such a fashion is referred to as an alternative agent that offers designers time saving advantages of various design applications.

3. COMPUTER AIDED DESIGN

3.1 Overview of Computerized Design Aids and CAD

Computerized Design Aids and CAD are computer systems that support design intelligence work through problems using a computer model of expert human reasoning. The structure of an intelligent design decision support system generally consists of: knowledge base/ knowledge acquisition/ module/ inference engine/ user interface and explanation of results. The knowledge base contains information from human experts in addition to facts provided by the user during interaction with the system. Most of today's computer aided design systems are well suited to creating geometric objects. Nevertheless, users find common tasks, such as quickly furnishing a room, hard to accomplish, especially since placing objects is conceptually different from creating them (Lesperance, Levesque, Reiter, 1999).

The use of computers in the design process is very essential in the context of providing communication between components, and data transfer between design phases. User expectations and complex interactions of design process can be well provided for by setting correct parameters automatically with the use of computers. More specifically, computer based problem solving techniques permit the user to generate quite easily design alternatives of a design problem that could be validated by regulatory authorities, without any professional input. Acceptable computer applications for design generally comprise of an interface between the user and a knowledge system that is built out of the experiences of expert practitioners in the given field.

The widespread use of CAD has become a kind of common knowledge today. Moreover, CAD (computer aided design) has become a major professional digital tool within the areas of drawing and design including engineering sciences. CAD-software consists often of several programs and sub-programs located in a system specific file structure. For instance AutoCAD really is a kind of "common CAD-platform", and the numerous engineering applications created above it, really include

the application area specific information and knowledge about the "real things". Very common is that new users use the standard features and possibilities of the system. However more advanced CAD-use always includes also some fine-tuning of the CAD-application. This is necessary to make it work efficiently, and for it to become a 'personalized tool'. A CAD-application also requires some maintenance work to keep the application in working condition (Björk, 1989).

In comparison to more common software applications such as word processing or image manipulation, CAD is usually harder to learn. Nevertheless, CAD can also be a tool to enhance the desired ambiance and concept of a design. CAD can become an even more powerful tool if the user can take advantage of most of the applications features. It could even be compared to a rather complex relational database system, in which all of the systems administrative work and maintenance is accomplished. Moreover, CAD-applications' interfaces can often be customized to meet the users' needs. A proper CAD-interface should be logical, straightforward and rational although clumsy interfaces can also be operated effectively. Being accustomed with the system and being motivated to work with it are really the most essential things (Björk, Penttilä, 1991).

Summarily, design decision support systems including computer aided design systems and alternative generating systems are designed to help the designers and reach the same conclusion that human experts would be expected to reach if faced with a similar problem. The structure of these systems enables users to interact with the computer. In addition to allowing someone with lesser qualifications be successful in different phases of design applications in the bounds of maintaining the notions of what the design process involves.

3.2 Evolution of the Computer Aided Design Approach

In the field of architectural design, in recent years, attempts have been made to assist designers in performing design tasks through the use of computational methods. The use of computers as designing support tools has implications in different phases of the design process. Computer based design systems provide support for calculation, documentation, animation and modeling of designs. Specifically, in last decade expert systems have been in wide use. They are used in many different fields. Bank

loan officers use expert systems for guidance in approving and rejecting loan applications. The military uses expert systems to analyze battle field conditions and make tactical suggestions. Expert systems are used in a broad range of disciplines, including automobile repair, medical diagnosis, oil exploration, financial planning, chemical analysis, surgery, locomotive repair, weather prediction, computer repair, trouble-shooting satellites, computer system configuration, operation of nuclear plants, newspaper layout, interpreting government regulations, tax preparation, and many others. The use of well-designed computer based systems emulates the reasoning and data-compiling processes to solve problems and generate reasonable alternatives (Markowitz, 1995).

However, the use of artificial intelligence in design systems assists designers at a much higher level of the design process than traditional computer aided design systems. Artificial intelligence in various applications of different fields has been improved a lot in recent years. The main concern of those working in this area was developed more user-friendly computerized tools. These new tools have been programmed to solve the problematic issues of several specific fields. For example, new architectural software has helped designers in the different phases of design. These new tools take into consideration user expectations and generate convenient solutions to design problems. These new systems that generate justifiable design concept alternatives can be considered as interactive computerized tools for design process assisting designers in achieving their goals (Eastman, 1991).

3.3 Evaluation of CAD, CAAD, AI

The design process is an interactive and multidisciplinary process. Each participant has its own view. In computational design, these views are translated to visualized results. Computational supporters are considered to be ideal as they generate well defined unambiguous analytic solutions. But this ideal can only be achieved by the use of simulations within the context of CAD, CAAD and artificial intelligence as there are no other methods available to study interrelated sub-designs in a virtual environment.

CAD (computer-aided design) is currently already a neutral and quite wide abbreviation covering generally ‘designers’ digital toolbox’ (Björk, Penttilä, 1991).

Moreover, a CAD-system covers the whole structure including a workstation (hardware) and software. CAD-workstation includes all hardware components needed to make an application work. CAD-application is a mainly graphical, software package developed to a certain application area. CAAD (computer-aided architectural design) concentrates especially into architectural-CAD (Björk, Penttilä, 1991). Artificial intelligence has been utilized to build design systems in two distinct ways; either to replicate the behavior of designers while designing or to provide design support to designers. However; much of CAD, CAAD and AI-based design systems are passive systems in the sense that they do not modify their behaviors in response to the changes in the design environment. In architectural designing most of the current CAD systems can only be used at the very late stages of the design process after most of the major design decisions have been made and very few can be used during the conceptual stages of designing. That's why they should be proved within the concept of artificial intelligence (Reffat, Gero, 2000).

Consequently, the process of design is about solving complex dynamic problems. Simulation is used in these complex dynamic problems where exact analytic and mathematical methods are problematic and not feasible. In order to define the realities and rules of architectural design process and analyze the complex interactions between the phases, design decision support systems developed using concepts from artificial intelligence concerns with the organization and representation of design knowledge; and the processes which emulate design as an activity. Therefore, the systems developed within the context of intelligent design have the potential for providing more significant help to designers in the context of possibilities offered by these technologies.

3.4 Expert Systems

Knowledge based system and related expert system are the computer software approaches. In other words, an expert system is a computer program which performs many functions in a way which would be considered intelligent if done by a human expert. Expert systems are also known as knowledge-based systems. They are computer systems that rely on a knowledge base of rules that pertain to a specific application area. Experts in the application area give rules of thumb to use in certain situations. The rules of structure are linked into if-then rules to solve the problem.

Essentially, the expert system contains the expert's thought process in troubleshooting the problem. Additionally an expert system could be viewed as one heuristic rule, or if-then statement, since all rules in the system work together to define a specific condition, which is achieved by asking the user relevant questions. Thereafter, the system reacts with one or more actions. It is postulated that an expert system becomes adaptive during an interaction with a human user if this system proves to be capable of the formulation of new relevant questions and the adjustment of actions. As a matter of fact, the system may be said to learn through its interactions with humans (Iynizio, 1991).

The system architecture of an expert system is comprised of the following components: user interface, database, knowledge base, inference engine, and synthesizer. These kinds of intelligent systems interact with the user by interface between the user and a knowledge system. Knowledge base stores information from human experts in addition to facts provided by the user during interaction with the system in a formal and hierarchical way and rule-based statements. Moreover, the knowledge base of a rule based expert system consists of a large number of rules and objects. Rules are collections of "if....then" conditions, which are employed by the system to synthesize its solution. Objects are attributes that describe items of the user interest and objects are related to other objects by some other rules describing the relations and containing the information of reasonable standards. The inference engine, a program searching for the closest match between requested actions and the set of rules in other words; it applies the closest expert solution for each user action. It deduces knowledge that is not directly contained in the base and it turns the knowledge based system into an interactive system with which the user can interact and it also formulates questions and may understand answers provided by the user in a natural language, it is also the mechanism for conveying recommendations to the user. Intelligent user interfaces, possibly employing an object - oriented environment, allow efficient interaction with the user (Shash and Reffat, 1994).

Finally, expert systems are able to solve problems. Solving problems is their intention. They are able to break a large problem into smaller parts in order to come to a solution. They understand the information given to them by the user. If the information is contradictory or ambiguous, the expert system asks for clarification or

more information. Expert systems can learn. If an expert system is faced with a new problem, it will save the information and the solution the user chooses for future use.

3.5 Conclusions

The objective of design intelligence is to make meaningful decisions toward satisfactory design solutions. By exploring design and processing data leading to new creative design solutions, it is possible to take the advantage of a computer beyond its being a drafting tool. Computers and computer programs equipped with adequate domain knowledge are able to generate creative solutions and evaluate the satisfactory response. However, the greater responsibility is to define the realms and rules of architectural design processes in the context of possibilities offered by these technologies. There are different systems handling these interactions but the most capable techniques are those that are developed using concepts from artificial intelligence have the potential for providing significant help to designers.

4. AI in DESIGN

4.1 Understanding of How AI Works

Artificial intelligence is a field of computer science that seeks to understand and implement computer based technology that can simulate characteristics of human intelligence and human sensory capabilities. The characteristics of human intelligence include learning, adapting, reasoning, self-correction, and automatic improvement. Knowledge representation is the central issue of artificial intelligence and based on a particular computer language containing letters, numbers, strings or special characters that describe the things in digital environment. In the context of possibilities offered by these different encoded structures, alternative generating systems are used to solve a wide variety of application problems. In a scientific domain that is sufficiently complex, a significant amount of human expertise is required to come up with solutions. Moreover, their use in solving semi structured or unstructured design problems is more appropriate in order to satisfy the system input and functional specifications (Reffat, Aref, 1994).

Intelligent computer programs developed within the concept of artificial intelligence are needed to enable the user to interact with the computer and used to solve a wide variety of design problems. As an example, the main objective of a living room design is to provide a good environment within the user expectations. These expectations have been steadily changing a direct result of changes in the living conditions of the users. Most people expect their merchandise to be adequately furnished. Physical variables of a living a room and expected functions so the determined merchandise taking place in the room, figure out the main furnishing parameters of the room. Therefore, careful analysis of these parameters which relate to people's responses is essential. Various solutions of compiled data may well be provided by the use of software combining the inputs in different ways. The use of computers and the software developed in the concept of artificial intelligence gives the users the opportunity to solve time consuming design problems with data

processing in a short amount of time. Convenient results can also be obtained by generating a variety of alternatives produced by interactive user applications and choosing between them. Therefore, the cornerstone of intelligence is the ability to solve problems by reasoning. These reasoning systems consist of four components:

1. Knowledge System holding facts and assertions about a problem area and it is not a database. This knowledge representation is formed by array structures, semantic networks, property hierarchies, list structures, predicate calculus expression sets, and rules.
2. Language System is needed for stating specific problems to be solved. The developer may be able to set interface behaviors.
3. Problem Processing System uses knowledge in the knowledge system to infer solutions to problems stated with language system.
4. Presentation System is important for presenting responses (Holsapple, 1996).

In addition, the inference engine is an important AI mechanism since it makes inferences from reasoning knowledge. As a result, the alternative producing artificial systems are programmed to solve design problems in a way which would be considered intelligent if done by a human expert. This intelligence is the kind mainly needed in order to generate solutions for the problematic issues of any field.

4.2 Possibilities of AI

Artificial intelligence is the study of the computations that make it possible to perceive, reason, and act. The engineering goal of artificial intelligence is to solve real-world problems using artificial intelligence as an armamentarium of ideas about representing knowledge, using knowledge, and assembling systems. The scientific goal of artificial intelligence is to determine which ideas about representing knowledge, using knowledge, and assembling systems explain various sorts of intelligence. Consequently, as the world grows more complex, we must use our material and human resources more efficiently, and to do that, we need high-quality help from computers. Since artificial intelligence can help the users to solve difficult, real-world problems, thus creating new opportunities in business, engineering, and

many other application areas. There are possibilities to use them in farming, in manufacturing, in medical care, in household work, in schools.

A design process can be subdivided into conceptual design, analysis, and detailed design. These different applications of design form the subjects of research. Some of these titles explore computer based adaptive systems and their capabilities. Capabilities which enable the system to adapt to interactions with users, explore strategies in situations where environments change. In addition to that, various computer-based problem solving structures like the programs developed for the organization of the movable and immovable equipment, space planning and also furnishing within the vision of the designer to support the needs of the user. This often covers most of the related aspects of comfort analysis developed for the internal environments of residences. These programs supporting the design process assist in solving real world problems and generate design alternatives which would normally require a human expert's interpretation.

Consequently, they are designed to reach the same conclusion that human expert would achieve if faced with similar problems and explore alternative strategies to make critical decisions quickly and efficiently. These adaptive programs have the ability to interrogate the relations within the given parameters of specific issues generating alternatives automatically. This has two main advantages: it offers non-expert user to work in a specific field without any interpretation with human experts and it also offers the expert users time saving processes of complex interactions while reaching a conclusion.

4.3 Various Programming Languages Used in AI

Expert systems were believed to represent the future of artificial intelligence and of computers in general. To date, however, they have not lived up to expectations. Many expert systems help human experts in such fields as medicine and engineering, but are helpful only in special situations. Today, the hottest area of artificial intelligence is neural networks, which are proving successful in a number of disciplines such as voice recognition, natural-language processing and some alternative generating applications in design. The explanations of these specific applications are mentioned shortly below:

games playing: programming computers to play games such as chess and checkers; expert systems : programming computers to make decisions in real-life situations (for example, some expert systems help doctors diagnose diseases based on symptoms); natural language : programming computers to understand natural human languages; neural networks : simulating intelligence by attempting to reproduce the types of physical connections that occur in animal brains; robotics : programming computers to see and hear and react to other sensory stimuli (Zreik, 1992).

There are several programming languages that are known as AI languages because they are used almost exclusively for AI applications. The most commons are: Prolog, C, C++, Pascal, Basic, Php, Java, ASP, Perl, Fortran, Lisp, Auto-Lisp, Delphi.

4.4 Evaluation of the Programming Languages

One of the most widely used languages for AI is Lisp. In addition to its success in AI, Lisp pioneered the process of Functional Programming. Lisp uses two different types of data structures: atoms and lists. Atoms are similar to identifiers, but can also be numeric constants. Lists can be lists of atoms, lists, or any combination of the two. All computation is performed by applying functions to arguments and variable declarations are rarely used. Accordingly, Lisp has built-in garbage collection, so programmers do not need to explicitly free explicitly dynamically allocated memory (Berkeley, 1974). The areas where Lisp has been used effectively are artificial intelligence (AI Robots, Computer Games (Craps, Connect-4, BlackJack), Pattern Recognition), Air Defense Systems, Implementation of Real-Time, embedded Knowledge-Based Systems, List Handling and Processing, Tree Traversal (Breath/Depth First Search) and for Educational Purposes (Functional Style Programming).

In addition to Lisp, another programming language which operates within AutoCAD to enhance and customize its native abilities is Auto-Lisp. Auto-Lisp can be used to customize AutoCAD in two principal ways. First, Auto-Lisp can automate repetitive procedures. This saves the user time and can eliminate many errors. Second, Auto-Lisp can enhance AutoCAD, giving it new commands and features, all the way from doing calculations to creating complete drawings and more. Moreover, Auto-Lisp was developed from the Lisp programming language, which has been around for a

long time. Auto-Lisp, while it retains the syntax of Lisp, is streamlined to run inside AutoCAD and has many added functions so it can interact with both the Auto-CAD commands and the drawing database. Not only does Auto-CAD come with the Auto-Lisp language already built in, but also with numerous samples Auto-Lisp programs found in a "sample" directory. As a result, all of these written programs in order to meet the specific needs can enhance user productivity with AutoCAD (Hood, 1990).

A "C" program is a set of functions that return or not return a value, and global variables. Functions and variables have prototypes in header files for external use. The programmer must manage the memory itself, using pointers and functions to allocate free fields of memory (Ammeraal, 1991). In addition to this C++ describes classes into header files, and body of methods into source files. By declaring instances of classes, users can reuse different sets of variables and methods without having to define them again. Memory management is unchanged. Overloading allows declaring a method with different parameters. Classes inherit one from other and share their methods (Dale, Weems, and Headington, 1996).

The use of Java Programs make the process clearly slower than that C++ ones. But they run under Windows, Linux and so ones. Java uses syntax that is similar to the C's one. Moreover, pointers do not exist in Java. Objects are passed by reference, simple variables by value. Garbage collector and multi-threading are built in and each file holds a single public class (Arnow, 1998). JavaScript has been created by Netscape to merge procedural programming with html inside web pages, making pages dynamic. The language is similar to Java, and elements of a web page are accessed as a hierarchy of objects: document, window, form, table, etc. Shortly, JavaScript is an object oriented language and uses the elements of the page as objects. There is no file management and no input- output functions accordingly it contains dynamic variables and arrays (Anuff, 1996).

C++ Builder is an environment of C++ and uses C++ codes. Moreover, C++ Builder's components appear to be just like any other C++ class at first glance. But there are differences between components in C++ Builder and the standard C++ class hierarchies that most C++ programmers work with. Some differences are described here:

All C++ Builder components descend from TComponent. Components are most often used as is and are changed through their properties, rather than serving as “base classes” to be sub-classed to add or change functionality. When a component is inherited, it is usually to add specific code to existing event handling member functions.

The Visual Component Library components can only be allocated on the heap, not on the stack (that is, they must be created with the new operator). Properties of components intrinsically contain runtime type information. Components can be added to the Component palette in the C++ Builder user interface and manipulated on a form and also components often achieve a better degree of encapsulation than is usually found in standard C++ classes.

The advantage of using Borland C++ Builder is that it makes it easy to implement the Graphical User Interface and also creates a professional appearance. Additionally, C++ Builder includes all the tools necessary to start designing applications: A blank window, known as a form, on which the user interface is designed, an extensive class library with many reusable objects, an Object Inspector for examining and changing object traits, a Code editor that provides direct access to the underlying program logic, and a Project Manager for managing the files that make up one or more projects (Miano, 1997).

4.5 Introduction to C++ Builder

The main types of knowledge representation for an expert system can be enumerated as: rule based systems; object oriented systems; logic based systems and mixed representation systems.

A rule-based approach is a type of production system, which uses specialized kinds of knowledge structures, reasoning, and control strategies. The other important components of an expert rule based system are the knowledge acquisition and explanation subsystems. Knowledge representation in a rule-based, object-orientated expert system is described through the establishment of appropriate relationships utilizing rules, objects, and agents. It is assumed that a relatively primitive computer-based adaptive capability can be of significant value in a problem-solving

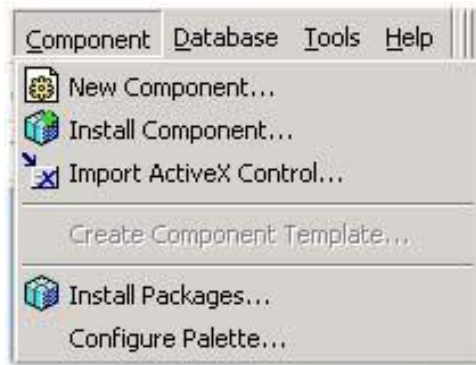


Figure 4.2 Component Resource Menu Options used in creating new C++ Builder component resources, editing, renaming, and deleting existing component resources.

Everything an object 'knows' is captured in its variables (data), everything it can do is expressed in its methods. This notion not only keeps simple things simple, but also makes complex things simple as well. No matter how many objects or what types of objects exist inside a system, relationships among those objects are clearly described by their methods. The more objects the system contains, the more complex it can become. When a change occurs in an object, other objects will react correspondingly to adapt to the change thus, the system can reorganize itself automatically. Since methods are encapsulated with data into objects in an object-oriented computer system, when a change is necessary no matter how insignificant it seems, the whole system will adapt to this change automatically through the reactions of other objects. However, in a conventional system (i.e., non-object-based), a change will require significant restructuring.

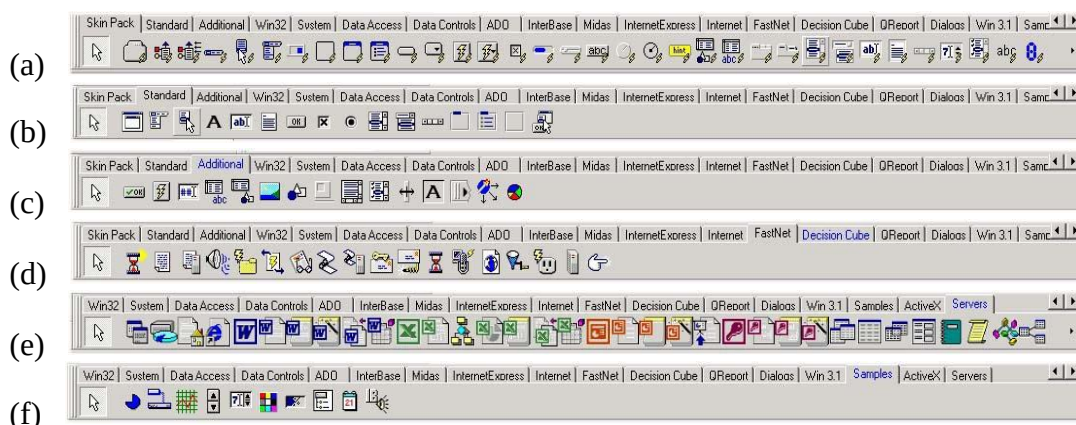


Figure 4.3 (a), (b), (c), (d), (e), (f) The components on the various pages of the Component Palette supporting Graphical User Interface.

User interface management is an important part of many projects concerning software engineering, since the user's relation to a software system is mainly determined by the quality of the graphical user interface. In order to produce a user friendly environment for the user, specifications are done with text, images and function tables. Graphics and multimedia elements can add polish to the applications. C++ Builder offers a variety of ways to introduce these features into the application. To add graphical elements, pre-drawn pictures can be inserted at design time. These are created using graphical controls at design time, or drawn dynamically at runtime. Moreover, the simple Graphical User Interface can be built using the forms on Borland C++ Builder and selecting components from the Component palette and dropping them onto forms. Many visual components are provided in the development environment itself on the Component palette. Components can be selected from the Component palette and dropped onto the form to design the application user interface. Once a visual component is on the form, its position, size, etc. can be adjusted. C++ Builder components are grouped functionally on the different pages of the Component palette. For example, commonly used components such as those to create menus, edit boxes, or buttons are located on the Standard page of the Component palette (Miano, 1997).

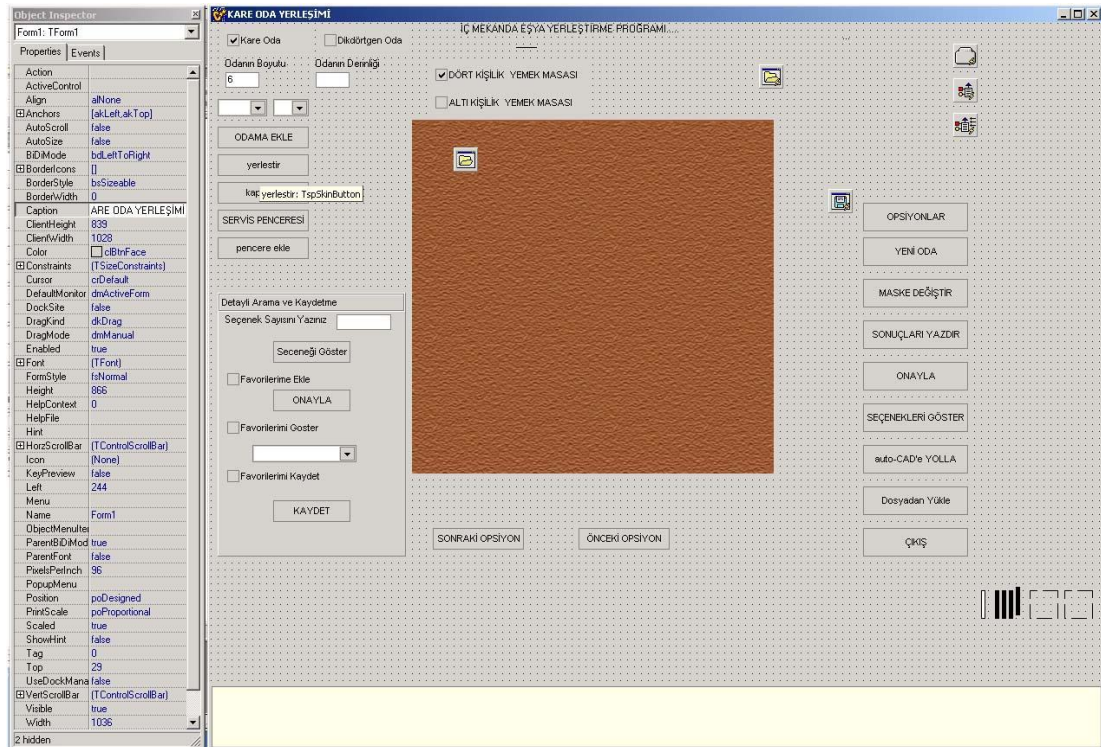
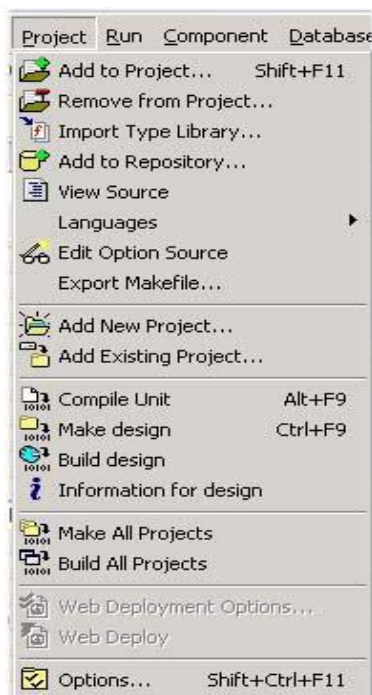
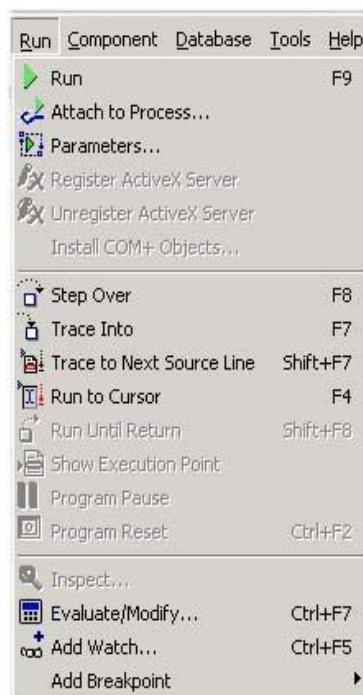


Figure 4.4 An overall structure of the system objects.

There are many other tools such as an image editor on the toolbar and an integrated debugger on menus to support application development in the IDE, command-line tools including compilers, linkers, and other utilities. If these tools are used properly, C++ Builder can be used to design any kind of 32-bit Windows application—from general-purpose utilities to sophisticated data access programs or distributed applications. Other than these, by making code in object oriented fashion, it makes it easier to extend this program for another purpose. It also makes it easier for a programmer to understand the flow of this program. The Visual Component Library is based on the properties, methods, and events model. The Visual Component Library is a hierarchy of objects, written in Object Pascal and tied to the C++ Builder IDE that allows the user to develop applications quickly. Using C++ Builder's Component palette and Object Inspector, The Visual Component Library components can be placed on forms and specify their properties without writing code.



(a)



(b)

Figure 4.5 (a) Project Menu, offering the possibilities of adding, removing, reordering projects. (b) Run Menu commands, used to debug the program.

Taking this a step further, Borland C++ builder is a useful tool that provides the user with a very productive and easy to use environment, eliminating many of the errors associated with command line programming and the compiler at the heart of the software works efficiently in comparison to other C++ compilers. It also implements much of the Windows Graphical User Interface as to make it extremely easy to produce Windows programs that make use of dialogues and forms. TApplication, TScreen, and TForm are The Visual Component Library classes that form the backbone of all C++ Builder applications by controlling the behavior of the project (Miano, 1997).

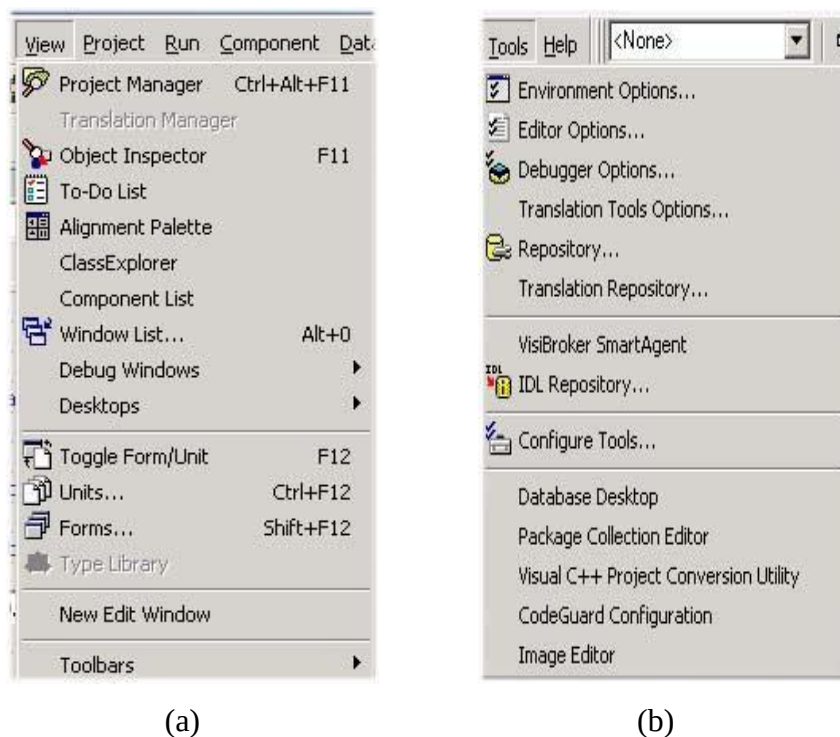


Figure 4.6 (a) View Menu commands, needed to display or hide different elements of the C++ Builder environment and open windows that belong to the integrated debugger. (b) Tools Menu commands, used to view and change environment, editing, debugger settings.

The TApplication class forms the foundation of a Windows application by providing properties and methods that encapsulate the behavior of a standard Windows program. TScreen is used at runtime to keep track of forms and data modules that have been loaded as well as system specific information such as screen resolution and what fonts are available for display. Instances of the TForm class are the

building blocks of the application's user interface. The windows and dialog boxes of the application are based on TForm. (Miano, 1997)

C++ Builder is an object oriented programming language and there are ranges of benefits that can be identified for object orientation. The main benefits can be summarized, as object orientation is a more intuitive programming paradigm than the other excepted approaches such, as procedural. The objects respond to the use in specific ways while interacting with them. Moreover, there is no need to worry about exactly what type of object the user will get at run time, only that it must respond to the message the user send it. This means that it is a great deal easier to write reusable, compact code, than in many other languages. As long as the external behavior of these objects appears to remain the same, the internals of the object can be completely changed. However, external uses of the object need never know. More than these, the interfaces can be developed which make the user available when the object is implemented.

Consequently, because of its (C++ Builder) object oriented feature which allows inheritance from other classes such as increased code reuse and simplification of code; graphics and multimedia elements and other visual components stored in the Visual Component Library, this project has been implemented using Borland C++ Builder and the code for this system is written in an object oriented fashion.

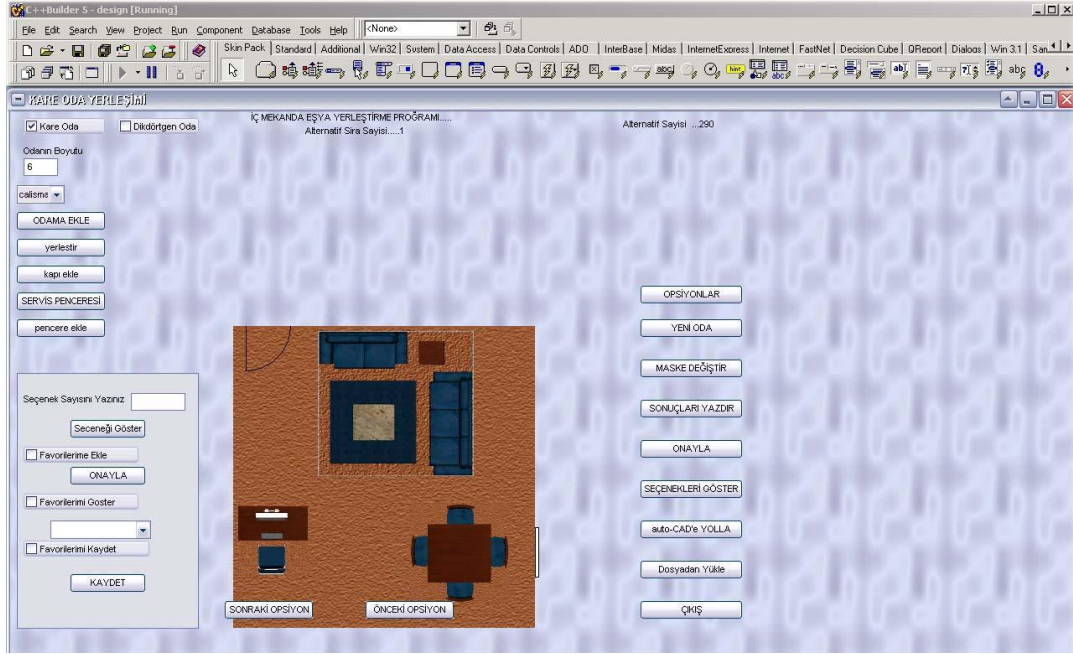


Figure 4.7 An example of a design composition generated in an object oriented fashion in C++ Builder environment.

4.6 Proposed System Architecture

Indoor quality of living room by means of well-located functions within the user expectations and well-decided relations between each component taking place in the room, aims at providing a comfortable place for the users to perform their activities. But, each user has his own needs so that it might be what was good for one is obsolete for another.

The combination of each and organization consists of a series of criteria used to investigate the internal comfort conditions and determined how these conditions meet the user requirements. For example, components such as eating table, desktop, and sofa need to work together with the organization of the right places for decided functions, for instance dining, working and living space. This software aims to get the user equipped and organized to create effective and efficient living environment in the living room. The alternative generating system also achieves balance through the selected functions within the physical conditions and other components of the room such as door, window, room-size and circulation area, etc. A wide range of generated alternatives allows the user to select his own solution.

The structure of this expert system consists of: knowledge base, database, inference engine, user interface, explanation of results and data-integration with a CAD system. Knowledge base stores information, principles of furnishing, in addition to facts provided by the user during interaction with the system in a formal and hierarchical way and rule-based statements. Rules are collections of "if...then..." conditions, which are employed by the system to synthesize its solution. To set and explain the rules which define the relations between each component, a collection of facts including sentences about general facts is recorded. The rule is composed of parameters and values. If all or some of the conditions in the clause "if" are met, the rule assigns a value to the parameter. The "if" part of the rule consists of one or more of the qualifier's value and conditions that can occur and the "then" part consists of another qualifier value or variable or choices and represents actions that must be taken.

Inference engine of the system is the program searching for the closest match between requested actions and the set of rules in other words; it applies the closest expert solution for each user action. The inference engine deduces knowledge that is not directly contained in the base and it turns the knowledge based system into an interactive system with which the user can interact.

The application of interactive interface allows efficient interaction between the user and the different merchandise. This process allows the users to select the content of merchandise such as various types of sofa differing from each other in size and shape, and other equipment that will be included in the room like serving window opening to the kitchen, door, window and their various standardized features. Standardized feature is one of the main design criteria while planning as well as the physical parameters which form the room. Things like size differences in width and height that decide the path that the system will work on.

The explanation part provides a visual description to the user of why the system asked for some information and what kind of specific conclusions the system arrived at. This system is integrated with a CAD system and designed to allow the designer to interact through graphical interface tools, components and a set of menus, to specify the requirements of the desired design solution. Furthermore, data integration

with the CAD system is demonstrated with an icon in the interface of the system and a command in the CAD environment, which can be easily applied by the user.

4.7 Conclusions

There are a number of computer software applications for home design. However, some of these computer based design programs developed to support the process of design are intended principally to assist in drawing and simulation of a design and do not make decisions and generate alternatives. Therefore, these systems are inadequate in the content of processing wide range of relevant knowledge and complex interactions.

Some of the others are based on retrieving a previous design from a structured database and adapting it to the new circumstances. They simply process in the concept of generalization which forms a general description from a specific item. In addition to these, there is another data-searching program type. It only includes only scaleable bitmaps in the memory, assisting manufacturers to provide their own catalogues for the consumer to try out in a simulated home. All these kinds of systems offer to the user a selection between alternatives. However, they are still inadequate as the inference engine of the system does not work properly in turning data into an interactive system with which the user can interact. In conclusion, some of the developed computer based design programs specialized in different phases of design process, do not allow efficient interaction with the user. Therefore, intelligent user interface and inference engine processing the information provided by the user during interaction with the system are needed with the aim of arriving at a specific conclusion responding to the user needs.

Consequently, this work addresses the potential consequences of applying intelligent problem solving techniques as an approach for furnishing analysis of the living rooms of residential environments.

5. PROJECT DESCRIPTIONS

5.1 Implementation

This study offers a computerized tool that can be used while furnishing a living room. A major function of a living room is providing a comfortable environment for its users. Environment can be defined conventionally in terms of space organization, indoor quality, furnishing in the context of well-located functions within the user expectations and well-decided relations. Design professionals such as interior designers and architects incorporate such issues into the designs of the environments for the purpose of achieving qualified environments for the tenants. The living room quality is a combination of components. It involves an interaction between both the user and the environment. The goal is to create the most convenient living situation for the user. One that successfully accommodates all the activities that goes on in it. These components referring to dining, working and sitting actions have to meet a number of user expectations to achieve comfortable conditions for the user. In this project, general furnishing parameters were extracted for generating satisfactory furnishing results of a living room accordingly an alternative generating system is developed.

Human-computer collaboration proposes an approach in which computers and humans co-operate to solve problems, especially to solve those problems that neither can effectively solve alone. For example, an expert system helps non-expert humans to solve problems and make decisions based on internal expertise. While engaged in the process of problem-solving and decision-making, human beings typically draw upon a broad range of personal experience and often rely upon intuition. Computers, on the other hand, have a narrow area of expertise, greater computational power, and lack a good mechanism for knowledge acquisition through experience. The use of a computer aided system in furnishing permits the non-expert users to behave as designers and offers to all users the opportunity to generate satisfactory results in

short time. This tool has proven to be effective in solving problems of furnishing process.

Object-oriented fashion of this expert system includes writing rules and object methods that manipulate the behaviors of data therefore, information and rules can define knowledge. Objects of the living room represent information as the object-oriented system processes information and object describes its states with the data it holds, and manipulates the data through its methods. All the features of the system mentioned are supported with the codes written with C++ Builder language and can be realized by the use of interactive interface which support the transformation of data between the software and the user. The explanation of all the icons placed in the interface, the methods how the user can work properly with the details of these icons and the data needed from the user for these tools to designate the expectations and also the requirements. Moreover, the functions and the rules in the system working together to define a specific condition, which is achieved by asking the user relevant questions, could also be seen below.

The explanations of the icons taking part on the interface:

- Selecting shape of the room:

First of all this tool is useful to define the shape of the room that the user is going to furnish and selecting this shape is as simple as clicking on it and there are two possibilities to be checked: square and rectangle room. It is essential to guide the program to communicate with the other related icons such as room sizes to form the shape of the selected room type since all the selected merchandise will be located on that layout with the given data. Commonly used living room shapes have been selected, and these are optional as it can be seen in the interface. These options are square and rectangle and become active when checked.

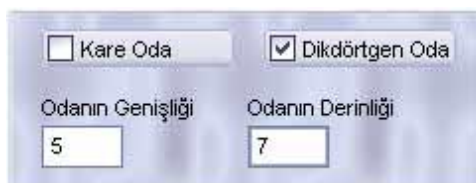


Figure 5.1 'Select shape of the room' tab.

- Editing room size:

After the selection of the room type another window opens and become active when the user edits the required data about the room size. As mentioned, some icons interrelate to each other. For instance, if the user selects square, the dimension of only one edge will be enough to form a square, only one box becomes visible and the other box needed to get the dimension of other edge to form a rectangle becomes invisible to avoid confusion. Additionally, two boxes named as “room width” and “room depth” opens to convey the information to the inference engine in order to form a rectangle, if the “rectangle” has been checked. Not until the user selects the ‘arrange’ icon will the program arrange all the selections. Up until this point it is still possible for the user to change his mind about the shape and the size of the room.

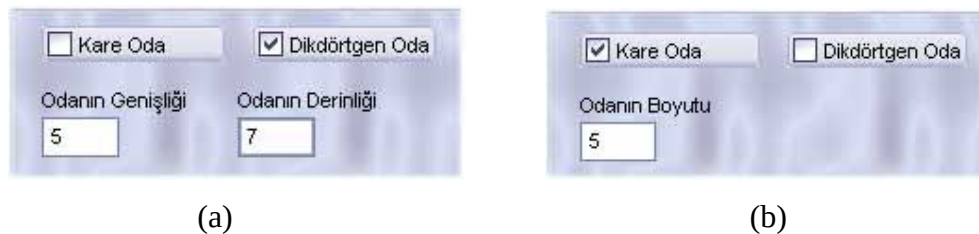


Figure 5.2 (a) Edit box used to edit sizes of a rectangle room (b) similar to (a) but used to edit the size of a square room.

- Selecting furniture and deciding their numbers:

Another step is the selection of furniture that can be realized by the use of a combo box containing all types of furniture, number pane and “add to my room” icon. Each time the selection filter helps the user to see what can and can't be selected for the project. Elements that are not displayed can't be selected. In addition, other box displaying the numbers does the same thing by displaying or not displaying the numbers as the objects are activated.

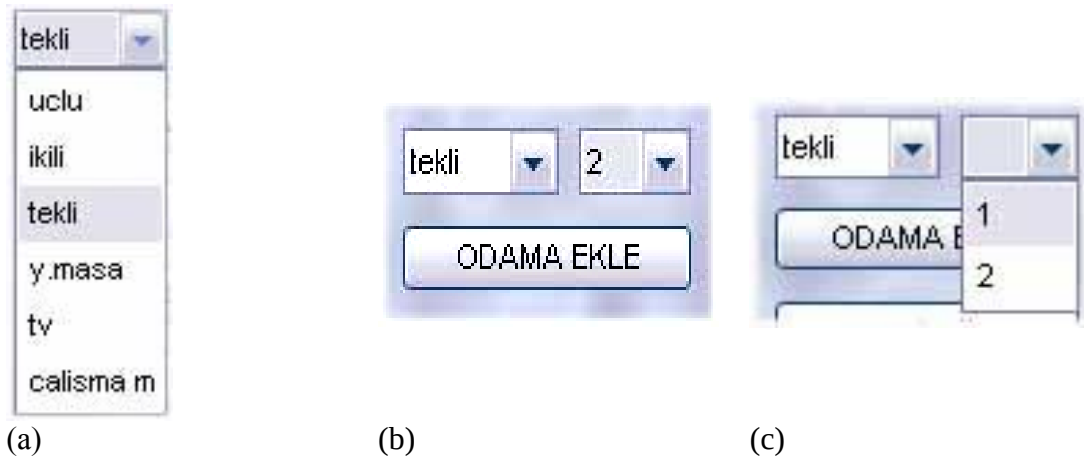


Figure 5.3 (a) Combo box used to choose furniture (b), (c) explanations of the editable numbers of the chosen furniture.

The selectable furniture types in that combo box are: sofa(3), sofa(2), sofa(1), eating table, desktop, and television. The user picks only the elements he wants to be furnished. As the cursor moves on to the combo box, the object that becomes currently active can be inserted by clicking on it.

Generally, after any progress on the project, it will have more elements to select in order to get the details of the room or any component. Thus, after the selection of most elements in that list, another combo box, needed to edit the number of the element, opens. The items listed in the number pane will vary according to the type of element. These numbers automatically change in relation to the active object. For instance, a maximum of two can be seen when sofa(2) or sofa(1) is currently active and the possible number for sofa(3) is only one.

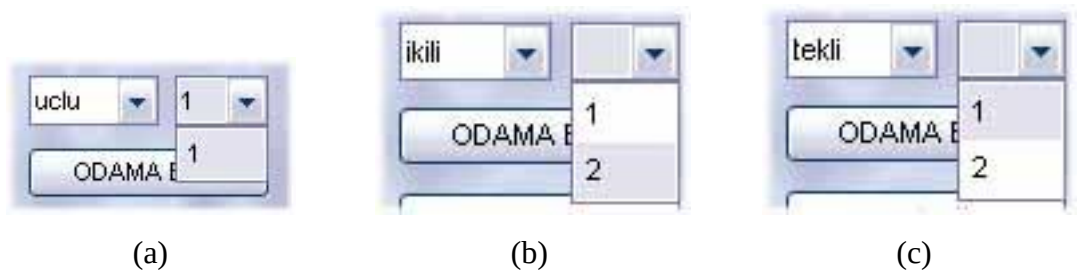


Figure 5.4 (a) 'Choose furniture' combo box showing the selection of sofa(3), (b) and (c) describes the editable numbers of the sofa(2) and sofa(1).

When the user activates eating table, television and desktop, number pane becomes invisible as there can not be more than one desktop or eating table in living room. However, there is the possibility to decide their properties with other sub-boxes.

These other possibilities are called sub-boxes. They are used to define the details. Normally, they are invisible in the furnishing process but they become visible when the user clicks the related element.

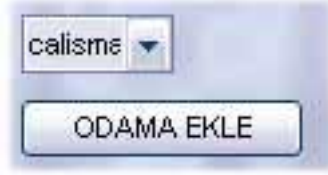


Figure 5.5 ‘Choose furniture’ combo box showing the selection of desktop.

Different elements have different types of properties. When the eating table is clicked, two sub-boxes open that allows the user to choose sub-types of the selected element. There are two possibilities for eating table which differs from each other in size: eating table for four people (200*200cm) and the other is for six people (200*300cm) and they can be added to the list in one simple operation, clicking on it.



Figure 5.6 (a) ‘Choose furniture’ box showing the selection of eating table (b) ‘Select the type of the eating table’ tab (c) The list of the selected movables.

“Add to my room” icon should be clicked after all these selections to transfer their information to the engine. This is the way to inform the program what elements are going to work with. If this is not done, the engine will not have the information it needs. Consequently, any of the desired choices will not be added selections to the list. Failure to hit this button will cause these choices not to be seen on that layout. In addition to this, the program informs the user about the selected objects with a text box at the bottom of the interface changing the text dynamically related to the selections. The images below can be seen on the screen and they are needed to inform the user when an object has been selected twice. It is useful to remind the user what he has chosen. After all of this information has been passed on, it is then permissible to continue with the furniture selection “ok” should be clicked.



Figure 5.7 (a), (b), (c) Error tabs appearing when any of the movables are selected twice.

At least one element (sofa3, sofa2) of the sitting group should be chosen to signal to the engine to the process. The engine will then arrange the selected objects according to their functions, as there should be the sitting function in the living room. But, the selection of eating and the working functions, as if they will be located in the room or not, is optional.

- Selecting immovable equipment (door, window, serving window) and deciding their characteristics:

“Add door” button is used to define the behavior of the door when clicked.



Figure 5.8 ‘Add door’ button

In order to ask the user all the characteristics of the door, another window appears. This tab allows the user to choose and also edit the characteristics of the selected element. These behavior characteristics include the information about the distance from the selected corner, the width of the door and the edge on which it will be located.

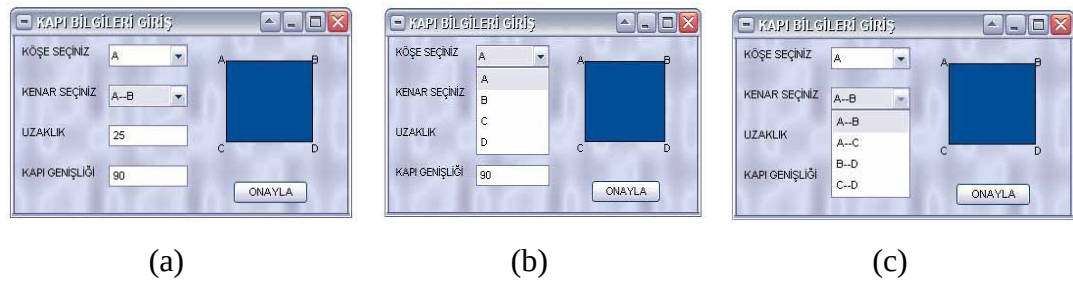


Figure 5.9 (a), (b), (c) Explanations of needed data from the user for door selection.

A schematic rectangle can be seen in the figure above which show the names of the corners and the edges before selection. The first combo on that window contains the selectable corners: A, B, C, D; the second one includes the names of the edges: AB, AC, BD and CD; the third one asks for the distance from the selected corner and the last forth one is needed to define the width of the door. After this selection, the “approve” button should be clicked to send all of this data to the engine. If this is not done correctly, properties will not appear on the menu. In addition, the program calculates and designates the reasonable opening side of the door when the adequate data asked in this window, is given by the user. If the distance from the selected corner is larger than the half size of the edge which means the door is closer to the other corner of that edge, the opening side dynamically changes. This window includes information that allows the program to accurately illustrate and dynamically change the opening side according to the distance to the corner. These properties of this component can be changed, as long as the user closes the window by clicking the “approve” button. At last, in order to avoid any data coincidence as these all are decided by the user; the program compares this information with the other immovable equipment’s. If there is any, an error tab opens to inform the user about controlling the given data and requesting that it be exchanged with reasonable data.

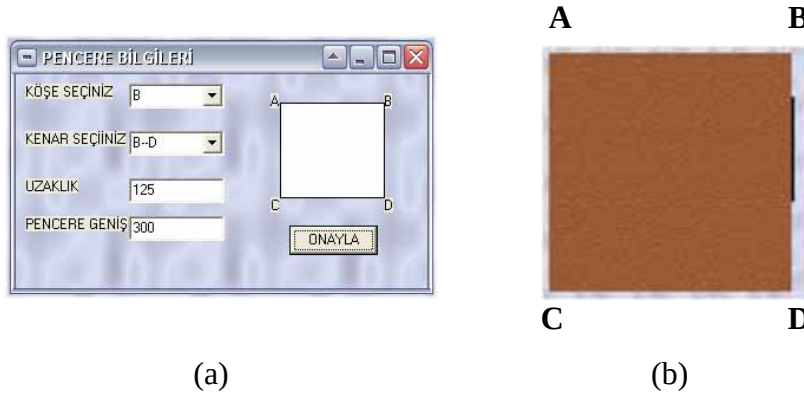


Figure 5.10 (a) Explanation of needed data from the user for window selection
(b) demonstration of a window generated with the given parameters.

Other immovable equipment can be chosen the same way door selection was made. Corner, edge, distance from the corner and the width are needed to locate the serving window and window. After clicking “ok”, this information is also collected in the engine. The selection of this information is optional to the user.

Another important characteristic of the serving window is that the selected eating table's position will be decided within the settings of the serving window. In the context of reasonable standards, eating function should be close to the serving window since it refers to a connection with the kitchen. For this reason, in the case of serving window selection, it is vitally important that the program automatically locate the eating table according to the given parameters and the rules written in the software, to avoid nonsense alternatives.

Consequently, if the serving window is selected there will be fewer alternatives due to the fact that it will be a restriction while furnishing. However, if there is no serving window, there will be more alternatives because the engine will search for the table's location all around plan lay-out.

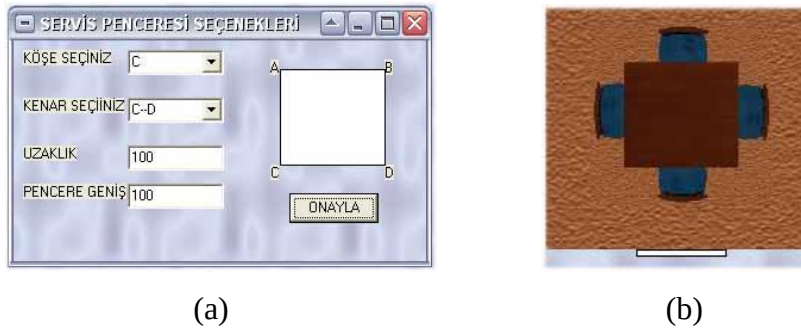


Figure 5.11 (a) Explanation of needed data from the user for serving window
(b) demonstration of a serving window generated with the given parameters.

- Arranging selections:



Figure 5.12 'Arrange' button

After clicking the “arrange” button, any data modification about the movable objects is avoided since this button is the way to tell the program; ‘with these selections I am going to work’. By the way, all the settings and options chosen are locked in the engine and they will not be generated until the user tells the program “calculate solutions”.



Figure 5.13 ‘Calculate solutions’ button

To start the calculation, some of the immovable equipment should be selected which are not optional, as their selection is important while furnishing, like door referring to the entrance. With this button, plan layout refreshes itself with the given room sizes and the proportions between the width and height of the selected rectangle. Additionally, it covers this invisible layout with a decided image to make it visible and scaled.



Figure 5.14 (a) Showing the data of a rectangle room edited by user and its demonstration (b) showing the data of a square room edited by user and its demonstration

The concept of this program is to illustrate solutions after having the data of both immovable and movable equipment. Therefore, although the engine is ready to start the calculations for movable objects, data is locked to be compared and calculated with all other information.

- Deciding the minimum distance between the movable equipment and frequency of the generated alternatives:

When all the selections about the equipment are completed, there is a possibility to choose the settings that control the way how the program will run. The user can adjust these settings also to change the speed of the process.

There are different ‘display modes’ to specify these two optional settings. The first setting is about the selection of minimum distance between the objects when their position is close to each other. The second refers to the amount of alternatives, which the program will generate. In other words, the second one is used to limit the

amount of the alternatives by searching the area with checked mode. For example, if the second mode has been checked, the planning area will be searched in every forty centimeters for each movable component's location by analyzing their relations with immovable equipment.

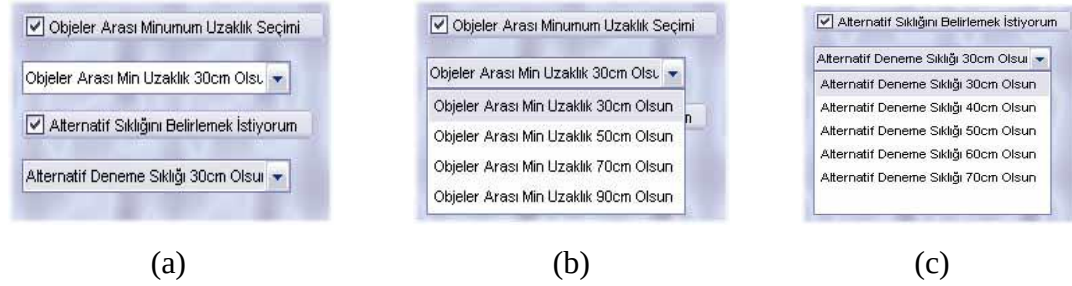


Figure 5.15 (a) Showing the two tabs used in defining the frequency of alternatives (b) showing the selectable distance options between the selected objects (c) showing the selectable distance options used by generator module in defining the mentioned frequency.

There are four modes for the first setting and five for the second setting. One common point in these two settings is that when the mode number becomes higher, fewer solutions will be generated. Said another way, these modes can be used to make the program eliminate close solutions automatically. Different modes can be used for a variety of applications. Alternatives can be automatically viewed in any of these modes. They are also useful when the user wants to navigate through the speed of the process. These setting are optional as mentioned before; if none of the modes are checked, the program will arrange itself to generate maximum alternatives.

- Approving the selections:

First, the program has to collect all the data concerning the desired equipment. Then it is ready to compile all of this information. Using the “calculate a solution” button, the user can calculate all of the data collected in the engine.

The first step is to know if the selections fit the room or not. The circulation area is calculated and then added to the total area of the selected equipment and then compared with the total area. If this number is larger than the total area, the user is informed by an error tab, “no alternatives generated”. Second, every possible settlement of sitting group is added to the list to be tested. For example; if sofa(3), sofa(2) and two sofa(1) are selected for sitting group, there are ten different

alternatives and the same process explained above is calculated again and again for each sitting type. The number of alternatives also alters when the content of sitting group changes. There is the possibility of choosing a TV as part of the sitting group. Every alternative with TV or without TV differs from each other in size as does the area. Therefore, after calculation, some of them may be eliminated if they do not fit with the other selected components. Displaying or not displaying action in the project is decided automatically within the software by a function containing defined formulation. The images below are showing the different settlement alternatives of one of the sitting group. Additionally there are eleven different sitting groups which make twenty-two when the TV is selected without their rotated conditions. Every alternative has its rotated condition.

Selectable alternatives and different combinations of sofas coming together to form a sitting group, can be summarized as:

3-2-1.....sofa(3)+ sofa(2)+sofa(1)
 3-2-(2)*1.....sofa(3)+ sofa(2)+2 unit sofa(1)
 3-2.....sofa(3)+ sofa(2)
 2-(2)*1.....sofa(2)+2 unit sofa(1)
 3-(2)*1 sofa(3)+2 unit sofa(1)
 (2)*2..... 2 unit sofa(2)
 2-1 sofa(2)+sofa(1)
 (2)*2-(2)*1..... 2 unit sofa(2)+2 unit sofa(1)
 3-1 sofa(3)+sofa(1)
 3..... sofa(3)
 2 sofa(2).

All the compositions in Figure 20 can also be created with TV.

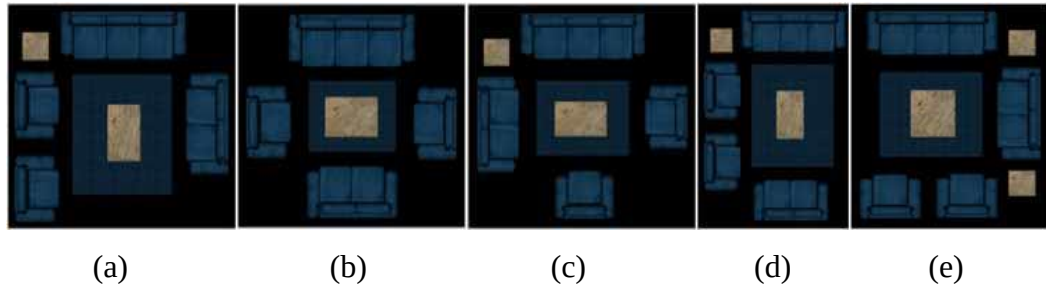


Figure 5.16 (a), (b), (c), (d), (e) Examples of furnishing design compositions when sofa3, sofa2 and two of sofa1 are selected.

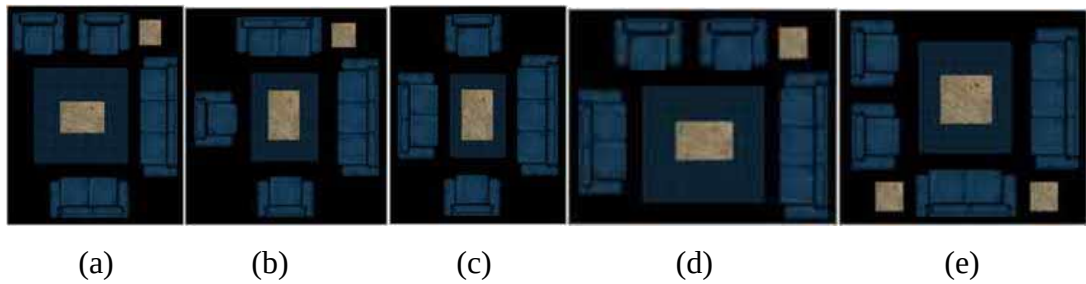


Figure 5.17 (a), (b), (c), (d), (e) Rotated alternatives of the compositions in Figure 5.16 as they differ in width and height.

Not only the sitting group but also other eating table and the desktop if selected join to the list to be calculated and compared. Their rotated conditions also tested as an alternative. Then at the same time selected functions and their fitting alternatives are located one by one on the plan lay-out.

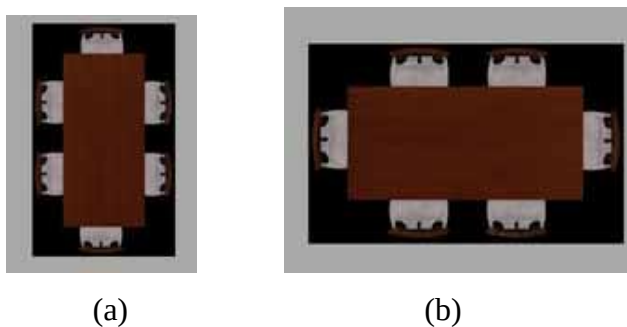


Figure 5.18 (a) Image of the not-rotated eating table for six people (b) showing the rotated alternative of the same image in (a).

Depending on the complexity of the project, the speed of the computer and the display mode options chosen, generating alternatives can take some time. For

example if the serving window is not selected the program runs with sixty-four vicious circles and can generate more than 10.000 alternatives. However, if the serving window is selected, the program will run with sixteen vicious circles and obviously fewer solutions will be generated. In the second case, the performance of the program will be higher. While these calculations keep going on related images are called that is why it can be somewhat slow to come up with “display” button. However; if the serving window is selected, the program will run with sixteen vicious circles therefore fewer solutions will be generated and the performance of the program will be higher. As mentioned in some conditions to inform the user error tabs can appear. For example, if none of settlement alternatives of the selected equipment fit the room, the error will say “no alternatives generated”. Moreover, if the serving window is selected, the location of table will be close to it and a control is done to compare its stable position with immovable equipment of the room like door.

- Listing the generated alternatives:

All views of reasonable locations are loaded when the “display” button becomes active. A view can be displayed by clicking on it. The view that appears in the drawing area is the first alternative’s 2D Plan view. This displays the merchandise as if you were looking at it from above. In addition, the number of the results can be seen in the status bar at the top.

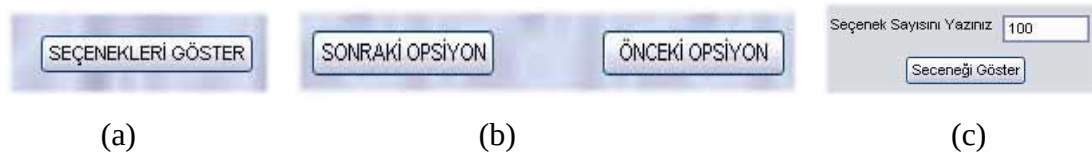


Figure 5.19 (a) ‘View alternatives’ button (b) two possible icons can be used in demonstrating alternatives arranged in order (c) edit box, used in demonstrating the edited alternatives.

There are two ways to display the solutions after clicking the “display” button. One way is to go with one by one from the beginning. However, it may take more or less steps to reach the desired convergence. The other way is to edit the number of the alternatives in the views pane and let the program display this solution. This is preferred when trying to skip the similar solutions.

- Adding the satisfactory ones to the favorites and displaying the edited alternative:

There is also the possibility of creating and manipulating an imaginary folder that would contain the selected options with the use of “add to my favorites” button. This tab allows the user to collect the satisfactory results in memory. Searching to find the viewed and satisfied options again is a time consuming process, as there can be an incredible number of options. However, checked results can be viewed easily during the process, as they are loaded to the memory. The Combo box interrelated to “view favorites” shows the numbers of alternatives edited and added to the favorites and when any number in that box becomes active, the program displays the view of that number. These two tools, “add to my favorites and “view favorites” are working collectively and useful to get the satisfactory solutions into the memory and to call the convenient options easily.

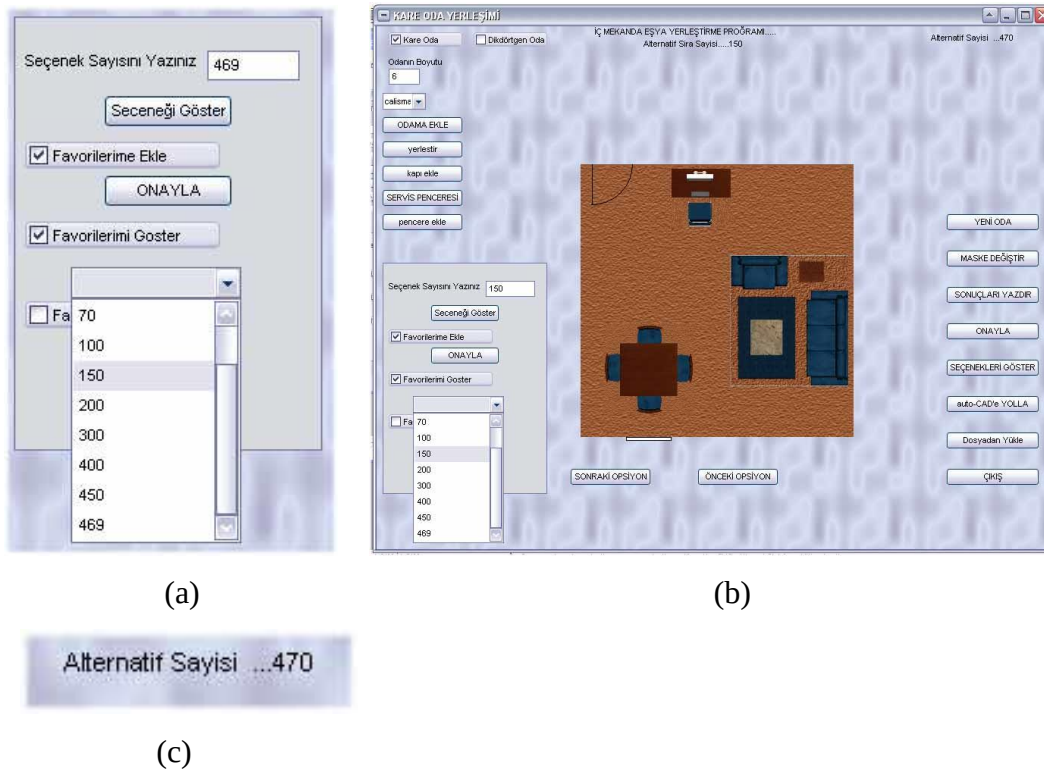


Figure 5.20 (a) Combo box showing the satisfying results according to the user's (b) furnishing composition produced while exploring the design space (c) number of generated alternatives.

- Saving the edited alternatives as a text file:

In order to save the satisfactory selections, another tab, “save favorites”, should be checked. External catalogs are saved as distinct text files and can be opened, closed

or deleted. When the save file pane opens, the text file containing the data of each alternative should be given a name and then saved to make the selected results permanent. The program will refresh all edited parameters when it is opened again. Therefore, there is no way to see the selections without saving them in text format. This is the only way to view these files again and can call them by a given name.

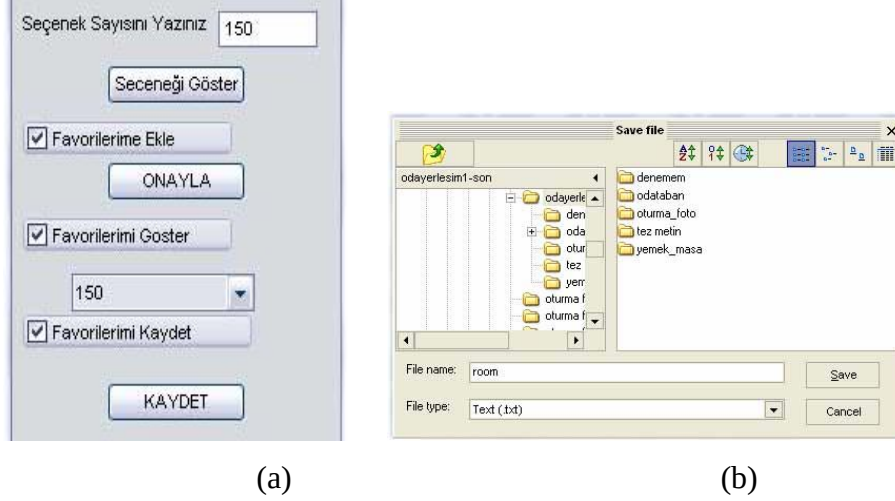


Figure 5.21 (a) Tab containing all buttons used while viewing and saving the alternatives (b) 'Save file' window used while saving the edited alternatives as a text file.

- Loading from file:

As the user saves the favorites, a current file is created with a given name. The current text file is the list of data referring to the all elements in the alternatives saved along with the project. When the program is re-opened, the file can be opened as well. This is how the user knows what has been selected before and which results are considered as satisfactory.

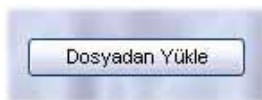


Figure 5.22 'Load from file' button.

To load the saved text file into the program, "load from file" button should be clicked. After that, "open file" window and the name of the file in it can be seen. In order to allow the program to display each result, file should be opened by double clicking on it. Moreover, loaded results can be viewed one by one with the tools, "next option" and "previous option" at the bottom.

The text above is the first line of the list shown in the image below and it gives an idea for the format of the data. There are reagents between every data to tell the engine about the numbers. Moreover, this line begins with the data related to immovable equipment and continues with the movable ones'. Each rectangle defining the location of each component can be calculated and drawn with the given data in this text format.

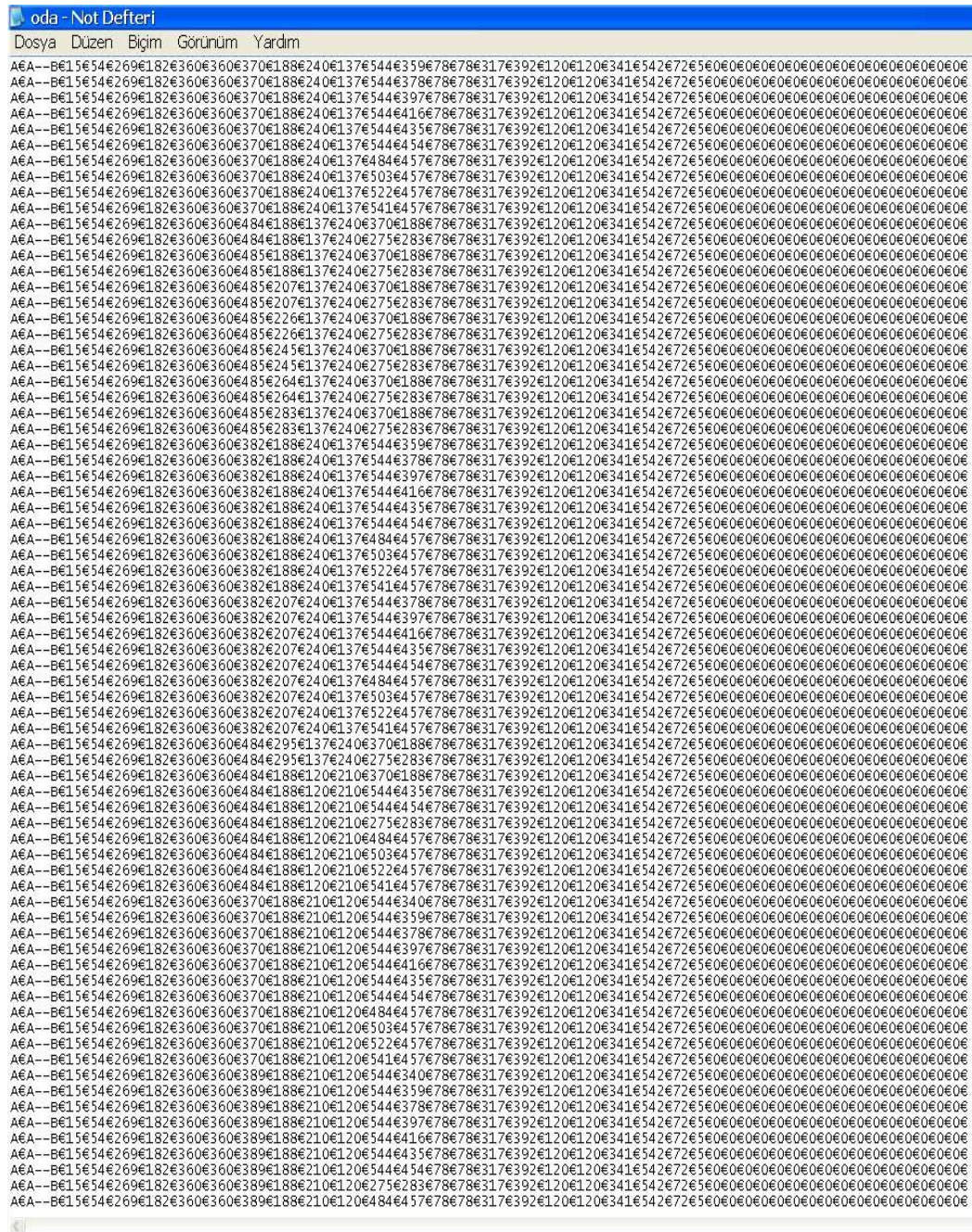


Figure 5.25 Data of all alternatives generated with given parameters, in text format.

All information about alternatives is automatically stored in text format under the same folder with exe. Basically, this information includes room and each component's width, height sizes and left, top coordinates. Since C++ Builder is object-oriented, coordinates and width, height sizes of each settlement, used to find the coordinates of each corner, can be easily defined in CAD environment with the use of the formulized data in the text file. As mentioned; this text file contains information in a special format as to allow the Auto-Lisp to transform the data into its format. This transformation can be viewed with the command "ODACIZ" which is developed in Auto-Lisp and available in CAD environment. "ODACIZ" command helps to demonstrate the calculations in Auto-Lisp. This command draws rectangles with the given parameters. Rectangles which appear on the screen refer to different components selected during the furnishing process and can be modified. It is only to illustrate the locations in order to guide the expert user while planning.

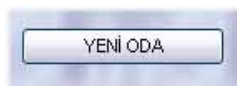


Figure 5.26 'New Room' button

- Asking for a new room:

"New room" button is the way to restart the program without closing it. After finishing all the applications about the first project, a new plan layout should be created in order to re-fresh the edited parameters. Additionally, as to test the other parameters, the user should reapply the mentioned methods and edit the needed data for each tool in every new project.

- Displaying all the selectable options:

"Options" works as a guide while using editing tools. It shows all the selectable components and visualizes their different settlement alternatives. The views of alternatives dynamically changes when the cursor moves on the names of the selectable components. These components consist of different elements and their alternatives differ from each other in size. While the processing some of these alternatives may not be seen on the lay-out as their size does not fit the room with other selections. In order to work with this eliminated settlement of selected component, room sizes can be enlarged and tried again.

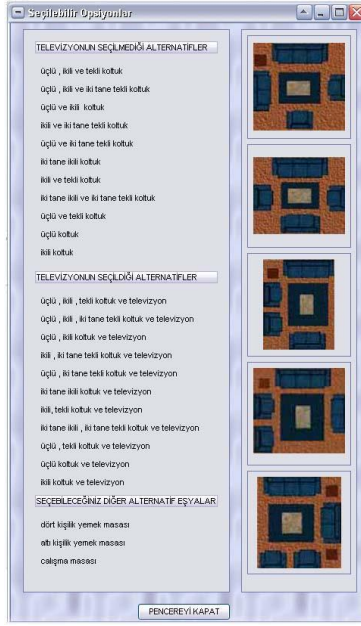


Figure 5.27 Demonstration of all selectable movables within the program.

- Exit

This tool is exactly used to quit the program at any time. To re-open it, the EXE should be clicked.

- Changing the mask of the interface:

“Change the mask” gives user the freedom to select the screen layout between fifteen alternatives to suit the desired design. Some different types of interface: layout, variances in color, texture and behavior of activated icons. Some of them can be seen below.

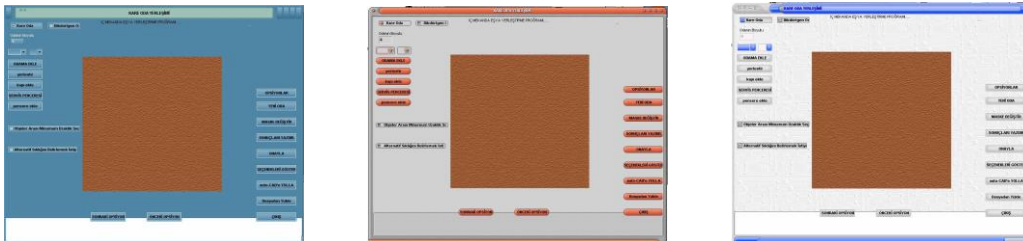


Figure 5.28 Various alternatives of mask compositions.

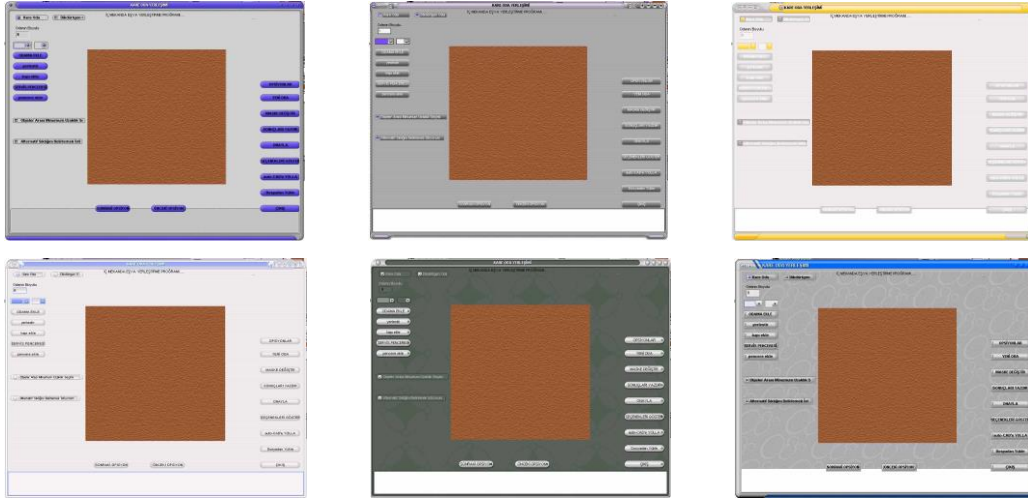


Figure 5.28 Various alternatives of mask compositions.

Combination of different colors and textures for each image is created with the use of Photo-Impact and the behaviors of icons are formed in Skin-Builder. Skin-Builder has a library that allows for the changing dynamically of almost any aspect of the visual appearance of forms and icons. The skin-data include ini-file and *.bmp, *.jpg files and ini-file and is also supported by the C++ Builder program. Therefore, "SkinBuilder" can be named as the skin application program of C++ Builder. To generate these layouts, source files should be created and then with "SkinBuilder" program they should be combined in ini-file format. Then, to load one of these ini-file formatted lay-outs, any of the folders in the menu should be opened and applied.

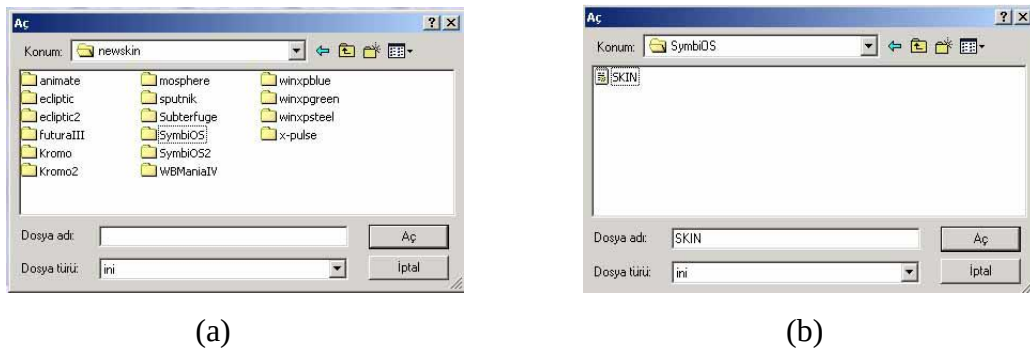


Figure 5.29 (a), (b) 'Open' window used to apply the selected mask composition to the interface.

In short, this tool is needed to produce a user friendly environment. Moreover, it asks for the user's opinion about the layout. In fact while deciding and applying the satisfactory layout, user has been included in the design process of the interface.

All these tools explained above are supported with codes forming twenty-six functions in the text part of the system. In addition, these functions undertake the main task while generating alternatives. If we summarize, “approve” button and the functions are inter-related to each other. When the “approve” button is clicked, engine starts to run all the functions in a pre-arranged order. It calls the needed functions to search for data and draw a conclusion.

The flow chart below shows the general concept of the system by demonstrating the flow of data and correlation between functions and also when which function becomes active. Detailed flow chart can also be seen in Appendix-B. Moreover, the explanation part of the functions defines the usage of applicable design knowledge by the system during the generation of alternatives.

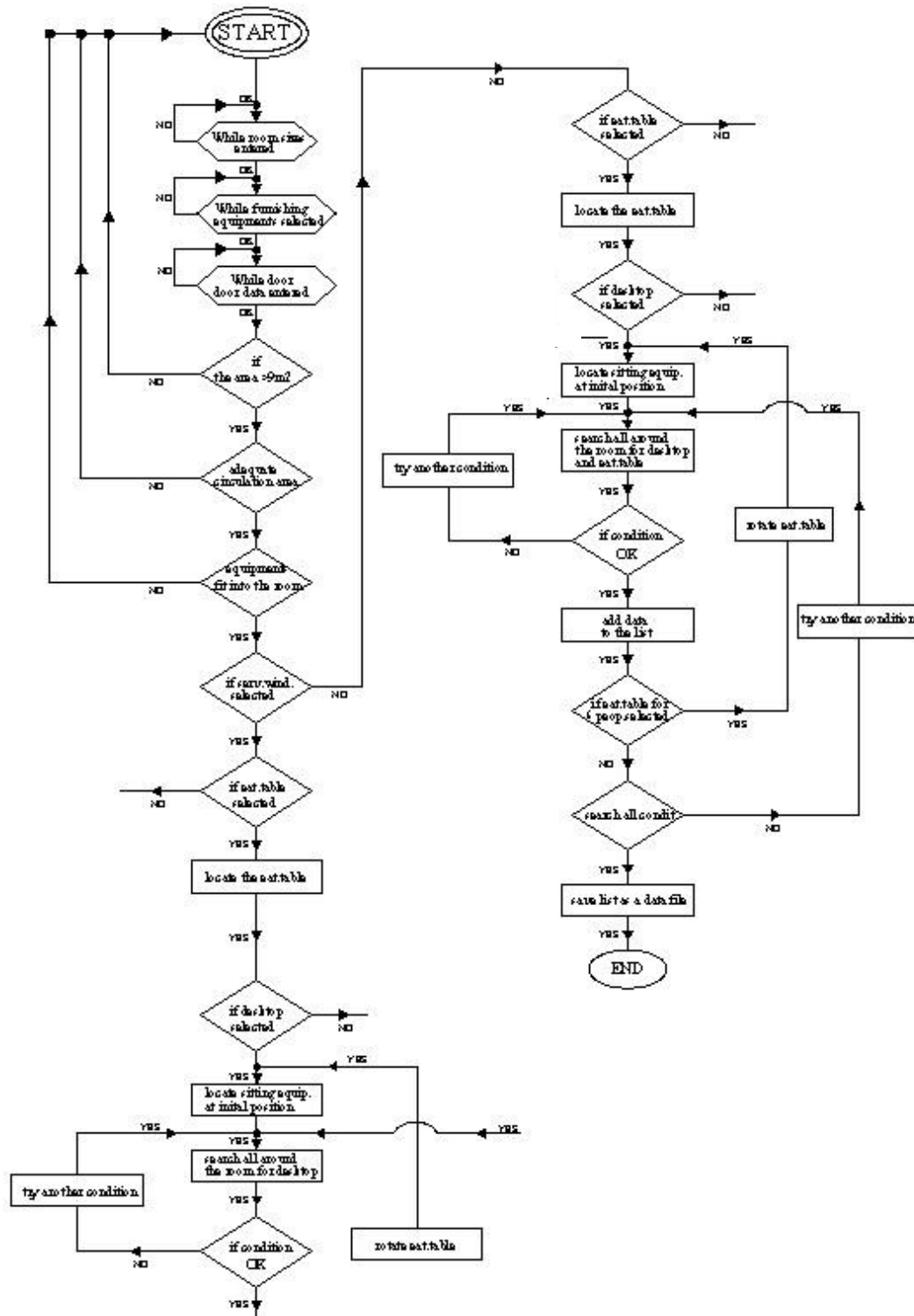


Figure 5.30 Flow chart

1. Calculate the area needed for the sitting group:

OturmaGrupAlanHesapla(int uclu, int ikili, int tekli);

This function searches out for the area of selected sitting group. All the information

like area, width, height about the groups is predetermined. In the example structure of data for sofa(3), sofa(2), sofa(1) can be seen. Additionally, each sitting group behaves like an object and has its own features since every object has its own properties. One of these features is its area. First, in order to find the required area, the search engine looks for whether the TV has been selected or not. If selected, it starts to process under “if(tvvar)”. There are five maximum alternatives for the sitting groups. This is why for each of their data five global variables are assigned.

Table 5.2 Sample from the codes, run while calculating the area of the sitting group.

```

if(!tvvar)
{
if(uclu==1 && ikili==1 && tekli==1)
{
oturmagrubu.alternatif.bir.area=(11*scala*scala);
oturmagrubu.alternatif.iki.area=(10.6*scala*scala);
oturmagrubu.alternatif.uc.area=(12.0*scala*scala);

oturmagrubu.alternatif.bir.height=(3.7*scala);
oturmagrubu.alternatif.iki.height=(3.6*scala);
oturmagrubu.alternatif.uc.height=(3.0*scala);

oturmagrubu.alternatif.bir.width=(2.9*scala);
oturmagrubu.alternatif.iki.width=(3.0*scala);
oturmagrubu.alternatif.uc.width=(4.1*scala);

}

if(tvvar)
{
if(uclu==1 && ikili==1 && tekli==1)
{
oturmagrubu.alternatif.bir.area=(14.6*scala*scala);
oturmagrubu.alternatif.iki.area=(14.0*scala*scala);
oturmagrubu.alternatif.uc.area=(16.0*scala*scala);

oturmagrubu.alternatif.bir.height=(3.8*scala);
oturmagrubu.alternatif.iki.height=(3.6*scala);
oturmagrubu.alternatif.uc.height=(4.0*scala);

oturmagrubu.alternatif.bir.width=(4.1*scala);
oturmagrubu.alternatif.iki.width=(4.1*scala);
oturmagrubu.alternatif.uc.width=(4.2*scala);
}
globarea1=oturmagrubu.alternatif.bir.area;
globarea2=oturmagrubu.alternatif.iki.area;
globarea3=oturmagrubu.alternatif.uc.area;
globarea4=oturmagrubu.alternatif.dort.area;
globarea5=oturmagrubu.alternatif.bes.area;

globheight1=oturmagrubu.alternatif.bir.height;
globheight2=oturmagrubu.alternatif.iki.height;
globheight3=oturmagrubu.alternatif.uc.height;
globheight4=oturmagrubu.alternatif.dort.height;

```

```

globheight5=oturmagrubu.alternatif.bes.height;
globwidth1=oturmagrubu.alternatif.bir.width;
globwidth2=oturmagrubu.alternatif.iki.width;
globwidth3=oturmagrubu.alternatif.uc.width;
globwidth4=oturmagrubu.alternatif.dort.width;
globwidth5=oturmagrubu.alternatif.bes.width;

```

2. Locate the eating table:

YemekMasasiYerlestir();

First, the search engine looks for whether or not the serving window is selected, while settling the eating table. If selected, the engine determines the location automatically through the defined formulation in the codes. If it is not selected it locates the table temporarily to continue its search. It is searching for other suitable places. The eating table also behaves like an object and “yemekmasasi.X1” is one of the classifications of the object property. In the example, the dimensions for two different kinds of eating tables can also be seen.

Table 5.3 Sample from the codes, run when eating table is selected.

```

if(altiliyemek && yatay)
{
    yemekboyutX=3*scala;
    yemekboyutY=2*scala;
    yemekyariboyut=1.5*scala;
    kaymaX=0.5*scala;
    kaymaY=0.5*scala;
}
if(altiliyemek && !yatay)
{
    yemekboyutX=2*scala;
    yemekboyutY=3*scala;
    yemekyariboyut=1*scala;
    kaymaX=0.5*scala;
    kaymaY=0.5*scala;
}
if(dortluyemek)
{
    yemekboyutX=2*scala;
    yemekboyutY=2*scala;
    yemekyariboyut=1*scala;
    kaymaX=0.5*scala;//left deki degisimler icin
    kaymaY=0.5*scala;// top daki degisimler icin
}
if(servis_penceresi->ModalResult==mrOk)
if(kose=="A")
{
    if(kenar=="A--B")

        yemek_masasi->Top=odataban->Top+kaymaY;
        yemek_masasi->Left=odataban->Left+(scala*uzak)+((genis*scala)/2)-(yemekyariboyut);
        yemek_masasi->Width=yemekboyutX;
        yemek_masasi->Height=yemekboyutY;

```

```

}
yemekmasasi.X1=yemek_masasi->Left;
yemekmasasi.Y1=yemek_masasi->Top;
yemekmasasi.X2=yemek_masasi->Left+yemek_masasi->Width;
yemekmasasi.Y2=yemek_masasi->Top+yemek_masasi->Height;
yemek_alan=(yemekmasasi.X1-yemekmasasi.X2)*(yemekmasasi.Y1-yemekmasasi.Y2);
y_genis=yemekmasasi.X1-yemekmasasi.X2;
y_yuksek=yemekmasasi.Y1-yemekmasasi.Y2;

```

3. Control of the meeting points:

ObjelerBirbiriyleCakistimi(int X11,int X12,int Y11,int Y12,int X21,int X22,int Y21,int Y22)

The main idea of this function is to investigate the intersection between the coordinates of any two objects. If there are any meeting points, objects overlapping with one another, the engine “returns false” and skips this settlement to continue with other one. Moreover, each of the two objects from the user selections are grouped and investigated in the given format. It is checking to see if these two objects coincide or not. This can be seen in the example.

Table 5.4 Sample from the codes, run in the control of meeting points.

```

if(X11>X12)
{
tmp=X11;
X11=X12;
X12=tmp;
}
if(Y11>Y12)
{
tmp=Y11;
Y11=Y12;
Y12=tmp;
}
if(X21>X22)
{
tmp=X21;
X21=X22;
X22=tmp;
}
if(Y21>Y22)
{
tmp=Y21;
Y21=Y22;
Y22=tmp;
}
for(k=X21-alternate;k<X22+alternate;k++)
for(m=Y21-alternate;m<Y22+alternate;m++)
{
if((k>=X11 && k<=X12) && (m>=Y11&& m<=Y12))
{
//ShowMessage("İKİ NESNE BİRBİRİ İLE ÇAKIŞIYOR ");

// yemek_masasi->Visible=false;

```

```

        return false;
    }
}
return true;

for(yemek_Xindex=(odataban->Left+(0.1*scala));yemek_Xindex<=odataban->Left+odataban-
>Width-(0.1*scala)
-(YEMEKGEN*scala);yemek_Xindex++)//yemek masasinda (ab) yatayda kayma
{//YEMEK MASASI AB KENARINDA KAYIYOR

    yemeksolX=yemek_Xindex;

    yemeksagX=yemek_Xindex+(YEMEKGEN*scala);
    yemeksolY=(odataban->Top)+(0.1*scala);
    yemeksagY=odataban->Top+(0.1*scala)+(YEMEKYUK*scala);

if(ObjelerBirbiriyleCakistimi(oturma_index,oturma_index+abs(X1-
X2),Y1,Y2,yemeksolX,yemeksagX,yemeksolY,yemeksagY))//oturma ve yemek masasi cakisma
kontrol
if(ObjelerBirbiriyleCakistimi(calismasolX,calismasagX,calismasolY,calismasagY,kapi.X1,kapi.X2,kapi.Y1,kapi.Y2))//calisma ve kapi cakismasi
if(ObjelerBirbiriyleCakistimi(calismasolX,calismasagX,calismasolY,calismasagY,yemeksolX,yemeksagX,yemeksolY,yemeksagY))//calisma ve yemek cakismasi
if(ObjelerBirbiriyleCakistimi(oturma_index,oturma_index+abs(X1-
X2),Y1,Y2,calismasolX,calismasagX,calismasolY,calismasagY))//oturma ve calisma cakisma
if(ObjelerBirbiriyleCakistimi(yemeksolX,yemeksagX,yemeksolY,yemeksagY,kapi.X1,kapi.X2,kapi.Y1,kapi.Y2))//yemek kapi cakismasi kontrol

```

4. Locate the equipment of sitting group and desktop:

OturmagrubuCalismaMasasiYerlestir()

This is the primary function. It becomes active after the button “approve” is selected. It guides the other sub-functions. This means that it arranges when the others will work and begin to use the related codes. This is important due to the great amount of data that is involved and used by these other functions. This main function helps the other sub-functions to search for the related pre-determined data of the objects with which they will work. To summarize, the engine knows the user selections then it explores for the related codes to set up and call the other functions as shown in the example.

Table 5.5 Sample from the codes, run after the ‘Approve’ button is clicked.

```

if(uclunum==1 && ikilinum==1 && teklinum==1)//1-1-1-1
{
//1-1-1-1
if(tvvar)
{
//3tane
if(ymasasi && serpenvar)
{
//ymasasi && serpenvar
if(calisma || !calisma)
{
//calisma

```

```

BosAlanaSigiymu(globarea1);
if(BosAlanaSigiymu(globarea1)==true)
{
    KenarABTeldeYerles(globheight1,globwidth1);

    KenarACTeldeYerles(globheight1,globwidth1);
    KenarBDTeldeYerles(globheight1,globwidth1);
    KenarCDTeldeYerles(globheight1,globwidth1);
}
if(!tvvar)
{
    //3tane
    if(ymasasi && serpenvar)
    { //(ymasasi && serpenvar)
        if(calisma || !calisma)
        { //(calisma)
            BosAlanaSigiymu(globarea1);
            if(BosAlanaSigiymu(globarea1)==true)
            {
                KenarABTeldeYerles(globheight1,globwidth1);
                KenarACTeldeYerles(globheight1,globwidth1);
                KenarBDTeldeYerles(globheight1,globwidth1);
                KenarCDTeldeYerles(globheight1,globwidth1);
            }
        }
    }
}

```

5. Decide on whether the selections fit into the empty space:

BosAlanaSigiymu(int gelenoturma alan)

At the beginning, engine starts to calculate the data through the given formula in order to get the results concerning the suitability of a design selection. Every selection and their settlement alternatives (rotated not rotated) have different areas. The engine gets the data and puts it into that formula. It then decides which one of the alternatives or the objects can be located. The necessary data for this formula is listed in the example. These are: if selected; area of eating table, desktop, sitting group, the area that the objects should not be close to walls, sweeping area of door, circulation area. If the total of these is greater than the room area then, the engine returns false. This means that no results can be generated with the selected equipment.

Table 5.6 Sample from the codes, run while calculating the empty space and deciding on whether the selections fit into the room.

```

{
int kullanilan_alan,area,kalan_alan,kenarayanasma_alan,sirkulasyon_alani;
area=(odataban->Height*scala)*(odataban->Width*scala);//odanın alanı hesaplanıyor
kullanilan_alan=0;
if(ymasasi==true) kullanilan_alan+=yemek_alan;//global
if(kapivar==true) kullanilan_alan+=kapi_alan;//global
if(calisma==true) kullanilan_alan+=(1.7*3600);
kenarayanasma_alan=(odataban->Width*10*2*scala)+((odataban->Height-20)*10*2*scala);

```

```
sirkulasyon_alani= (area*0.25);
kullanilan_alan+=sirkulasyon_alani;
kalan_alan=area-kullanilan_alan;
if(kalan_alan>=gelenoturma_alan) return true;
else if(kalan_alan<gelenoturma_alan) return false;}
```

6. Searching each edge to find out reasonable locations for the sitting activity:

KenarABTeldeYerles(int oturma_hyuksek, int oturma_wgenis);

This function is the way of locating the sitting groups within their rotated and not rotated positions. The data of the rectangle referring to the layout of the sitting group in that system is clearly defined below. All of this data is needed to introduce the data-flow scheme to the engine and to explain how the engine will process the data (X and Y coordinates of the first and last corners of rotated and not rotated alternatives). As the engine gets the context of the sitting group from the user, it starts searching to find out the related data in order to put it into these formulas. Consequently, the compiled data refers to the location of that dynamic object. Number 6,7,8,9 are related functions, only the searching edges are different as mentioned in the given names.

7. KenarACTeldeYerles(int oturma_hyuksek,int oturma_wgenis);

8. KenarBDTeldeYerles(int oturma_hyuksek,int oturma_wgenis);

9. KenarCDTeldeYerles(int oturma_hyuksek,int oturma_wgenis);

Table 5.7 Sample from the codes, run to explore suitable locations on the edge AB for the sitting group.

```
KenarABTeldeYerles(int oturma_hyuksek, int oturma_wgenis);
rotatesizX1=(odataban->Left)+0.1*scala;
rotatesizY1=(odataban->Top)+0.1*scala;
rotatesizX2=odataban->Left+(0.1*scala)+oturma_wgenis;
rotatesizY2=odataban->Top+(0.1*scala)+oturma_hyuksek;
rotateliX1=(odataban->Left)+0.1*scala;
rotateliY1=(odataban->Top)+0.1*scala;
rotateliX2=odataban->Left+(0.1*scala)+oturma_hyuksek;
rotateliY2=odataban->Top+(0.1*scala)+oturma_wgenis;
```

10. Searching for reasonable locations for desktop:

CalismaMasasiKenarABYerles (int X1,int Y1,int X2,int Y2, AnsiString Kenar);

The data-process logic of this function is similar to the functions in number 6,7,8,9. If the desktop is selected by the user from the combo-box, the engine automatically gets the data and calculates for the location with the help of the functions in number

10,11,12,13. All of these together, 6, 7, 8, 9, 10, 11, 12, 13 are interrelated with each other. When the engine locates the sitting group, it then starts to search all the edges for the desktop. It is searching for suitable places within the given parameters set by the user. For example, the minimum distance between two objects and a comparison of these with the other objects' at the run time. At this time the engine starts calculating and recording the suitable ones into its memory. In addition, there are also the rotated and not rotated alternatives for the desktop. Necessary data like the X and Y coordinates of the cross corners is also necessary while controlling any meeting points with the other objects. Examples of these situations are mentioned below. The data concerning the edge is placed at the end of the given format where the processor will always get these letters, "AB, AC, BD, CD".

11. CalismaMasasiKenarACYerles(int X1,int Y1,int X2,int Y2, AnsiString Kenar);

12. CalismaMasasiKenarBDYerles(int X1,int Y1,int X2,int Y2, AnsiString Kenar);

13. CalismaMasasiKenarCDYerles(int X1,int Y1,int X2,int Y2, AnsiString Kenar);

Table 5.8 Sample from the codes, run to explore suitable locations on the edge AB for the desktop.

```
CalismaMasasiKenarABYerles (int X1,int Y1,int X2,int Y2, AnsiString Kenar);
{
if(calisma && !rotate)
{
CalismaMasasiKenarABYerles(rotatesizX1,rotatesizY1,rotatesizX2,rotatesizY2,"AC");
}
}
if(calisma && rotate)
{
CalismaMasasiKenarABYerles(rotateliX1,rotateliY1,rotateliX2,rotateliY2,"AC");
}
}
if(!calisma && rotate)
{
CalismaMasasiYOKKenarABYerles(rotateliX1,rotateliY1,rotateliX2,rotateliY2,"AC");
}
}
```

15. Searching for reasonable locations for eating table:

YemekmasasiDortKenarDolas(int o_index,int x1,int x2,int y1,int y2,int calsolX,int calsagX,int calsolY,int calsagY,AnsiString O_kenar,AnsiString C_kenar);

When the data about the desktop is processed in the related functions it is now time for the eating table. If the eating table is selected, there is one important point about this object and it involves the serving window. If the serving window is selected, the program automatically finds the location without searching for other location

alternatives by putting the data into the given formula. If the serving window is not selected, a search for the list of suitable places for eating table is needed. To find them, each edge of the plan layout of the room is searched thoroughly by the defined letters below. The location of the table for six people also underlines an important point. This table has a rectangle plan scheme; the codes mentioned in number 16 highlight the details of its position. The rotation will differ according to its edges. Moreover, in the borders of logic, it is much more important to provide a more useable place for other components and circulation area. Therefore, it is suggested that the long edge be placed parallel to the edge of the room in question. The edges AB, CD, horizontal edges of the room, are parallel to the table's long-edge without any doubt.

Table 5.9 Definition of the horizontal edges of the room.

```
if(O_kenar=="AB")
else if(O_kenar=="BD")
{//O_kenar=="BD"
```

However as mentioned in number 16, other formulas are needed for AC and CD, vertical edges of the room, while calculating data for the parallel location of long-edge.

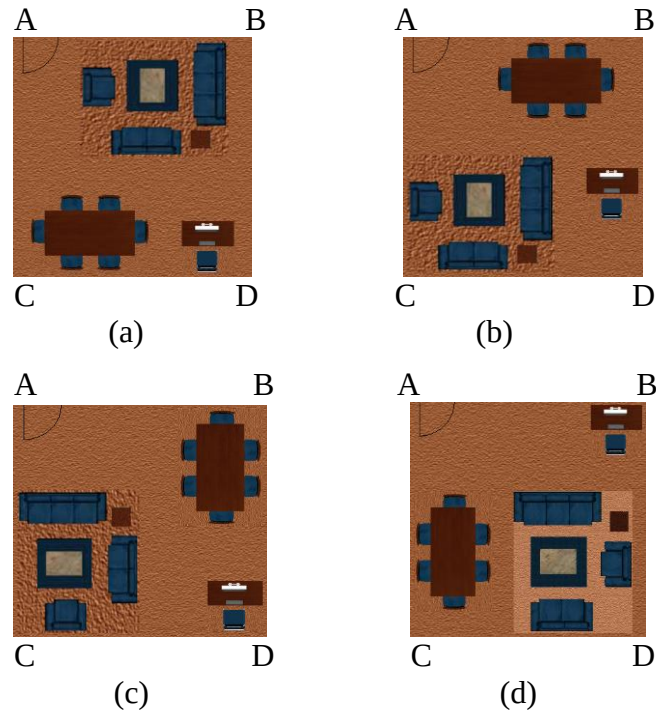


Figure 5.31 (a), (b) Showing the possible horizontal locations for eating table on the edges AB and CD (c), (d) showing the possible vertical locations for eating table on the edges AC and BD.

16. DikeydeYemekmasasiDortKenarDolas

Table 5.10 Definition of the vertical edges of the room.

```
else if(O_kenar=="AC")
{
  //O_kenar=="AC"
  DikeydeYemekmasasiDortKenarDolas(oturma_index,X1,X2,Y1,Y2,calsolX,calsagX,calsolY,calsagY
,O_kenar,C_kenar);
  return;
}
else if(O_kenar=="CD")
```

17. Searching the edges AB, CD to find out reasonable locations for sitting group when the serving window is not selected:

YataySerYokCalYokYemekDortDolas

This function requires detailed codes and formulas than the ones in number 15 and 16 to guide the engine. These will become active if the serving window and desktop are not selected. Functions in number 17 and 18 are nearly the same, except the place of the searching index which means one searches the edge in horizontal line; other searches the edge in vertical line. In addition to these points, one is the reverse of the other, in dimension which refers to width-height information of table.

Summarily the process of data for the location on AB, CD edges and AC, BD edges, as the rotation differs for the table for six people, width and height information are reversed. However, there is no differentiation for the location of table for four people as it has a square plan. After getting width and height sizes of the table, the engine calculates for any meeting point between selected objects.

Table 5.11 Sample from the codes, used while searching for different location alternatives for eating table rotated, not rotated and verifying the suitable ones.

```
for(yemek_Xindex=(odataban->Left+(0.1*scala));yemek_Xindex<=odataban->Left+odataban-
>Width-(0.1*scala)
-(YEMEKGEN*scala);yemek_Xindex++)//yemek masasinda (ab) yatayda kayma
{
  //YEMEK MASASI AB KENARINDA KAYIYOR

  yemeksolX=yemek_Xindex;
  yemeksagX=yemek_Xindex+(YEMEKGEN*scala);
  yemeksolY=(odataban->Top)+(0.1*scala);
  yemeksagY=odataban->Top+(0.1*scala)+(YEMEKYUK*scala);

  if(ObjelerBirbiriyleCakistimi(oturma_index,oturma_index+abs(X1-
X2),Y1,Y2,kapi.X1,kapi.X2,kapi.Y1,kapi.Y2))//oturma grubu ve kapi cakismasi
  if(ObjelerBirbiriyleCakistimi(oturma_index,oturma_index+abs(X1-
X2),Y1,Y2,yemeksolX,yemeksagX,yemeksolY,yemeksagY))//oturma ve yemek masasi cakisma
kontrol
```

```

if(ObjelerBirbiriyleCakistimi(yemeksolX,yemeksagX,yemeksolY,yemeksagY,kapi.X1,kapi.X2,kapi.Y1,kapi.Y2))//yemek kapi cakismasi kontrol
    if(abs(Y1-Y2)< odataban->Height-(0.2*scala))//oturma grubunun yuksekligi ve oda kontrolu
        {
            //tum kosullar saglandi
            yemek_masasi->Visible=true;
            yemek_masasi->Top=odataban->Top+(0.1*scala);
            yemek_masasi->Left=yemek_Xindex;
            yemek_masasi->Width=(YEMEKGEM*scala);
            yemek_masasi->Height=(YEMEKEYUK*scala);
            //ShowMessage("hi");
            OdayaAitDatalariYaz();
        }
        //tum kosullar saglandi
        yemek_Xindex+=jumper;
    }
    //YEMEK MASASI AB KENARINDA KAYIYOR

```

18. Searching the edges AC, BD to find out reasonable locations for eating table when the serving window is not selected:

DikeySerYokCalYokYemekDortDolas

19. Search the edges AC and BD to find out reasonable locations for eating table when the eating table is not selected:

DikeySerYokYemekYokCalVarDortDolas

Another combination through user selections is the selection of sitting group and desktop. Since there will be no eating table, the serving window should not be selected as mentioned in the given name of the function. This function becomes active when the sitting group is located in a place on an edge of the room temporarily and is used to allow the desktop to circulate all of the edges. By then, the suitable alternatives are recorded into the memory and sitting group moves on that edge again to let the desktop circulate all the edges over again. During this circulation, the number of the alternatives in the list can differ through the selected dimensions according to the search frequency.

20. Search the edges AB and CD to find out reasonable locations for the sitting group when the eating table is not selected:

YataySerYokYemekYokCalVarDortDolas

In this function, sitting group's left coordinate becomes the searching index as it changes while moving on the horizontal edges of the room, AB, CD. However when the sitting group moves on the vertical edges of the room, its top coordinate becomes

the searching index. In addition, left coordinate does not change when it is on the edges AC and BD. Related codes about these situations can be seen in number 19 and 20.

21. Display the results on the form:

SonucuFormdaGoster

This function is essential to distinguish the text files produced in run time and the text files of saved favorites. When the user runs the program and if the program generates alternatives, the search engine automatically writes down all data about the alternatives in a text file. In addition to this there is also an option which offers the user the ability to save satisfactory alternatives in a text file with a given name. Every row in that text file is temporarily given a number. The index part in codes starts with zero and the engine while loading from file, adds one to that number as the first row begins with “1” every time. This index is also necessary to call and read the lines with a given number, and also get the data in the separated parts by the character “€”. As the engine knows flow of data in lines by counting characters, it can group the data defining objects. At last, by using the recorded and grouped data, the search engine becomes available to show both the favorites and the all generated alternatives on the form.

Table 5.12 Sample from the codes, run while viewing the results on the form.

```
if(!yukle)
{
    filedir=locd+"odamiz.txt";
    if(FileExists(filedir))
    {
        sonucbilgileri->LoadFromFile(filedir);
        globalsayi=sonucbilgileri->Count;
    }
}

else if(yukle)
{
    if(FileExists(yuklepath))
    {
        sonucbilgileri->LoadFromFile(yuklepath);
        globalsayi=sonucbilgileri->Count;
        // SonucuFormdaGoster(0);
    }
}
```

22. Classify the data of each equipment selected:

parcala

To introduce the special character, "€", to the engine, this function is needed. Additionally, it is the way to explain the program to search for the data between two of these characters and record it as the property of the object in question. More clearly, when the engine sees the first character, it starts counting and when it comes across with the second one, it stops counting and records the counted data referring to one object. In given format; data between first two from the beginning defines the door and the others can be listed systematically as: room; sitting group; desktop; eating table and serving window and windows.

Table 1.13 Sample from the codes, run while classifying data of each object.

```
sayac=index;  
sayacsatir=sonucbilgileri->Strings[index];  
parcala(sayacsatir,"€");
```

23. Display the door:

```
KapiyiYerineKoy(info[0],info[1],StrToInt(info[2]),StrToInt(info[3]));
```

24. Display the image of sitting group:

```
OturmaGrubununResminiKoy(StrToInt(info[10]),StrToInt(info[11]),option);
```

When the new room button is clicked the functions in number 23, 24, 25 become active and refresh the picture of the objects on the plan layout through selections. Openly, number 23 is needed to refresh the picture of the door; number 24 for sitting group's picture; and the last, number 25 for eating table's picture. Dislike other objects door is drawn in the run time and consists of one line and an arc. In order to erase the applied one, door drawing is also needed to be refreshed as the parameters can be changed in next applications.

25. Display the image of eating table:

```
OturmaYemekMasasiResminiKoy(StrToInt(info[18]),StrToInt(info[19]));
```

26. Save the data of the objects in a text file:

```
OdayaAitDatalariYaz
```

This function is necessary to let the engine write the information of the objects in a text file. All the data seen below is written in an order. This categorization helps the

engine to call back the recorded data easily and make the Lisp available to read all these with the aim of converting them to drawings in a CAD environment. When the lisp opens, it counts the characters to get the required data; first row up to the character except for numbers and the letters of the alphabet defines the data of door; second defines the room's; third defines sitting group's; fourth defines the desktop's; fifth defines the eating table's and the last three from the end defines the selectable three window's. Some of the objects may not be selected as seen above; in that case all space left for data of the object in question will be filled by "0" to mention that 'there is not such an object in the selections'. Additionally every row in that text file refers to a location alternative and the number of the rows listed is the sum of the alternatives which is also shown to user on the interface.

Table 5.14 Sample from the codes, run while saving data of each object in a text file.

```
int
yaz_odaleft,yaz_odatop,yaz_odawidth,yaz_odaheight,yaz_grupleft,yaz_gruptop,yaz_grupwidth,yaz_g
rupheight;
int
yaz_calismaleft,yaz_calismatop,yaz_calismawidth,yaz_calismaheight,yaz_ymasaleft,yaz_ymasatop,ya
z_ymasawidth,yaz_ymasaheight;
int
yaz_spenleft,yaz_spentop,yaz_spenwidth,yaz_spenheight,yaz_pen1left,yaz_pen1top,yaz_pen1width,y
az_pen1height;
int
yaz_pen2left,yaz_pen2top,yaz_pen2width,yaz_pen2height,yaz_pen3left,yaz_pen3top,yaz_pen3width,
yaz_pen3height;
int yaz_pen4left,yaz_pen4top,yaz_pen4width,yaz_pen4height;
if(calisma==true)
{
yaz_calismaleft=calismamasa->Left;
yaz_calismatop=calismamasa->Top;
yaz_calismawidth=calismamasa->Width;
yaz_calismaheight=calismamasa->Height;
}
else
{
yaz_calismaleft=0;
yaz_calismatop=0;
yaz_calismawidth=0;
yaz_calismaheight=0;
}
satir1=yaz_kapikose+"€"+yaz_kapikenar+"€"+IntToStr(yaz_kapiuzaklik)+"€"+IntToStr(yaz_kapigeni
slik)+"€";
satir2=IntToStr(yaz_odaleft)+"€"+IntToStr(yaz_odatop)+"€"+IntToStr(yaz_odawidth)+"€"+IntToStr(
yaz_odaheight)+"€";
satir3=IntToStr(yaz_grupleft)+"€"+IntToStr(yaz_gruptop)+"€"+IntToStr(yaz_grupwidth)+"€"+IntTo
Str(yaz_grupheight)+"€";
satir4=IntToStr(yaz_calismaleft)+"€"+IntToStr(yaz_calismatop)+"€"+IntToStr(yaz_calismawidth)+"€
"+IntToStr(yaz_calismaheight)+"€";
satir5=IntToStr(yaz_ymasaleft)+"€"+IntToStr(yaz_ymasatop)+"€"+IntToStr(yaz_ymasawidth)+"€"+I
```

```

ntToStr(yaz_ymasaheight)+"€";
satir6=IntToStr(yaz_spenleft)+"€"+IntToStr(yaz_spentop)+"€"+IntToStr(yaz_spenwidth)+"€"+IntToStr(yaz_spenheight)+"€";
satir7=IntToStr(yaz_pen1left)+"€"+IntToStr(yaz_pen1top)+"€"+IntToStr(yaz_pen1width)+"€"+IntToStr(yaz_pen1height)+"€";
satir8=IntToStr(yaz_pen2left)+"€"+IntToStr(yaz_pen2top)+"€"+IntToStr(yaz_pen2width)+"€"+IntToStr(yaz_pen2height)+"€";
satir9=IntToStr(yaz_pen3left)+"€"+IntToStr(yaz_pen3top)+"€"+IntToStr(yaz_pen3width)+"€"+IntToStr(yaz_pen3height)+"€";
satir10=IntToStr(yaz_pen4left)+"€"+IntToStr(yaz_pen4top)+"€"+IntToStr(yaz_pen4width)+"€"+IntToStr(yaz_pen4height)+"€";
bilgisatiri=satir1+satir2+satir3+satir4+satir5+satir6+satir7+satir8+satir9+satir10;
odabilgileri->Add(bilgisatiri);
sayi=IntToStr(odabilgileri->Count);
alternatif->Caption=" Alternatif Sayisi ..." +sayi;

```

5.2 Limitations

The living room is one of the most important units and the main living area in a home environment. Its characteristics may change depending on the needs and desires of its user. This is often the direct result of the different pieces of movable furniture it contains. It is possible to sit, eat and even work in a living room. This room's plan layout may vary in shape and size as dictated by its user. In Turkey, most commonly used living room plan types are rectangular, 'L', 'T' and square shaped. However, within the context of 'total space', square shaped and rectangular plan schemes are becoming more and more popular largely due to the ease with which the equipment in the living room can be arranged. It is for this reason that the system developed accommodates square and rectangular shaped living rooms. However, with some slight code modifications it is possible to make the use the program to generate arrangement alternatives for the 'L' and 'T' shaped living room plans. In addition to this, the interior of the living room can be shaped in dimensions and selection of the movable furniture can be done within the software. Any of the components mentioned below can be chosen as components of the living room: desktop, eating table, sofa, desktop. There are also sub-categories for the selection of each component, which is useful while defining the type of the equipment, such as an eating table for six people or four people. However, the equipment can not be shaped in dimensions. Each piece behaves like an object. Additionally everything the object 'knows' is captured in its variables and everything it can do is expressed in its methods. Since this system is considered to be a drafting tool before moving in

the furniture, it did not seem so important to edit the dimensions of the equipment. When deciding the size of each component within the structure of the system, the most commonly used furnishing components were examined and then a determination was made concerning their size. As a result of this investigation, there does not seem to be noticeable differences in the furnishing products of different firms. However, it is possible to resize them in Auto-CAD after the simulation of any alternative within the system. The layout of the components that belongs to the current alternatives was determined in the CAD environment due to data integration between the furnishing agent and Auto-CAD. Therefore, each of the equipment can be replaced with more detailed drawings with the use of this possibility within the system.

In conclusion, this system offers various furnishing applications. However, it is acknowledged that there are also some limitations which can be considered significant by some authorities. For example, there were two possibilities considered as room shapes while other shapes were evaluated and labeled as not commonly used in recent years. Also, the resize of the components and scaled bitmaps are not available in the system as these applications are supported through the integration of another designing tool Auto-CAD. Although there are the above- mentioned limitations this system can cover most of the furnishing applications and generate many alternatives with slight modifications in code.

5.3 Consequences of the Alternative Generating Approach

It encourages the non-expert homeowners to design their own homes as being the main dynamic concern of the design process and the user of the design application. Furthermore, the design process of home and residential spaces which can be defined as a private establishment offering individuals, families more complex functions of sheltering, should involve the users, mainly being one of the major themes of these subjects. One of the most important places in housing is the living space where the users pass the time habitually for this reason; the living rooms should cover the expected functions in order to survive ongoing or customary behaviors. The social and physiological effects of these places should be built-in design process professionally within the user satisfaction. Due to evaluate the human needs and interrogate the hierarchical importance of different design approaches (theories) such

as meeting physical, social, cultural, technical and contextual requirements, the indoor environments should be designed and furnished in correlation between the user and the designer or the interactive interface of the expert systems.

Consequently, expert systems, optional computerized tools in design process, can be named as design supporters since they can create nearly the same design solutions that the human designer will achieve. Therefore, the consultation environment of these systems can be used by non-experts to obtain expert knowledge and advice. The interfaces of such data-processing systems developed to provide any subject related to home design is of great importance for the interaction with the user. In conclusion, computer based alternative generating systems and other design applications, problem solving techniques should be improved to extend the design process to the public and compile various multi-disciplined inputs of design proficiently.

5.4 Results, Applications and Impacts

This system also allows the users (expert and non-expert) to furnish their own living spaces, related to the chosen furniture for the living room. In the design process of this software, user can choose the conditions where the designer is free to innovate. However the design process also consists of more routine cases where the system works tightly to analyze the inputs and integrate them with the knowledge base of furnishing analysis of living rooms. The knowledge used to assemble the relations of the furniture with each other and the living room, and the analysis of their inputs was extracted from related books and projects. In addition to that, the structure of the data-searching and processing system was developed covering most of the related aspects of furnishing analysis representing all the possible relevant situations of chosen components. The system aims to find the all possible answers in a short run-time, determine the consequences of a user action thorough the given parameters and physical variables involved in each aspect of the room and develop indices to combine all the data in various ways which relate to people's responses.

6. GENERAL RESULTS

This study outlines some features of the alternative generating system, “furnishing assist”, that provide both interactivity and designing support in the preliminary stages of designing. These features help in exploring the design space for various settling compositions and maintain the integrity of relations in the design composition. The purpose of this system is to assist through a form of collaboration with the designer in designing and producing many solutions.

Considering the use of applicability conditions to maintain the design integrity adds a new dimension to design. One of the main concerns in design is the imposition of constrained relationships, semantics, on the shape of objects which enables the settlement manipulation by interrogating some fixed relations of objects to each other and also their locations of other objects. The potential benefits of this system are to bring expert knowledge within the reach of less experienced users, thus reducing the need for an experienced professional in this area to attend to every step of analysis. Also this system would save a considerable amount of man hours and would allow the user to preserve the expert’s valuable knowledge and provides a new way to preserve knowledge. By using this system, appropriate decisions and accurate design solutions could be made quickly. Moreover, integration with a CAD system offers an opportunity to customize the objects by using different drawing techniques.

Going one step further, this system can be more notable if the computer generated images of products can be re-dimensioned, re-colored, indeed re-formulated, as this opportunity will exist in the case of many home design products. For example, when the manufacturer constructs the images in a computer application that relates to his own design, then these images could be the designs themselves that the customer modifies. Moreover, since the design files can be used as generators for a CAD system, these can be improved as supporters for the customer’s final choice to be sent directly to the furnishing process as a bespoke item. Furthermore the customer may even design textile prints or decide on wood type for each object by selecting or

modifying the current ones. Such a process could be characterized as an alternative design process where the consumer uses technology to execute the final design, remote from the design studio.

There is one more significant point about the system since it can involve both the marketing behavior of suppliers and the shopping habits of the consumers. First of all consumer can specify from a wide range of color, wood and texture as well as a catalogue of furnishing items. Secondly, these items can be included as scaleable bitmaps; that's why it could only be a short step to manufacturers providing their own catalogues of such items for the consumer to try out the merchandise in their simulated home. Finally, the purchase of the final choice can then be fed into the supplier's order system, which may in turn be linked to the manufacturer's stock and production systems.

Consequently, this bewildering array of perspectives suggests a more comprehensive design tool through furnishing. Additionally, main concern of this future work is to provide useful designing support to the users while during their variant activities. These activities that the system can be developed for can be summarized as: the activities common to anyone involved in home design and the professional activities found in commercial production. As a result; the system, which will be developed for the first activity, as the next step of the current one will not only contain furnishing of the living room but also the rest of the home environment and the system for the second one will make it possible to accommodate merchandise, which will be chosen from the firm's catalogue and could be modified within the user's parameters, and simulate the furnishing.

REFERENCES

Adams, L., 1993. Windows Visualization Programming with C/C++: 3D Visualization, Simulation, and Virtual Reality, Blue Ridge Summit, Windcrest/McGraw-Hill, PA.

Akin, Ö., 1989. Psychology of Architectural Design, pp. 158-170, Pion, London.

Akin, Ö., 2000. Case Based Instruction Strategies in Architecture.
Digital PDF version <http://www.andrew.cmu.edu/user/oa04/Papers/CaseBased.pdf>

Akin, Ö., Sen, R., Donia, M., Zhang, Y., 1995. SEED-Pro: Computer Assisted Architectural Programming in SEED, *Journal of Architectural Engineering*, 153-161. Digital PDF version <http://www.andrew.cmu.edu/user/oa04/Papers/sp.pdf>

Ammeraal, L., 1995. C++ for Programmers, John Wiley and Sons Ltd., New York.

Anadol, Z., Akin, Ö., 1994. Determining the Impact of CADrafting Tools on the Building Delivery Process, in *International Journal of Construction Information Technology*, 1-8.
Digital PDF version <http://www.andrew.cmu.edu/user/oa04/Papers/CADrafting.pdf>

Anuff, E., 1996. The Java Sourcebook, Wiley Computer Pub., New York.

Arnold, D. M., 1998. Introduction to Programming Using Java : An Object-Oriented Approach, Mass.: Addison Wesley, Boston.

Berk, A. A., 1985. Micro-PROLOG and Artificial Intelligence, Collins, London.

Bijl, A., 1987. An Approach to Design Theory, In: *Design Theory for CAD*, North-Holland, **34**, 3-31.

Berkeley, E. C., Bobrow, D.G., 1974. The Programming Language LISP: Its Operation and Applications, Mass.: The M.I.T., Cambridge, MA.

Björk, B.C., 1989. Basic Structure of a Proposed Building Product Model, *Computer-aided design (CAD)*, Volume **21**, Number 2, 71-78.

Björk, B.C. and Penttilä, H., 1991. Building Product Modeling Using Relational Databases, Hypermedia Software and CAD Systems, *Microcomputers in Civil Engineering*, Volume **6**, 267-279.

Carrara, G., and Kalay, Y. E., 1994. Past, Present, Future: Process, and Knowledge in Architectural Design, *Knowledge Based Computer Aided Architectural Design*, Amsterdam: Elsevier, 389-396.

Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F. and Jeremes, P., 1994. Object Oriented Development: The Fusion Method, Prentice Hall International, Boston.

Copeland, J., 2001. Artificial Intelligence: A Philosophical Introduction, Mass.: Basil Blackwell, Cambridge.

Coyne, R., 1985. Knowledge Based Planning Systems and Design: A Review, *Architectural Science Review*, Volume 4, Number 11, 95-103.

Cumming, M, Akin, Ö, Shealey, M., Tuncer, B., 1996. An Electronic Design Assistance Tool for Case Based Representation of Designs, *proceedings of the 1996 ACADIA Conference* held in Tucson, Arizona, October 31 - November 3. Digital PDF version <http://www.andrew.cmu.edu/user/oa04/Papers/edatACADIA.pdf>

Brown, D. and Birmingham, 'W., 1997. Understanding the Nature of Design, *IEEE Expert*, Volume 12, Number 2, 14-16.

Dale, N., Weems, C., and Headington, M., 1996. Programming and Problem Solving with C++, Jones and Bartlett, Boston.

Dieterich, H., Malinowski, U., Kuhme, T., 1993. Adaptive User Interfaces: Principles and Practice, Amsterdam: North-Holland, 13-48.

Dobra, M., 1996. Artificial Intelligence will Evolve. Digital Version available at http://www.teklearning.com/ai_evo.asp

Dorst, K. and Dijkhuis, J., 1996. Comparing Paradigms for Describing Design Activity, pp. 253-270, John Wiley & Sons, Chichester.

Duerk, D., 1998. Architectural Programming, Van Nostrand Reinhold, New York.

Eastman, C. M., 1991. The Evolution of CAD: Integrating Multiple Representation, *Building and Environment*, Volume 26, Number 1, 17-23.

Flemming, U., 1994. Artificial Intelligence and Design: A Midterm Review, *Knowledge Based Computer Aided Architectural Design*, Amsterdam: Elsevier, 1-24.

Gero, J. S., Radford, A. D., Coyne, R., Akiner, V. T., 1985. Knowledge-based Computer Aided Architectural Design, *Knowledge Engineering in Computer Aided Design*, North Holland, Amsterdam, 57-88.

Gero, J. S., Sudweeks, F., 1998. Artificial Intelligence in Design '98, Dordrecht : Kluwer Academic Publishers, Portugal.

Giarratano, J., and Riley, G., 1998. Expert Systems: Principles and Programming, PWS, Boston.

Hood, J. D., 1990. Using AutoCAD with Auto-Lisp, McGraw-Hill, New York.

Holsapple, C. W., 1996. Decision Support Systems: A Knowledge-Based Approach, West Publishing Company, New York.

Available at <http://www.uky.edu/BusinessEconomics/dssakba/>

Iynizio, J. P., 1991. Introduction to Expert System: The Development & Implementation of Rule Based Expert System, McGraw - Hill Inc., New York,.

Kalay, Y. E., 1985. Redefining the Role of Computer in Architecture: From Drafting Modeling Tools to Knowledge - Based Design Assistants, *Computer Aided Design*, 319 - 328.

Kalay, Y.E., 1992. Principles of Computer-Aided Design: Evaluating and Predicting Design Performance, Butterworths, London.

Kalay, Y. E., 1992. Evaluating and Predicting Design Performance, John Wiley & Sons, Inc., Canada.

Koutamanis, A. and Mitossi, V., 1993. Computer Vision in Architectural Design, *Design Studies*, 23, 40-57.

Kumlin, R., 1995. Architectural Programming: Creative Techniques for Design Professionals, McGraw-Hill, New York.

Lansdown, J., Roast, C., 1987. 'The possibilities and problems of knowledge based systems for design', *Environment and Planning B: Planning and Design*, volume 14, 255-266.

Lesperance, Y., Levesque, H., J., Reiter, R., 1999. A Situation Calculus Approach to Modeling and Programming Agents, *Foundations of Rational Agency*, Kluwer, 275-299. Digital PDF version <http://www.cs.yorku.ca/~{olivier,lesperan,wolfgang}>

Luger, G. F., 1989. Artificial Intelligence and the Design of Expert Systems, Benjamin/Cummings Pub. Co, California.

Maher, M. L., Silva Garza, A. G., Bridge, K., Zhang, D. M., 1997. AI in Design: Case Based Reasoning in Design.

Available at <http://www.computer.org/pubs/expert/1997/features/x2034/x2034.htm>

Markowitz, J., 1995. Speech Recognition Overview: Hitchhiker's Guide to Artificial Intelligence, pp. 56-60, Miller Freeman Publications, California.

McCullough, J., 1991. Knowledge Based Systems in Architecture, Swedish Council for Building Research, Stockholm.

Miano, J., 1997. Borland C++ Builder How-to: the Definitive C++ Builder Problem-Solver, Waite Group Press, Corte Madera, California.

Mitchell, W. J., 1977. Computer-Aided Architectural Design, Petrocelli/Charter, New York.

Pan, X., 2000. An Experimental Adaptive Expert System. Digital PDF version http://www.cadrc.calpoly.edu/pdf/adaptive_expert.pdf.

Pham, D. T., Tacgin, E., 1991. Techniques for Intelligent Computer Aided Design, pp. 5-26, Springer - Verlag, UK.

Reffat, M. R., 1994. Towards an Interactive Intelligent Knowledge Based System for Building Design. Digital PDF version available at http://www.arch.usyd.edu.au/~rabee/publications_files/94ReffatUK.pdf.

Reffat, R. M., Aref, M. M., 1994. A Knowledge Based System for Comfort Analysis of Internal Environment of Hotels, Las Vegas, Nevada, USA. Digital PDF version available at http://www.arch.usyd.edu.au/~rabee/publications_files/94ReffatArefUSA.pdf

Reffat, R. M., 1994. Development of an Expert System for Evaluating the Environmental Quality of Office Buildings, *Master Thesis*, KFUPM, KSA. Digital PDF version available at http://www.arch.usyd.edu.au/~rabee/publications_files/01ReffHarknessJPCF-2.pdf.

Reffat, R. and Gero, J. S., 2000. Towards Active Support Systems for Architectural Designing, *eCAADe*, **18**, Bauhaus-Universitat, Weimar, 143-147. Digital PDF version available at <http://www.arch.su.edu.au/~john/publications/2001pdf/00ReffatGeroeCAADe.pdf>.

Reffat, R., 2001. Can Intelligent Agents Invent Creative Concept, *Proceedings of International Conference on Intelligent Agents, Web Technologies and Internet Commerce -IAWTIC'2001*, Las Vegas, USA, April 249-257. Digital PDF version <http://www.arch.usyd.edu.au/~rabee/publications.htm>

Reynolds, R. A., 1980. Computer Methods of Architects, Butterworths, London.

Rosenman, M. A., 1990. Application of Expert Systems to Building Design Analysis and Evaluation, *Building and Environment*, Volume **25**, Number 3, Great Britain, 221-233.

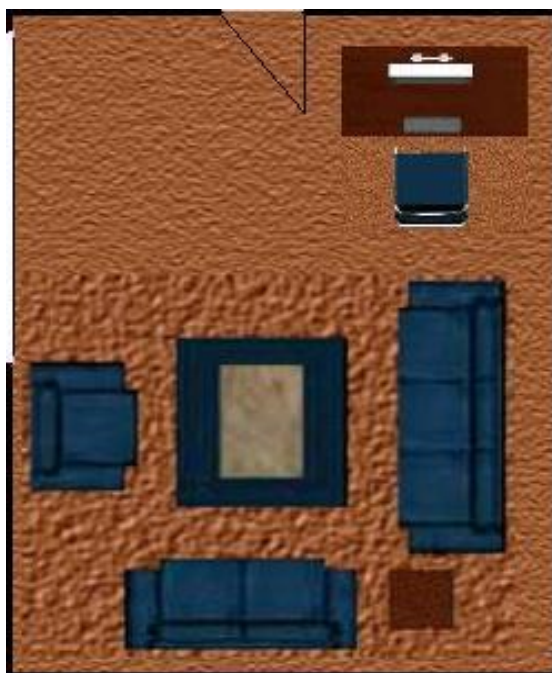
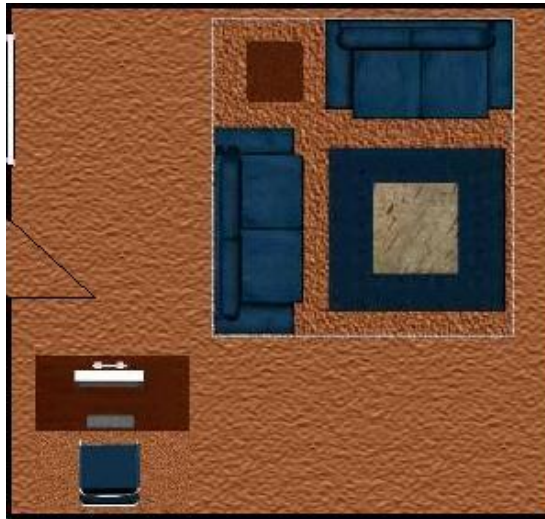
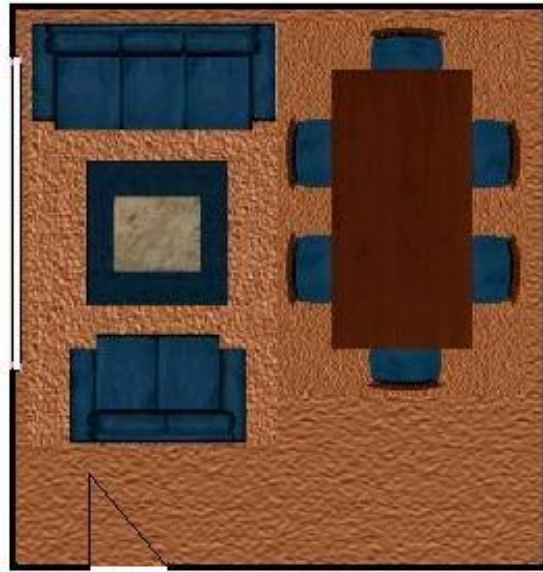
Rossignac, J. R., Borrel, P. and Nackman, L., 1989. Interactive Design with Sequences of Parameterized Transformations, pp. 92-125, Springer-Verlag, Berlin.

Shash, A. A. and Reffat, R. M., 1994. An Expert System as a Proficient Tool for Computer Aided Design and Evaluation of Housing Projects, *submitted to the 22nd International IHAS Symposium on Housing Construction and Financing*, Salzburg, Austria. Digital PDF version <http://www.arch.usyd.edu.au/~rabee/publications.htm>

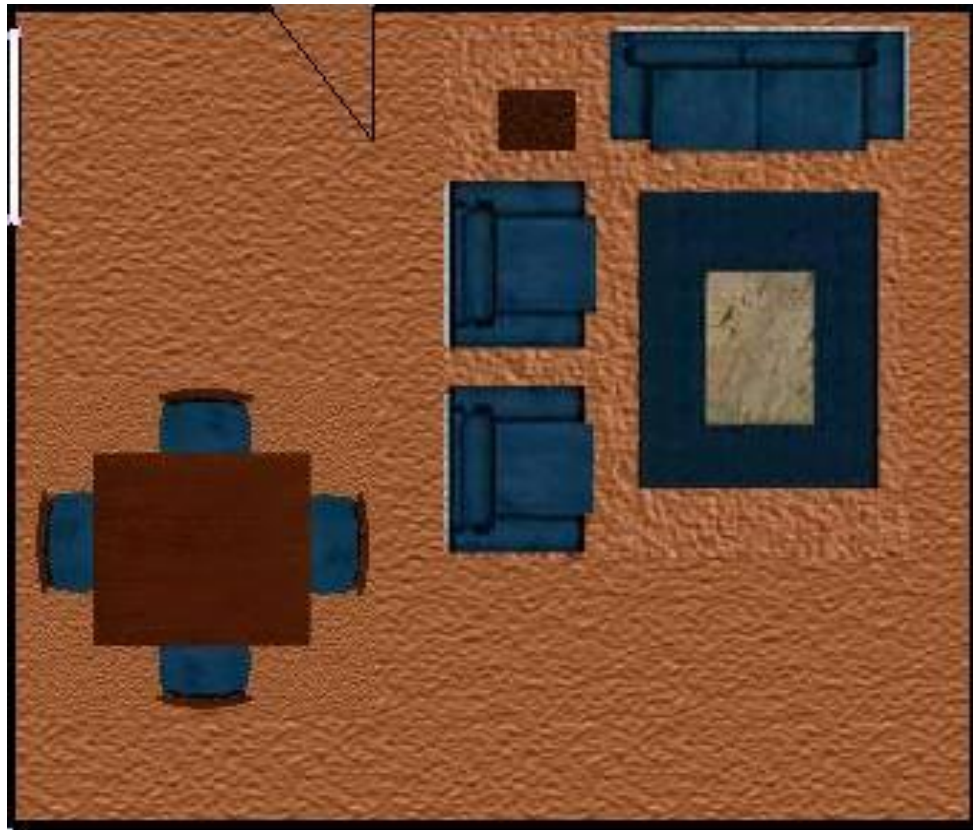
Taşlı, Ş., 1997. Knowledge Representation for Architectural Systems. Digital version <http://www.art.bilkent.edu.tr/iaed/cb/Tasli.html>

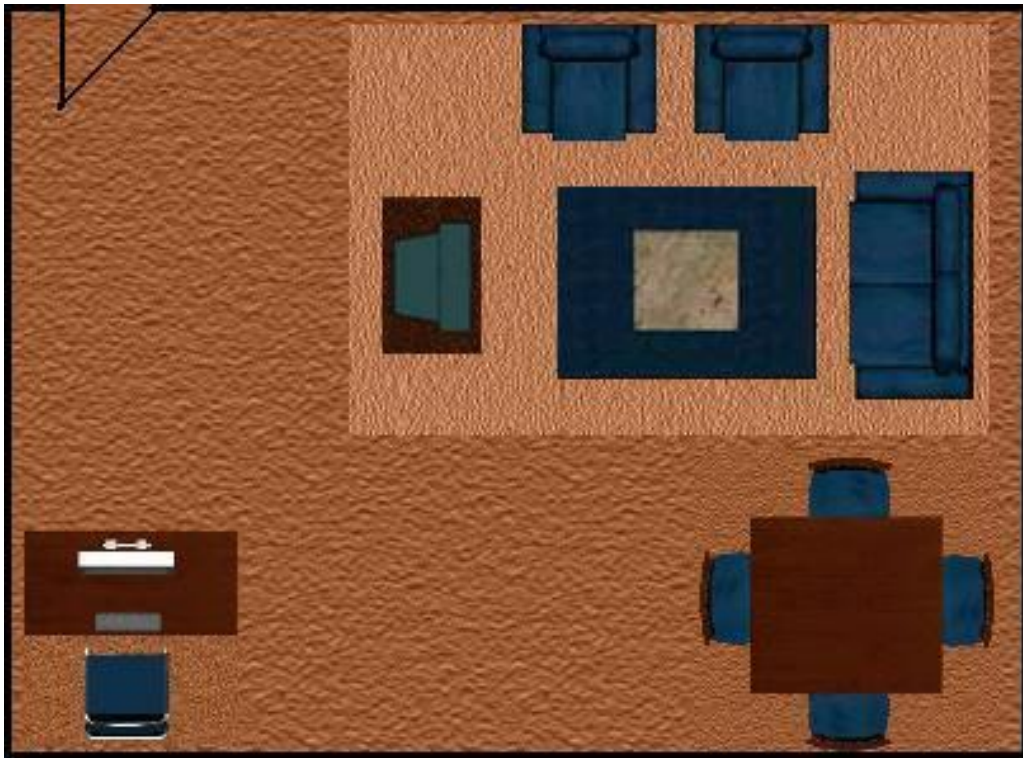
Zreik, K., 1992. What Could Artificial Intelligence Know About the Knowledge Involved in Design Process?, CAAD Futures '91, *Proc. International Conference on Computer Aided Architectural Design Futures*, Zurich, 284 - 386.

APPENDIX-A
WRONG APPLICATIONS OF FURNISHING









APPENDIX-B
FLOW CHART

CIRRICULUM VIATE

Selen Okçu was born in 1980 in Konya. She graduated from Konya Meram Fen Lisesi in 1997. In the same year, she started undergraduate studies in İ.T.Ü Department of Architecture. She graduated from İ.T.Ü in 2001 and started to the graduate program in the same university in Architectural Design Computing Program at the Department of Architecture. She worked as an architect in different companies and she has been working in İ.T.Ü as teaching assistant for six months in Department of Interior Design.