# CONTENTS

ACKNOWLEDGEMENT----------I

ABSTRACT----------II

LIST OF ABBREVIATIONS----------III

CONTENTS----------1

CHAPTER ONE: INTRODUCTION----------3

1.1 INTRODUCTION----------3

1.2 STATEMENT OF THE PROBLEM ----------4

1.3 OBJECTIVES OF THE RESEARCH----------4

CHAPTER TWO: LITERATURE REVIEW----------6

2.1 DRAFTING PRINCIPLES ----------6

2.2 ESSENTIALS OF DRAFTING: LINES AND LETTERING----------8

2.3 METHODS Of EXPRESSION ----------9

2.4 METHODS OF SHAPE DESCRIPTION ----------11

2.5 THE PHASES OF DESIGN----------13

2.6 INTRODUCTION TO PROGRAMMING----------14

2.7 HISTORY OF AutoLISP----------19

2.7.1 AutoCAD Releases----------19

2.7.2 History Of Lisp----------22

2.7.3 AutoLISP'S Roots in Lisp----------23

2.7.4 Basic Continuity----------23

2.8 INTRODUCTION TO AUTOLISP----------25

2.9 TYPES OF PROGRAMMING LANGUAGES----------26

2.10 A FEW SAMPLE DIFFERENCES----------27

CHAPTER THREE: DEVELOPMENT OF CONCEPT----------30

3.1 CUSTOMIZATION CONCEPT----------30

3.1.1 Creating Menu's Customize----------30

3.1.2 Creating Toolbars Customize----------36

3.1.3 Toolbars And Menu Files----------39

3.1.4 Loading Partial Menu's----------41
3.2 THE CUSTOMIZATION AUTOCAD PROGRAM----------47
3.2.1 Lower Die Block----------47
3.2.2 Guide Pin----------50
3.2.3 Assembly Die Blocks----------51
3.2.4 Drawing Of Gear----------52
CHAPTER FOUR: RESULTS & DISCUSSION ----------54
4.1 RESULTS----------54
4.2 DISCUSSION ----------57
CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATION----------60
5.1 CONCLUSIONS ----------60
5.2 RECOMMENDATIONS----------60
REFERENCES----------62

CHAPTER ONE

# INTRODUCTION

## 1.1 INTRODUCTION

The computer has grown to become essential in the operations of business, government, military, engineering, and research. It has also demonstrated itself, especially in recent years, to be a very powerful tool in design and manufacturing. Computer-aided design (CAD) involves and type of design activity which made use of computer to develop, analyze, or modify an engineering design.

Modern CAD systems based on interactive computer graphics (ICG). Interactive computer graphics denotes a user-oriented system in which the computer is employed to create, transforms, and display data in a form of picture or symbol. The user in the computer graphics design system is the designer, who communicates data and commands to the computer through any of several input devices. The computer communicates with the user via a cathode ray tube (CRT). The designer creates an image on the CRT screen by entering commands to call the desired subroutine stored in the computer.

AutoCAD, released by AutoDesk Corporation is the most versatile and universally used CAD system in engineering design. It provides the facility to produce drawing in two, two and half, and three dimensions. Three-dimensional objects can be drawn as wire frames, surface models or solid models as per the requirements for the drawing.

In any CAD system (AutoCAD being one), the designer constructs the graphical image from elementary entities (such as points, lines, circles, etc.) using three types of commands. The first type of commands generates basic geometric elements such as points, lines and circles. The second commands type is used to accomplish scaling, rotation, and other transformation of these

elements. The third type of commands causes the various elements to be joined into the desired shape of the object being created by the CAD system.

What was mentioned in the few previous lines shows that creating an image using a general purpose CAD system, such as AutoCAD, is laborious, tedious and time consuming, and this raises the need for customization of CAD systems.

CAD systems can increase the productivity in the drafting function by roughly five times over manual drafting. The productivity improvement in CAD as compared to traditional drafting depends on factors such as complexity of drawing, level of details required on the drawing, degree of repeatability of designed parts and extensiveness of library of commonly used parts.

Customizing of the drafting system so as to be user friendly to the specific jobs for which it was customized can tremendously increase productivity.

## 1.2 STATEMENT OF THE PROBLEM

The design section of Yarmouk Industrial Complex serves the tools and die factory, other factories of Yarmouk Industrial Complex as well as outside claims. Thus a large work pressure is imposed on this section, forming a bottleneck for the whole production of the complex.

In the mentioned design section, design and drafting are being carried out using the most recent releases of AutoCAD, by the most efficient technicians of the trade.

Something need to be done in order to increase the productivity of this section, so as to cope with the large size of the work required.

## 1.3 OBJECTIVES OF THE RESEARCH

In our complex (Elyarmouk Industrial Complex) there are many CAD work (Patterns, Tools, Dies and molds and different kind of parts) for especial products or for the other customer (eg. Sudan railways, Giad, Atbara Cement Factory, Cold Air Company, Saria Company and many small companies they try

to produce their job inside our complex).

In these jobs we use the AutoCAD to do all drawings in Technical Department Design Section and Tools and Dies Factory.

Due to the plenty of work there are some delaying to our customer in technical department design section and tools and dies factory.

The objective of this work is to customize AutoCAD by using the AutoLISP language to enhance the following:

a. Productivity of CAD.
b. Minimize the time of design.
c. Minimize the cost of design.
d. Increase the profit.
e. Minimize drawings errors, which depend on personal factors.

And thus improving the work methods in the Tools and Dies Factory and Design section in Elyarmouk Industrial Complex, to which the research aimed.

CHAPTER TWO

# LITERATURE REVIEW

## 2.1 DRAFTING PRINCIPLES

In beginning the study of graphics, you are embarking upon a rewarding educational experience and one that wi1l be of real value in your future career. When you have become proficient in it, you will have at your command a method of communication used in all branches of technical industry, a language unequaled for accurate description of physical objects.

Comparing it with word languages can see the importance of this graphic language. All who attend elementary and high school study the language of their country and learn to read, write, and speak it with some degree of skill. In high school and college most students study a foreign language. These word languages arc highly developed systems of communication. Nevertheless, any word language is inadequate for describing the size, shape, and relationship of physical objects. Study the photograph at the opening of this chapter and then try to describe it verbally so that someone who has not seen it can form an accurate and complete mental picture. It is almost impossible to do this, although possibly easier to describe, presents an almost insurmountable problem. Furthermore, in trying to describe either picture, you may want to sketch all or part *to* make the word description more complete, or gesture to aid in explaining shape and relationship. Thus we see that a word language- is often without resources for accurate and rapid communication of shape and size and the relationships of components.

Engineering is applied science, and communication of physical facts *must* be complete and accurate. Quantitative relationships are expressed mathematically. The written word completes many descriptions. But whenever machines and structures are designed, described, and built, graphic

representation is necessary. Although the works of artists (or photography and other methods of reproduction) would provide pictorial representation, they cannot serve as engineering descriptions. Shaded pictorial drawings and photographs are used for special purposes, but the great bulk or engineering drawings are made in line only, with separate views arranged in a logical system of projection. To these views, dimensions and special notes giving operations and other directions for manufacture are added. This is the language of graphics, which can be defined as the graphic representation of physical objects and relationships.

As the foundation upon which all designing and subsequent manufacture are based, engineering graphics is one of the most important single branches of sti.1dy in a technical school. Every engineering student must know how to make and how to read drawings. The subject is essential in all types of engineering practice, and should be under stood by all connected with, or interested in technical industry. All designs and directions for manufacture are prepared by draftsman, professional writes of the language, but even one who may never make drawings must be able to read and understand them or be professionally illiterate. Thorough training in engineering graphics is particularly important for the engineer because he is responsible for and specifics the drawings required in his work and must therefore be able to interpret every detail for correctness and completeness

Our object is to study the language of engineering graphics so that we can write it expressing ourselves clearly to one familiar with it, and read it readily when written by another. To do this, we must know its basic theory and composition, and he familiar with its accepted conventions and abbreviations. Since its principles are essentially the same throughout the world, a person who has been trained in the practices of one nation can readily adapt himself to the practices of another.

This language is entirely graphic and written, and is interpreted by acquiring a visual knowledge of thru object represented. A student's success with it will be indicated not alone by his skill ill execution, but also by his ability to interpret lines and symbols and to visualize clearly in space.

As a background for study, we shall introduce in this chapter various *as*pects of graphics that will be discussed at length later. It is hoped that this preview will serve as a broad perspective against which the student will see each topic, as it is studied, in relation to the whole. Since our subject is a, graphic language, illustrations are helpful in presenting even this introductory material; figures are used both to clarify the text

## 2.2 ESSENTIALS OF DRAFTING: LINES AND LETTERING

Drawings are made up of lines that represent the surfaces, edge, and contours *of* objects. Symbols, dimensional sizes, and word notes are added to these lines, collectively making a complete description. Proficiency in the methods *of* drawing straight lines, circles, and curves, either freehand or

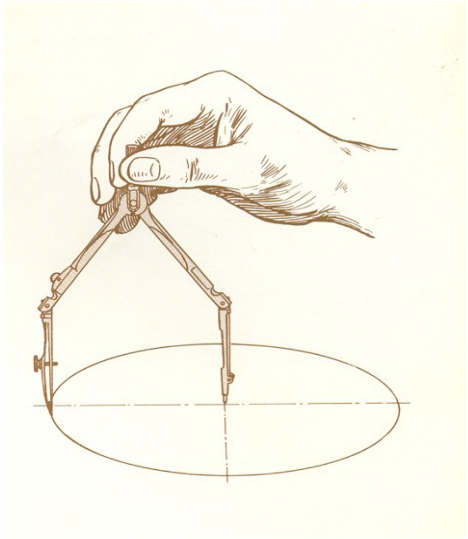

Fig. (2.1) Use of the instruments

with instruments, and the abili ty to letter word statements are fundamental to writing the graphic language. Furthermore, lines are connected according to the geometry of the object represented, making it necessary to know the geometry of

plane and solid figures and to understand how to combine circles, straight lines, and curves to represent separate views of many geometric combinations.

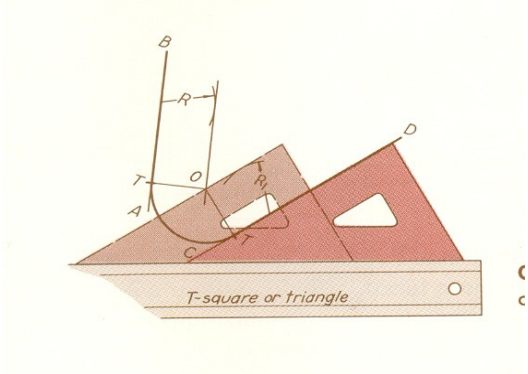


Fig. (2.2) Graphic geometry

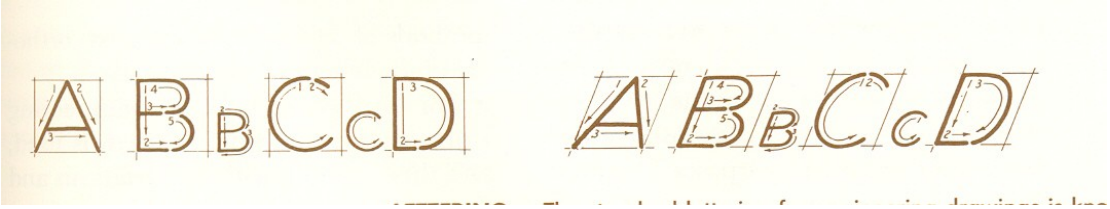

Fig. (2.3) Lettering

## 2.3 METHODS Of EXPRESSION

There are two fundamental methods of writing the graphic language: freehand and with instruments.

Freehand drawing is done by sketching the lines with no instruments other than pencils and erasers. It is an excellent method during the learning process because of its speed and because at this stage the study of projection is more important than exactness of delineation. Freehand drawings are much used commercially for preliminary designing and for some finished work.. Instrument

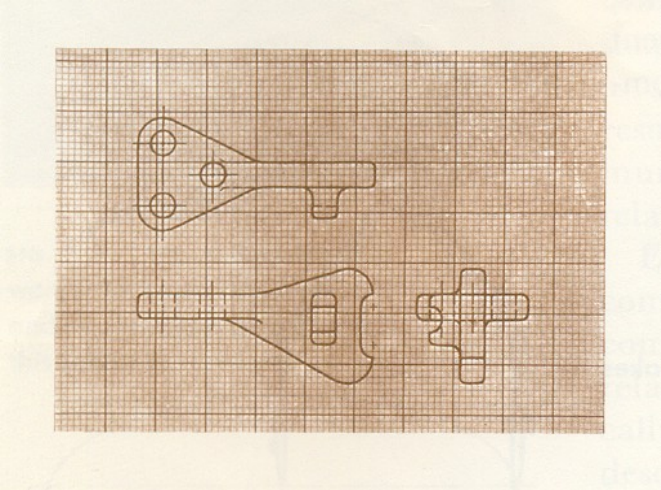

Fig. (2.4) Freehand drawing

drawing is the standard method of expression. Most drawings are made "to scale:' with instruments used to draw straight lines, circles, and curves concisely and accurately. Training in both freehand and instrument work is necessary for the engineer so that he will develop competence in writing the graphic language and the ability to judge work done under his direction.

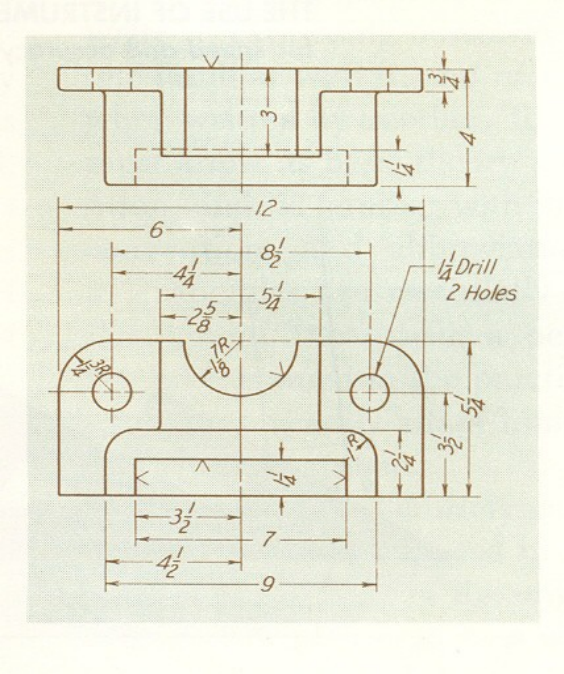


Fig. (2.5) Instrument drawing

## 2.4 METHODS OF SHAPE DESCRIPTION

Delineation of the shape of a part, assembly, or structure is the primary element of graphic communication. Since there are many purposes for which drawings are made the engineer must select, from the different methods of describing shape, the one best suited to the situation al hand. Shape is described by projection, that is, by the process of causing an image to be formed by rays of sight taken in a particular direction from an object to a picture plane.

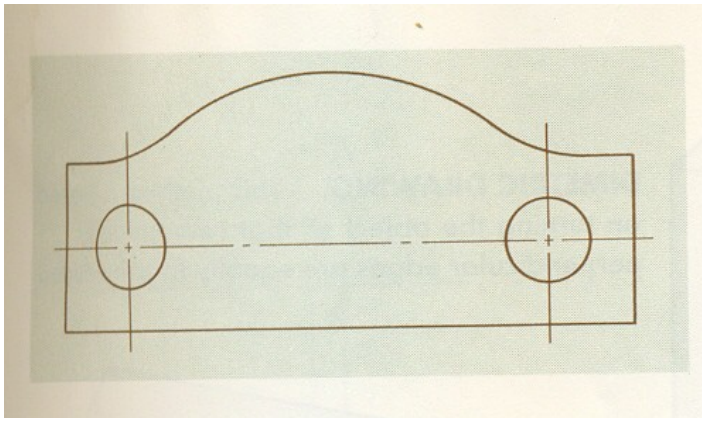

Fig. (2.6) One-view drawing

Following projective theory, two methods of representation are used: orthographic views and pictorial views.

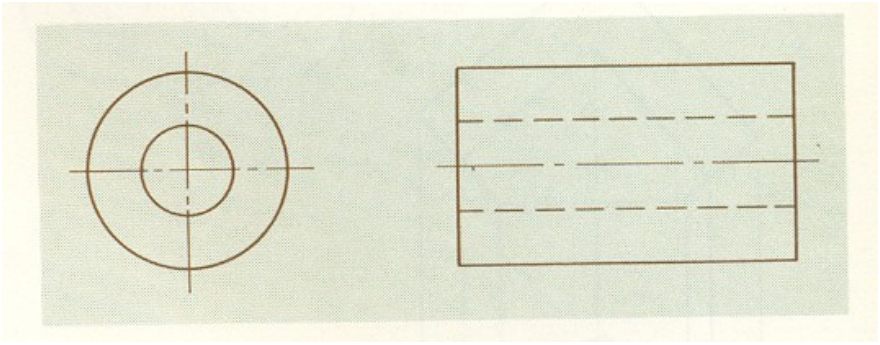

Fig. (2.7) Two-view drawing

For the great bulk of engineering work. the orthographic system is used, and this method, with its variations and the necessary symbols and abbreviations, constitutes an important part of this book. In the orthographic

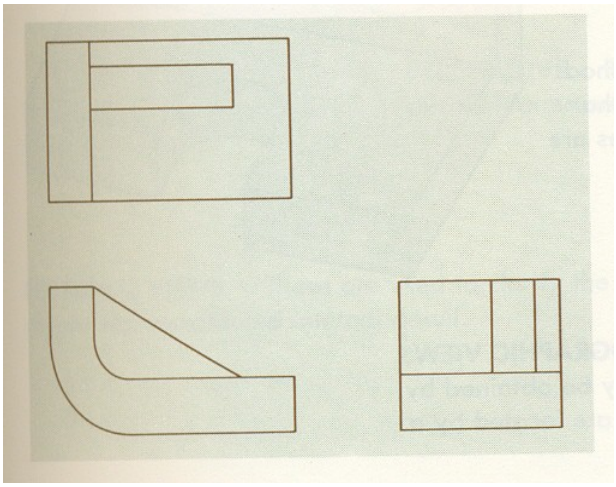

Fig. (2.8) Three-view drawing

System, separate views arranged according to Ill{' projective theory are made to show clearly all details of the object represented. The figures that follow illustrate the fundamental types of orthographic drawings and orthographic views.

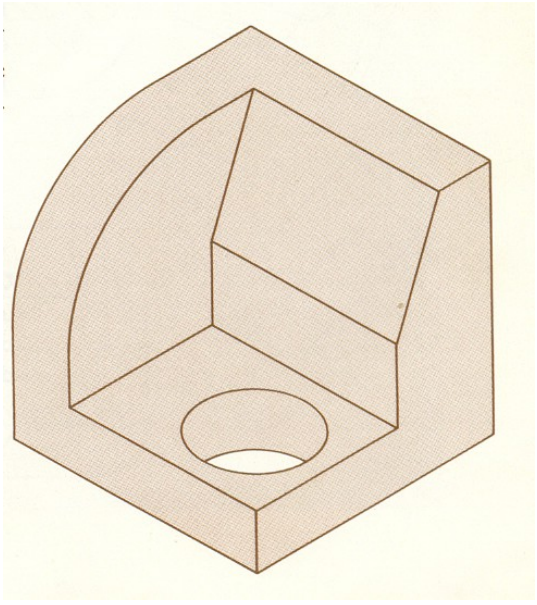

Fig. (2.9) Isometric drawing

“Pictorial representation” designates the methods of projection resulting in a view that shows the object approximately as it would be seen by the eye. Pictorial representation is often used for presentation drawings, text, operation, and maintenance book illustrations, and some working drawings.

## 2.5 THE PHASES OF DESIGN

The total design process is of interest to us in this chapter. How does it begin? Dose the engineer simply sit down at this or her desk with a blank sheet of paper and jot down some ideas? What happens next? What factors influence or control the decisions, which have to be made? Finally, how dose this design process end? The complete process, from start to finish, is often outlined as in Fig. 1-1. The process begins with recognition of a need and a decision to do something about it. After much iteration, the process ends with the presentation of the plans for satisfying the need.[ 5 ]

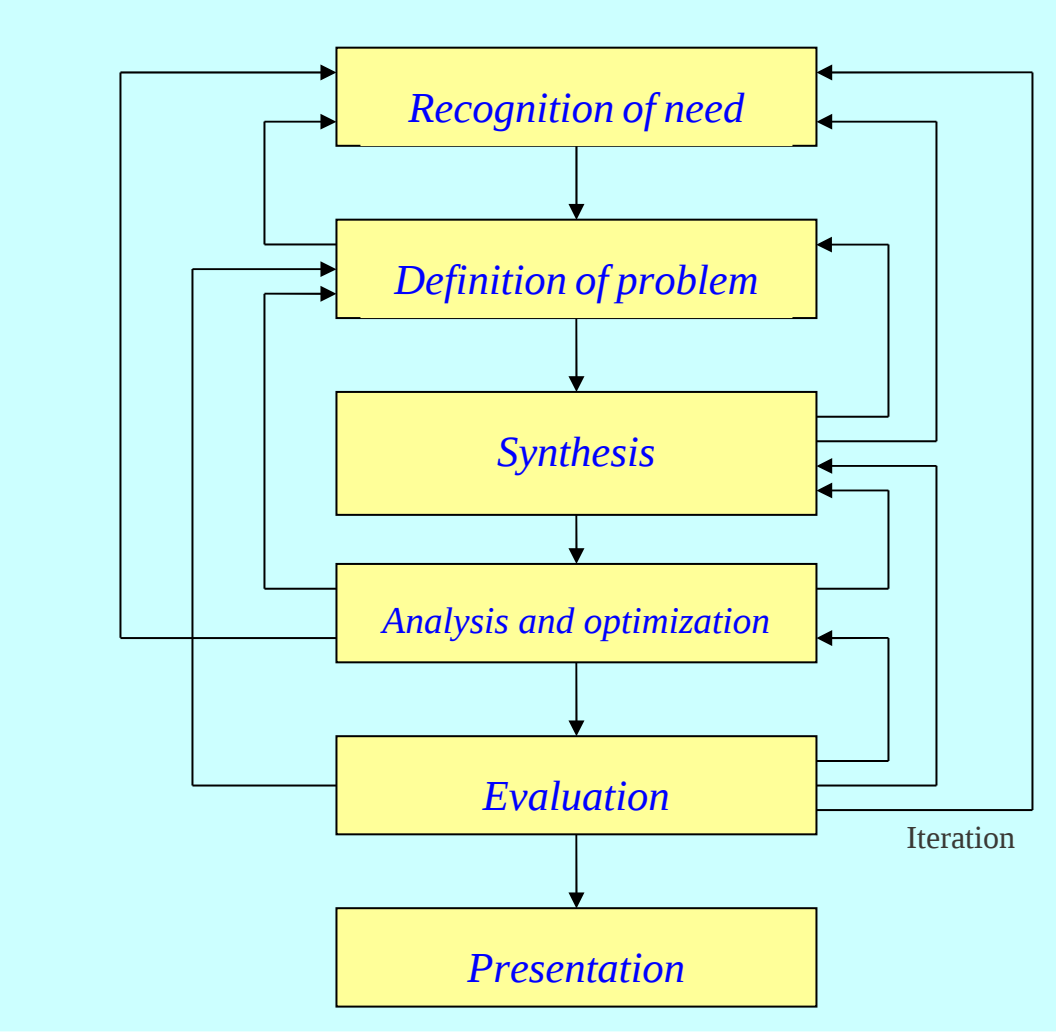


Fig. 2-10 The phases of design [5]

## 2.6 INTRODUCTION TO PROGRAMMING

The computer revolution has had an impact on every facet of today's modern engineering firm, changing everything from the way a company composes its correspondence to the method used to perform the necessary calculations for a particular project. This impact has changed not only the way everyday operations of a company are performed but also the amount and kind of knowledge its employees are expected to possess. For example, twenty-five years ago an entry-level engineer was expected to have a good understanding of the principles of engineering in addition to a working knowledge of manual drafting. Having those manual drafting skills enabled the engineer to communicate readily with the drafting team.

Calculations were carried out with tables and/or in many cases, a slide rule design concepts were sketched out by hand on either Vellum or Mylar and then passed down to a drafter, who completed the drawing. This process was very time consuming and at times resulted in mistakes from miscommunication between the design team and the engineer. In today's market, companies have been forced to downsize their overhead (staff) in order to stay competitive within a changing global market. This has changed the requirements for employment set forth by today's industry. Employees are now expected to have a larger range of skills that encompass an expanding job description. All companies now expect their entry-level engineers to have basic computer skills with basic office applications (word processors, spreadsheets, and databases), in addition to a fundamental understanding of engineering principles. Preference is now given to persons entering this field with computer programming experience, because they offer greater value per capital to the company. Of particular interest to today's employers are persons who can customize the AutoCAD workstation for engineering design applications using AutoLISP, Visual LISP, Visual Basic, C++, Visual C++, ActiveX, or ObjectARX. Engineers who can program are not limited to the conceptualization and reproduction of a project using a computer.

They are also able to create applications (programs) that can alleviate some of the tedious and costly aspects of the design process.

Before learning how to manipulate a computer to perform specialized tasks, it is necessary review a few basic terms and concepts. The components that make up a computer can be divided into two categories: Hardware and Software, as illustrated in Figure 1–11. Hardware is defined as the physical attributes of a computer, for example, the monitor, keyboard, central processing unit (the part of the computer where the logic and arthicmatic operations are performed), and mouse. Software, on the other hand, is defined as the programs or electronic instructions that are necessary to operate a computer. A Computer Program is an application that is designed to carry out a particular task. Examples of computer programs are AutoCAD, Word, Excel, Access, PowerPoint, and Mechanical Desktop. In fact it is the computer programs that have essentially transformed those machines from an intricate maze of resistors, transistors, and wires, into the powerful tools they are today. The way commands are arranged inside a computer program is known as Syntax. The actual commands used to construct a computer program constitute the Programming Language. Most programming languages are designed to look like naturally spoken languages. This allows the programmer to construct an application using normal thought processes and real world problem-solving techniques.

Examples of computer programming languages are C, C+, C++, Pascal, LISP, AutoLISP, Visual LISP, Diesel, Visual Basic, ActiveX, and ObjectARX.

When a programmer completes a program, it is converted from the language in which it was created to a format that the computer can understand. A program called a Compiler carries out this conversion process. The format that the compiler transforms the program into is called Machine Language, which is based on the binary system. This system is comprised of 1s and 0s; 1 means on

and 0 means off. A main program called the Operating System controls all computers.

The operating system is nothing more than a set of programs that manages
stored information, loads and unloads programs (to and from the computer's Memory), reports the results of operations requested by the user, and manages the sequence of actions taken between the computer's hardware and software.

Examples of operating systems are Windows 2000, Windows NT 4.0, Windows 98, Windows 95, OS2, Warp, and DOS. Operating systems can also be placed in one of two categories: GUI (Graphical User Interface) and Non-GUI (Non-Graphical User Interface). A Graphical User Interface operating system allows the user to interact with a computer by selecting icons and pictures that represent programs, commands, data files, and even hardware (see Figure 1–12). A Non- ***G***raphical ***U***ser ***I***nterface operating system relies on the user to enter commands through the keyboard
(See Figure 1–13). [2]

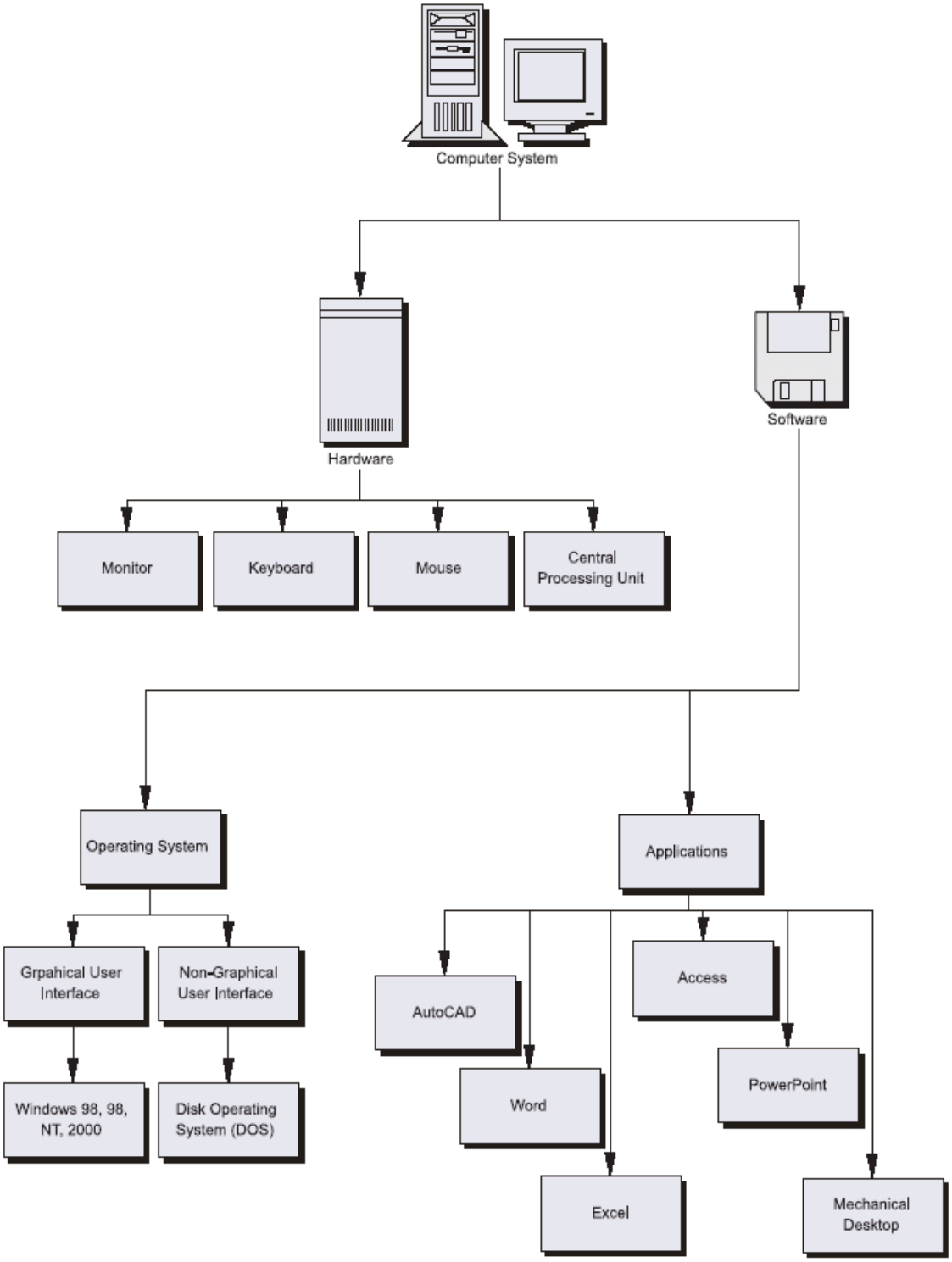


**Figure 1–11** The components of a computer [2]

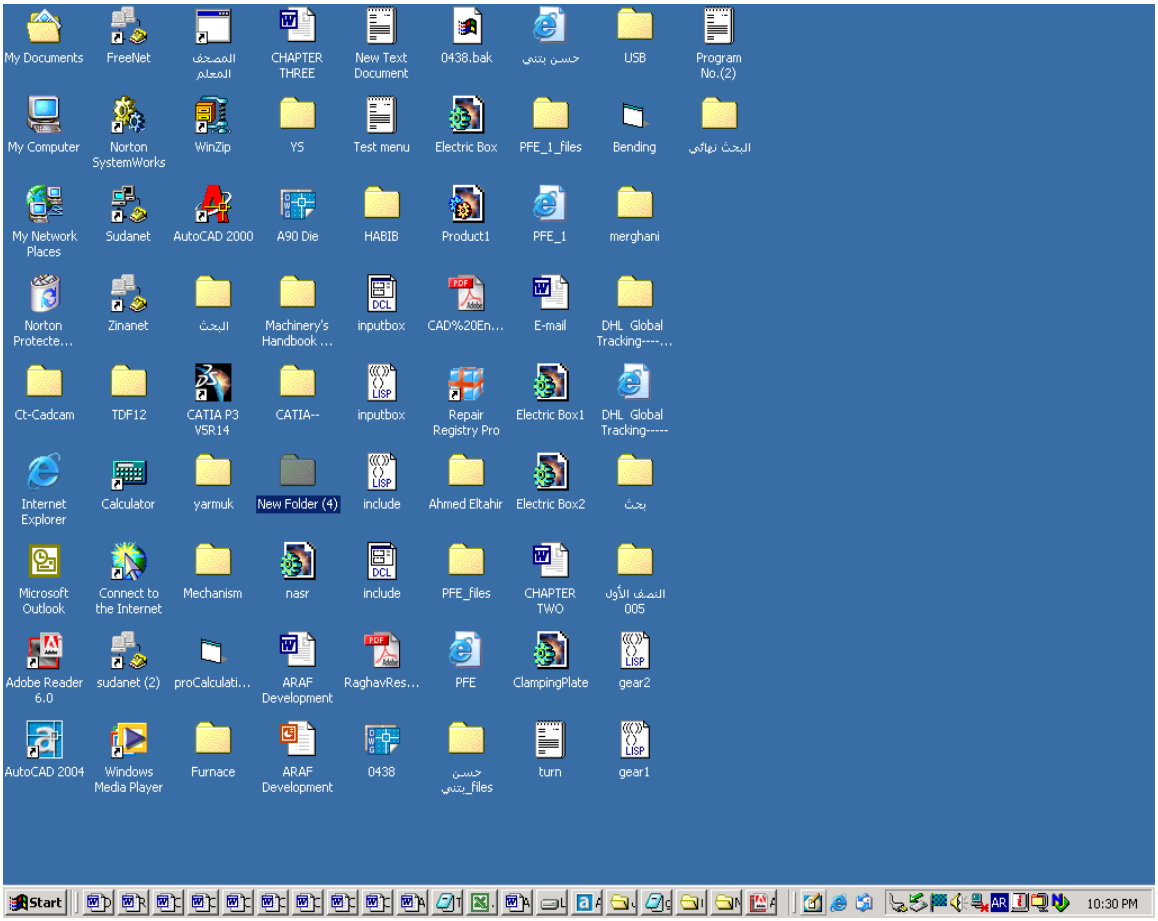


**Figure 1–12** A GUI operating system [2]

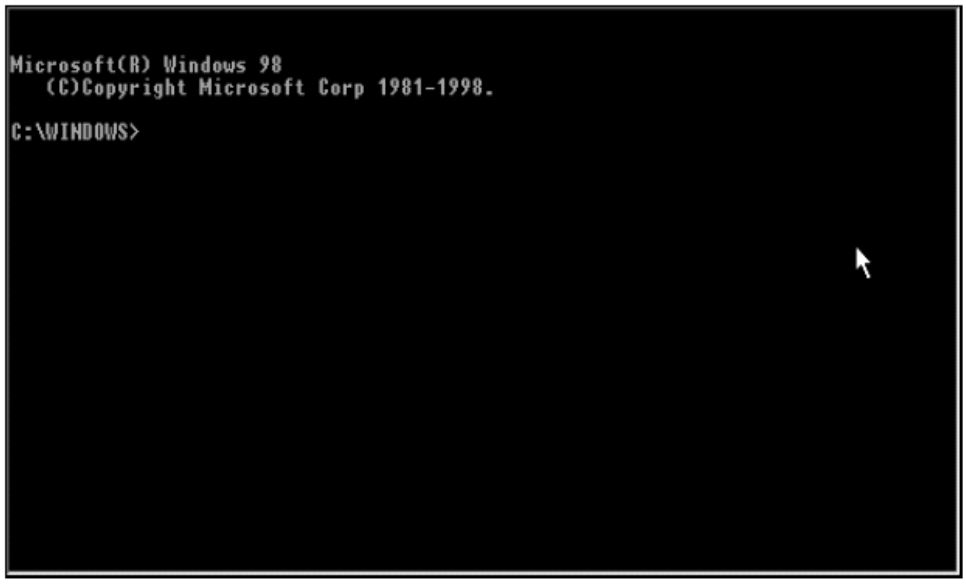


**Figure 1–13** A Non-GUI operating system [2]

## 2.7 HISTORY OF AutoLISP

The history of AutoLISP is closely tied to the history of AutoCAD, so a quick survey of the development of AutoCAD is given first.

### 2.7.1 AutoCAD Releases

AutoCAD was first demonstrated for the public in November 1982 at the COMDEX trade show in Las Vegas. The first shipments of AutoCAD came the next month. Here is a list of the major AutoCAD releases including a few selected features for each release to illustrate the development of AutoCAD. AutoLISP appeared between Release 6 and Release 7. Table No. (2.1) showed the releases and versions of AutoCAD

Table 2-1 AutoCAD releases and versions

| No. | Release or Version | Date of issued | Features |
|---|---|---|---|
| 1 | Release 1 (Version 1.0) | December, 1982 | No dimensioning, menus were limited to 40 items |
| 2 | Release 2 (Version 1.2) | April, 1983 | Price was $1000. The optional dimensioning add-on cost an extra $250 |
| 3 | Release 3 (Version 1.3) | August, 1983 | Added standardized colors, rubber-band cursor, ability to configure, and right-justified text. |
| 4 | Release 4 (Version 1.4) | October, 1983 | New commands UNITS, HATCH, BREAK, FILLET, ARRAY, WBLOCK, and SCRIPT as well as new methods for creating circles and arcs. |
| 5 | Release 5 (Version 2.0) | October, 1984 | Added non-continuous line types, user-named layers, dragging, an isometric grid, and attributes as well as the commands SAVE, VSLIDE, MSLIDE, OSNAP, and MIRROR. |
| 6 | Release 6 (Version 2.1) | May, 1985 | New ability to rotate plot 90 degrees, external programs run from the command prompt (ACAD.PGP file), objects highlighted as |

| | | | |
|---|---|---|---|
| | | | selected, new polyline entity (PLINE and PEDIT), 3D geometry and viewing (ELEV, VPOINT, HIDE), as well as new commands CHAMFER and BLIPMODE. |
| 7 | Version 2.18 | January, 1986 | This was a minor release between Version 2.1 and Version 2.5, but it did contain the first full version of the AutoLISP programming language. |
| 8 | Release 7 (Version 2.5) | June, 1986 | The FILLET command now handles circles and arcs, access to system variables through SETVAR, new commands UNDO, DTEXT, EXPLODE, TRIM, EXTEND, STRETCH, OFFSET, ROTATE, SCALE, DIVIDE, and MEASURE |
| 9 | Release 8 (Version 2.6) | April, 1987 | Added point filters, associative dimensioning, and the command 3DFACE |
| 10 | Release 9 | September, 1987 | Added pull-down menus, dialog boxes, and twenty text fonts |
| 11 | Release 10 | October, 1988 | Multiple "tiled" view ports, user-defined coordinate systems (UCS), perspective view, and six commands for creating surface meshes. |
| 12 | Release 11 | October, 1990 | Paper space and non-tiled paper space view ports, ordinate dimensions, external references (XREF), solid modeling ("Advanced Modeling Extension"), a line editor for editing single lines of text, abbreviations in the ACAD.PGP file, fractional input, cylindrical and spherical coordinate entry, and the SHADE command. |
| 13 | Release 12 | June, 1992 | Immediate access to the graphics screen (no more main menu), dialog boxes for layers, dimensioning, plotting, etc., grips, regions, and |

| | | | |
|---|---|---|---|
| | | | commands BHATCH, and RENDER |
| 14 | Release 13 | February, 1995 | A new ACIS solid modeler, true ellipses and splines, customized toolbars, real time zoom and pan, infinite construction lines and rays, multiple parallel lines, object linking and embedding (AutoCAD drawings can be embedded in other documents, such as a word processing document, or other documents, such as a spread sheet, can be included in an AutoCAD drawing), handling text in paragraphs, associative hatching and boundary hatching, spell checker, TrueType fonts, geometric dimensioning and tolerancing frames. |
| 15 | Release 14 | May, 1997 | No more DOS version. Added autosnap toggle, filter tracking, friendlier object properties tools, viewing drawings on the internet. |
| 16 | Release 2000 | April, 1999 | Ability to open several drawing in an AutoCAD editing session and copy/paste between them. New design center which streamlines borrowing blocks, layers, text styles, etc. from another drawing. QDIM command for quickly creating multiple dimensions. AutoTrackTM enhanced referencing of points relative to osnap points. New convenient procedures for editing object properties. Lines weights. |
| 17 | Release 2002 | June 2001 | Improved associative dimensions, friendlier editing and extracting of block attributes, and improved web-based features for long distance collaboration. [ 7 ] |

### 2.7.2 History Of Lisp

LISP was created in the context of AI (Artificial Intelligence) research. In 1956 John McCarthy developed the foundation for LISP during the Dartmouth Summer Research Project on AI.

During the early 60's, the principal dialect of LISP was Lisp1.5, then many different dialects and implementations for various computers appeared, including BBNLisp, Interlisp, MacLisp, NIL (New Implementation of Lisp), Franz Lisp, Scheme, Flavors, LOOPS (Lisp Object Oriented Programming System), SPICE-Lisp, and PSL (Portable Standard Lisp). The 70's and early 80's even saw the development of specialized computers known as Lisp Machines which were designed specifically to run LISP programs.

In 1981 many LISP programmers got together to identify the common aspects of certain dialects and thus created Common LISP. By 1984 Common Lisp was considered the de facto standard. Golden Common LISP was developed from Common LISP for the IBM PC, and David Betz developed XLISP, which is the dialect of LISP on which AutoLISP is based. [ 7 ]

### 2.7.3 AutoLISP'S Roots in Lisp

AutoLISP is a direct descendant of the XLISP dialect of the LISP programming language. In fact, AutoLISP can be considered both a subset and a superset of XLISP. It is a subset since it does not include all the functions contained in XLISP. It is a superset since it contains many functions not found in XLISP. These added functions allow AutoLISP to interact with AutoCAD's drawing database and commands. However, in concept, syntax, and

programming style, LISP (XLISP in particular) is clearly the progenitor of AutoLISP. [ 7 ]

### 2.7.4 Basic Continuity

AutoLISP has been included in AutoCAD ever since version 2.18 (January, 1986). Although it has grown, it has maintained its continuity over the years. Each time a new release of AutoCAD comes out, can expect to find a few new AutoLISP functions. Here are in table 2.2 some that have appeared in past releases of AutoCAD:

Table 2-2 What have appeared in AutoCAD releases

| No. | Release or Version | Date of issued | Features |
|---|---|---|---|
| 1 | Release 7 (Version 2.5) | June, 1986 | Introduced access to entities in the drawing database |
| 2 | Release 8 (Version 2.6) | April, 1987 | Introduced 3D points and the functions getcorner, getkword, and initget. |
| 3 | Release 9 | September, | Introduced the command function and ssget |

| | | | |
|---|---|---|---|
| | | 1987 | filters. |
| 4 | Release 10 | October, 1988 | Introduced enhanced functions findfile and load; new functions getenv, handent, and vports. |
| 5 | Release 11 | October, 1990 | Introduced enhanced function entget; new functions cvunits and entmake. |
| 6 | Release 12 | June, 1992 | Introduced enhanced functions ssget and initget; new functions alert, getfiled, and textbox, and programmable dialog boxes using the DCL (Dialog Control Language). |
| 7 | Release 13 | February, 1995 | Introduced several new functions including acad_strlsort, autoload, help, and acad_colordlg. |
| 8 | Release 14 | May, 1997 | Introduced Visual LISP™, a complete AutoLISP programming environment. |
| 9 | Release 2000 | April, 1999 | Visual LISP™ now built in. |

Naturally, these new functions and enhanced functions are welcomed additions. They represent growth and seldom make the old functions obsolete. For the most part, the old functions work the same in the latest release as in previous releases. In fact, when compared with operating systems, the format of the drawing database, and the user interface, AutoLISP is one of the more consistent aspects of AutoCAD. [ 7 ]

**2.8 INTRODUCTION TO AUTOLISP**

AutoLISP is a derivative, or dialect, of the LISP programming language. LISP (List Processing) is a high-level computer programming language used in artificial intelligence (AI) systems. In this reference, the term high-level does not mean complex, rather it means powerful. As a matter of fact, many AutoCAD users refer to AutoLISP as the "non-programmer's language" because it is easy to understand.

The AutoLISP dialect is specially designed by Autodesk to work with AutoCAD. It has special graphic features designed to work in the drawing

editor. It is a flexible language that allows the programmer to create custom commands and functions that can greatly increase productivity and drawing efficiency.

AutoLISP can be' used in several ways. It is a built-in feature of AutoCAD and is therefore available at the command: prompt. When AutoLISP commands and functions are issued inside parentheses, the AutoLISP interpreter automatically evaluates the entry and carries out the specified tasks. AutoLISP functions can be incorporated into the AutoCAD menu as toolbar buttons, screen menu items, and tablet menu picks. AutoLISP command and function definitions can be saved in an AutoLISP program file and then loaded into AutoCAD when needed. Items that are used frequently can be placed in the acad.lsp file, which is automatically loaded when AutoCAD starts.

The benefits of using AutoLISP are endless. Third party applications (add-on software that. enhances AutoCAD) use AutoLISP to perform specialized functions, such as creating special symbols.

A person with a basic understanding of AutoLISP can create new commands and functions to automate many routine tasks. He will be able to add greater capabilities to his screen, tablet, and toolbar menu macros. He can also enter simple AutoLISP expressions at the Command: prompt. More experienced programmers can create powerful programs that quickly complete very complex design requirements. Several powerful AutoLISP programs are found in the

AutoCAD Release 14 Bonus Utilities. Other new functions include the following:

- Automatic line breaks when inserting schematic symbols.
- Automatic creation of shapes with associated text objects.
- Parametric design applications that create geometry based on numeric entry.

Knowing basic AutoLISP gives a better understanding of how AutoCAD works. By learning just a few simple functions, one can create new commands that make a significant difference in your daily productivity levels. [3]

## 2.9 TYPES OF PROGRAMMING LANGUAGES

Computers can be programmed using various programming languages. The subject of computer programming stems back to the historic 1946 treatise of John von Neumann (et al) which described in detail the concept of storing programmed instructions to control the operations of a digital computer. From this fundamental notion, a wide variety of computer programming languages evolved.

During the 50's, 60's, and 70's various computer languages developed for different purposes. These languages can be grouped in several categories including:

Algorithmic/procedural languages such as Fortran, Algol, Cobol, and PL/1

Time-sharing languages such as BASIC

Process-control languages such as APT

String processing languages such as Snobol, and List processing languages such as LISP ("LISP" stands for list processing)[ 7 ]

## 2.10 A FEW SAMPLE DIFFERENCES

Despite the basic continuity of AutoLISP, there are some bothersome changes that appear from time to time. Here are some that came along with AutoCAD R13, R14, and AutoCAD 2000.

1. Ellipses -- In R13 ellipses have a true elliptical shape, whereas in R12 ellipses were made up of polylines having a series of circular arc segments. The new ellipses are stored in the drawing database differently than the old ones, and any AutoLISP programs which manipulated ellipses had to be revised for R13.
2. Solid Models -- In R13 all the solid modeling commands and entities changed from the previous AME (Advanced Modeling Extension) to the new ACIS modeler. All AutoLISP programs written for the previous set of solid modeling commands and entities were instantly outmoded.
3. Lightweight Polylines -- In R14 the PLINE command creates an entity called an LWPOLYLINE rather than a standard POLYLINE. (The "LW" stands for light-weight.) The standard POLYLINE entity still exists and is created by the 3DPOLY command or by splining or fitting an LWPOLYLINE. This new LWPOLYLINE is stored more efficiently in memory than the regular polyline. (All vertices are stored in one top-level entry, rather than each vertex being stored in a separate sub-entity.) AutoLISP programs which were written for AutoCAD R13 and earlier, and which extract database information on polylines or attempt to modify polylines, have to be reprogrammed so that they detect this new type of entity and work with its new method of storage.
4. ADS -- In R14 ADS is obsolete. The AutoCAD Development System (ADS) was the C-language extension of AutoLISP. External programs were written in C, compiled for a given operating platform, then called as part of an AutoLISP program. ADS is replaced by ARX (AutoCAD Runtime Extension).
5. AutoLISP Interpreter -- In AutoCAD 2000 the AutoLISP interpreter is new, which results in a number of significant changes, especially for advanced programming. (As just one example, in earlier versions of

AutoLISP, functions were stored internally as lists. Advanced programs which read or manipulate program code elements, as opposed to merely reading and manipulating data, need to use one of several new function-defining functions to continue this practice in AutoCAD 2000.) Also, Visual LispTM in AutoCAD 2000 has a number of significant differences from the Visual Lisp that was available for use in Release 14. There are a number of concerns for the beginning and intermediate programmer as well, a few of which are cited in the following items.

6. New Default Settings for Osnap -- In contrast to earlier releases of AutoCAD, which set all osnap modes "off" by default, AutoCAD 2000 sets certain osnap modes "on" by default. A good program should always save the user's current settings, set osnaps as needed throughout the program, then restore the user's settings when the program is finished, even in earlier versions of AutoCAD. However, those programs, which "got by" without doing this may produce bogus graphic results or may even crash when run in AutoCAD 2000.
7. Printed Messages Delayed -- In versions of AutoCAD before 2000, the output from the various print functions was displayed in the command prompt area immediately. However, in AutoCAD 2000 the messages are held in a buffer until a return character is encountered, such as "\n" or "\r". Depending on how a message or prompt has been programmed, this can create the following situation: an AutoLISP program is actually waiting for input but the message or prompt has not yet been displayed and AutoCAD is still displaying its command prompt. This would be confusing to any AutoCAD user, and requires reprogramming.
8. Upper and Lower Case -- In AutoCAD 2000 symbol table names (such as layer names, text style names, block names, etc.) are no longer just upper

case. If an application makes comparisons of such names, and assumes that they will all be upper case, the application will need to be revised.

9. Revised Error Messages -- The wording of many error messages is different in AutoCAD 2000. Any user defined error routines, which make use of the error string passed to it, will have to be revised to accommodate these new wordings.
10. Better Associative Dimensions -- Associative dimensions are now more firmly associated with part geometry, and a new system variable, DIMASSOC, replaces the old DIMASO. Programs which used to set DIMASO to 1, to make sure that associative dimensioning was turned on, should now set DIMASSOC to 2. [ 7 ]

CHAPTER THREE

# DEVELOPMENT OF CONCEPT

## 3.1 CUSTOMIZATION CONCEPT

In this chapter the development of the customization concept will be explained through the following steps:

### 3.1.1 Creating Menu's Customize

Now can write or, obtain, as many AutoLISP routines as you like, but it's too difficult to type every time something like:

(Load "DDStruc_Steel_Ver2")

To type this in, or even remember the name every time you need to load or, run the routine, is just basically headache and un-productive.

Always AutoCAD load all our routines programs from the AcadDoc.Lsp file. This though, would mean that they are all in memory, chewing up your system resources.

What I need to be able to do it, to load/run the routines from the AutoCAD menu. Not so long ago, in earlier releases, the only option had been to modify the standard AutoCAD menu. Not anymore. Now can create a custom menu known as a "Partial Menu" and install this to run side by side with the standard AutoCAD menu.

Here will take go through all the steps of developing a fully, functional Custom Standard Menu.

And will only be covering Pull Down Menu's. If you need information on Screen, Button or Tablet Menu's, then please refer to the AutoCAD Customization Manual.

To get started begin with designing our Pull Down or Pop Menu.

Fire up the text editor and type in the following saving the file as TDF.Mns:

(Please ensure that you save all files to a directory in the AutoCAD Search Path.)

```
***MENUGROUP=TDF, NDA
***POP1 //pull down name
P1-1 [TDF] //pull down label
P1-2 [Lower Block] //menu items
P1-3 [Guide]
```

P1-4 [Ballon]

P1-5 [Bolt]

P1-6 [Gear]

The first line:

***MENUGROUP=TDF is the name of the Partial Menu.

The second line:

***POP1 is the name of the Drop Down or POP menu.

The third line consists of 2 parts:

P1-1 is the "Name Tag" or "ID" of the menu item and allows you to access the menu item programmatically.

The second part:

[TDF-Menu] is the menu label that appears in the Pull Down Menu.

The first label in a pull down menu always defines the menu bar title whilst, the succeeding labels define menu and sub-menu items.

Now need to load the menu.

A new menu item will appear on the menu bar entitled "TDF".

It should look like Fig 3.1 below:

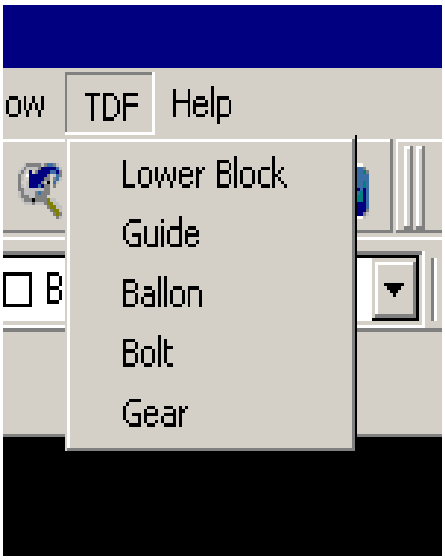

Fig 3.1 Menu Item

O.K. Now they got the menu to display but there's a problem. It doesn't do anything!!

Let's make it functional.

Create a new menu file entitled TDF.MNS and type in the following:

```
//      AutoCAD menu file - C:\TDF\TDF-NASR.MNS
  ***MENUGROUP=TDF,NDA
  ***POP1
            [TDF]
            [BY MR.NASR]
            [EDUCATION ONLY]
               [--]
            [lower_Die_Block_2D]^C^C(if (null C:lower_Die_Block_2D) (load
  "lower_Die_Block_2D"));lower_Die_Block_2D;
               [--]
            [lower_Die_Block_2DC]^C^C(if (null C:lower_Die_Block_2DC)
  (load "lower_Die_Block_2DC"));lower_Die_Block_2DC;
               [--]
            [lower_Die_Block_3D]^C^C(if (null C:lower_Die_Block_3D) (load
  "lower_Die_Block_3D"));lower_Die_Block_3D;
               [--]
            [Guide_Pin]^C^C(if (null C:Guide_Pin) (load
  "Guide_Pin"));Guide_Pin;
               [--]
            [&Assy_Die_Blocks]^C^C(if (null C:Assy_Die_Blocks) (load
  "Assy_Die_Blocks"));Assy_Die_Blocks;
              [--]
            [&balloon]^C^C(if (null C:balloon) (load "balloon"));balloon;
               [--]
            [&Gear]^C^C(if (null C:Gear) (load "Gear"));Gear;
               [--]

  ***TOOLBARS
```

**TDF
ID_TbTDF [_Toolbar("TDF", _Floating, _Show, 445, 154, 1)]
ID_lower_Die_Block_2D [_Button("lower_Die_Block_2D", "ICON8467.bmp",
"ICON_16_LINE")]^C^C(if(nullC:lower_Die_Block_2D)(load
"lower_Die_Block_2D"));lower_Die_Block_2D;
ID_lower_Die_Block_2DC[_Button("lower_Die_Block_2DC","ICON6334.bmp",
"ICON_16_lower_Die_Block_2DC")]^C^C(if (null C:lower_Die_Block_2DC)
(load "lower_Die_Block_2DC"));lower_Die_Block_2DC;
ID_lower_Die_Block_3D[_Button("lower_Die_Block_3D",
"ICON_lower_Die_Block_3D.bmp", "ICON_16_lower_Die_Block_3D")]^C^C(if
(null C:lower_Die_Block_3D) (load
"lower_Die_Block_3D"));lower_Die_Block_3D;
ID_Guide_Pin[_Button("Guide_Pin","ICON9169.bmp","ICON_16_Guide_Pin")]^
C^C(if (null C:Guide_Pin) (load "Guide_Pin"));Guide_Pin;
ID_Assy_Die_Blocks [_Button("Assy_Die_Blocks",
"ICON_Assy_Die_Blocks.bmp", "ICON_16_Assy_Die_Blocks")]^C^C(if (null
C:Assy_Die_Blocks) (load "Assy_Die_Blocks"));Assy_Die_Blocks;
ID_balloon [_Button("balloon", "ICON1478.bmp",
"ICON_16_balloon")]^C^C(if (null C:balloon) (load "balloon"));balloon;
ID_Gear [_Button("Gear", "ICON.bmp", "ICON_16_Gear")]^C^C(if (null
C:Gear) (load "Gear"));Gear;

***HELPSTRINGS
ID_BALLOON [Creates balloon segments: Balloon]
ID_GEAR [Creates gear segments: Gear]
ID_LOWER_DIE_BLOCK_3D [Creates lower_Die_Block_3Dsegments:
lower_Die_Block_3D]
ID_LOWER_DIE_BLOCK_2D [Creates lower_Die_Block_2D segments:
lower_Die_Block_2D]
ID_LOWER_DIE_BLOCK_2DC [Creates lower_Die_Block_2DC segments:
lower_Die_Block_2DC]
ID_GUIDE_PIN [Creates Guide_Pin segments: Guide_Pin]

ID_ASSY_DIE_BLOCKS [Creates Assy_Die_Blocks segments:
Assy_Die_Blocks]

// End of AutoCAD menu file - C:\TDF\TDF.MNS

Load this menu, following the same routine as you did for the first menu.

"TDF Menu1" will appear in the menu bar and the new pull down menu should look like Fig 3.1 below:

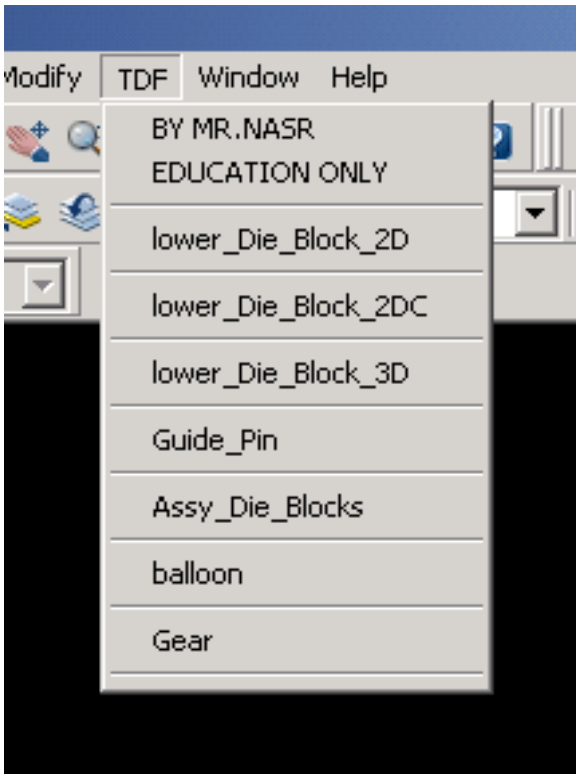


Fig 3.2 TDF Menu

:Let's have a close look at one of the menu items

balloon]^C^C balloon&]

.The first part, P1-1 is, of course, the menu item ID

The second part [&balloon] is the menu label. But did notice something ?'different? What is the '&' character doing in front of the 'B

If you precede any letter in the menu item label with '&', this will define the .letter as the shortcut key to this menu item

In other words, when press 'ALT B' the Line Menu Item will be chosen and any action affiliated to it will be triggered.

Following the item label, each menu item can consist of a command, parameter, or a sequence of commands and parameters. Let's look at the Line menu item macro.

C^C Balloon^

Just in case have a previous incomplete command, use the string ^C^C to start our menu macro. This is exactly the same as pressing CTRL+C or ESC, twice on the keyboard. Use ^C twice because some AutoCAD commands need to be cancelled twice before they return to the Command prompt. (e.g. The Dim Command.)

Can immediately follow this sequence with AutoCAD command, Line.

When a menu item is selected, AutoCAD places a blank after it. A blank in a menu macro is interpreted as ENTER or SPACEBAR. In effect, this is exactly the same as typing, at the command prompt " Balloon " followed by ENTER.

### 3.1.2 Creating Toolbars Customize

In addition to positioning and sizing toolbars, we can customize the toolbar interface. We can add new buttons in new location for quick access. Infrequently used tools can be deleted or moved to an “out-of-the-way” location. Entirely toolbars can be created and filled with redefined buttons, or custom button definitions can be created. Toolbars are customized using Customize dialog box.

To create a new toolbar first prepare program for toolbar inside the TDF.MNS file as mentioned above then:

A. Select Tools from the AutoCAD menu file and from tools menu select the customize dialog box.

B. Select toolbars from the customize dialog box and then select the new option the new toolbar dialog box appear in the dialog box write the toolbar name TDF and then save toolbar in menu group TDF, NDA as show in the Fig. (3.3)

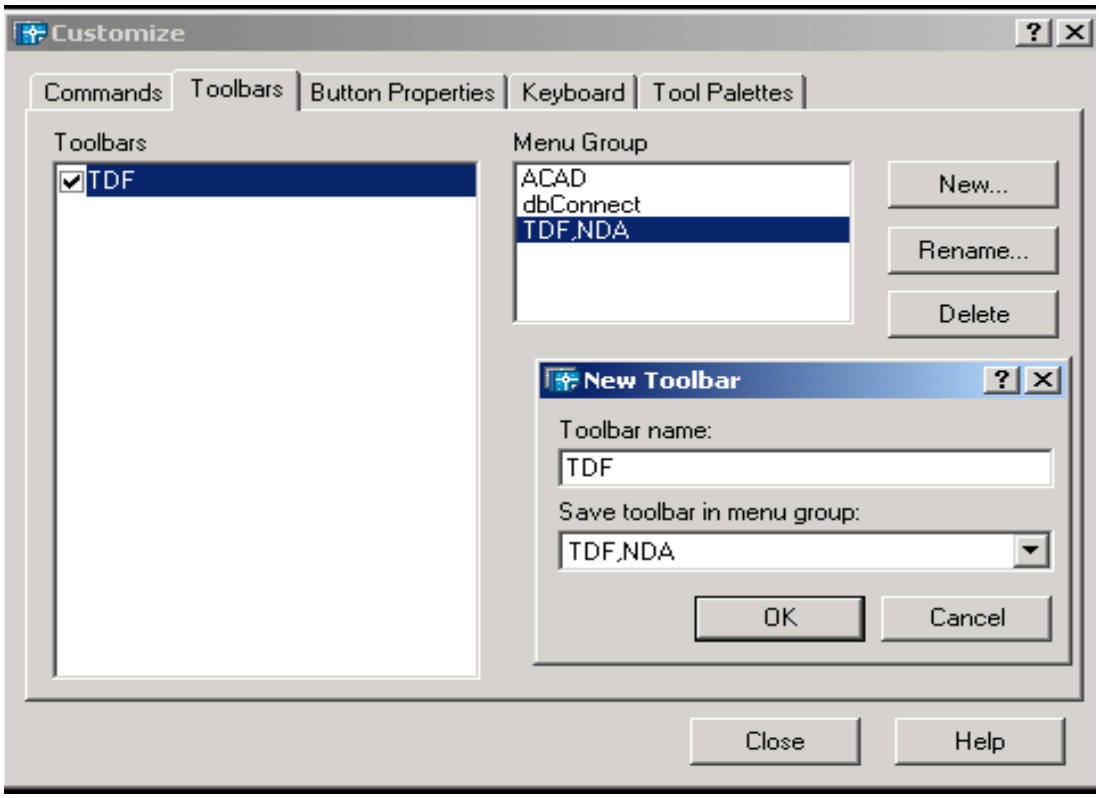


Fig. (3.3) The customize dialog box with tool bar clicked

C. Select properties from customize dialog box as shown in Fig. (3.4) and then duple click on the button image we need to change there properties let us select the Assy die blocks button image and this shown in Fig. (3.5) and then select the edit from Fig. (3.5) then the button edit dialog box shown in Fig. (3.6) in this dialog box we can open an image from any file inside my computer, draw new image, edit any image and save it in toolbar path.

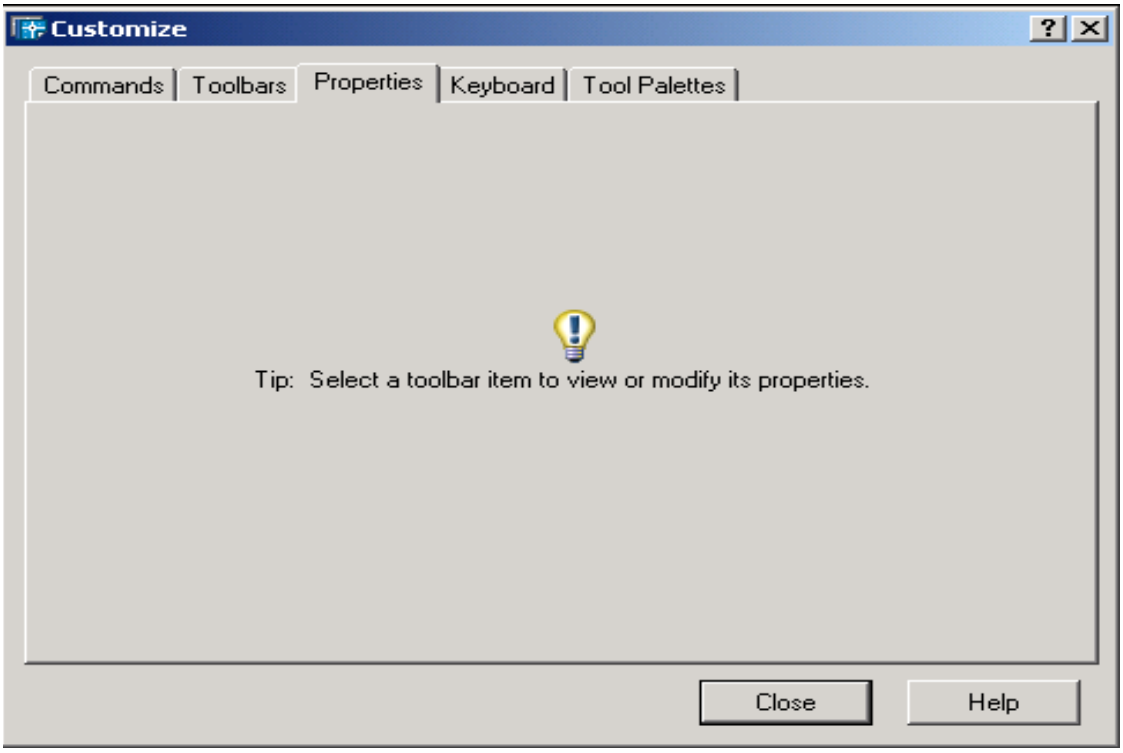


Fig. (3.4) The customize dialog box with properties clicked

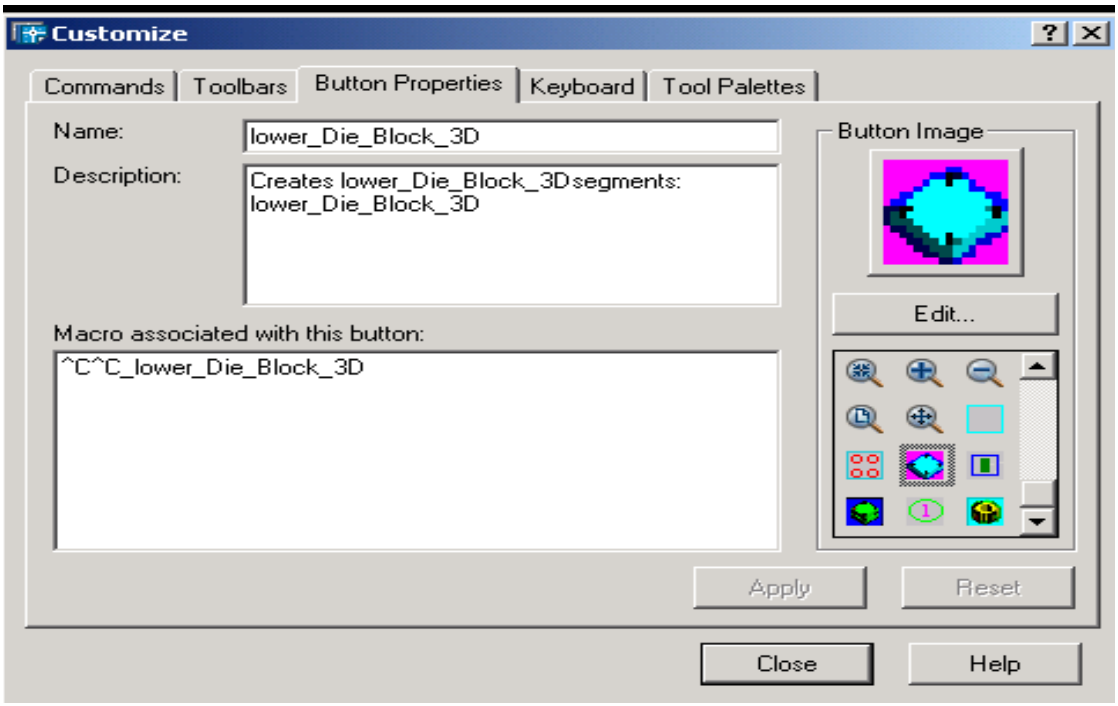


Fig. (3.5) The customized dialog box with the Assembly die block bottom selected

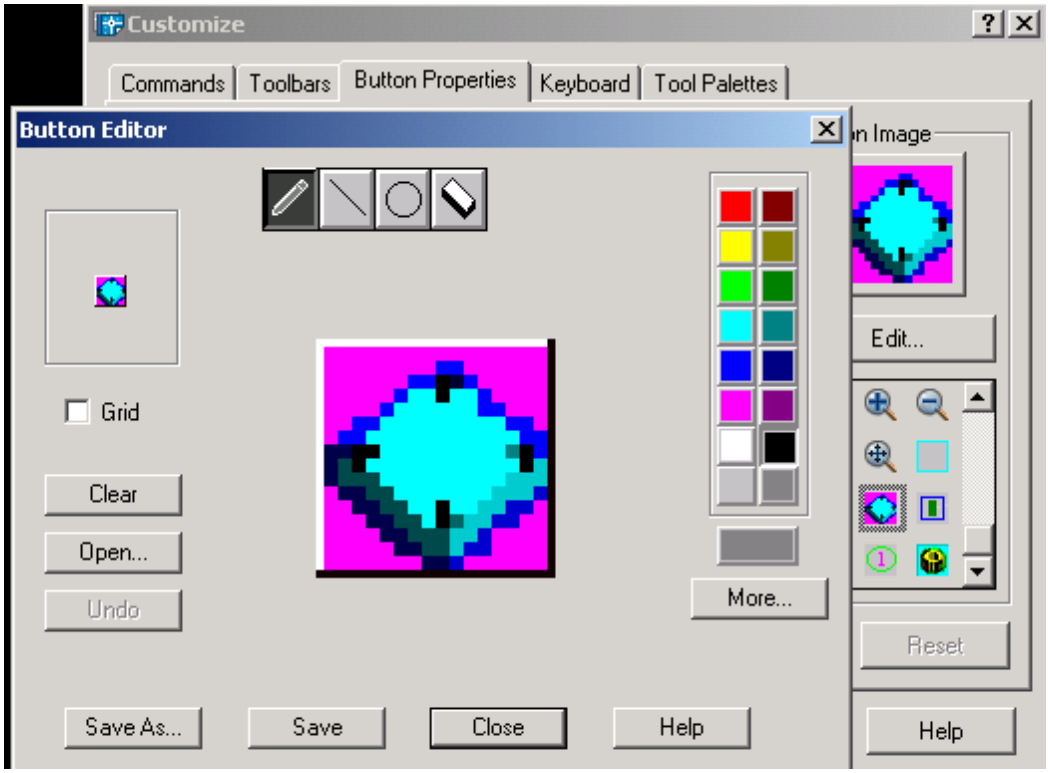


Fig. (3.6) T button edit dialog box

D. Select the keyboard from the customize dialog box and then see the commands of TDF toolbar as shown in Fig. (3.7).

### 3.1.3 Toolbars And Menu Files:

As toolbars are adjusted and customized, the changes are stored in files. AutoCAD maintains the record of these changes in on of several associated with menu system. Each of these files has the same file name, but different file extensions. The customize menu created is name TDF, and the files associated with this menu are located in the TDF folder.

The menu files associated with TDF menu are as follows:

- TDF.MNS. This is the menu source file. All code changes to the toolbar are recorded into this file. This is the file that is used in the creation of a compiled menu file.
- TDF.MNC. This is a compiled menu file. AutoCAD creates this optimized file to handle the menus in our drawing sessions.

- TDF.MNR. This is a resource menu file. It stores all of bitmap images associated with the menu for quick access.

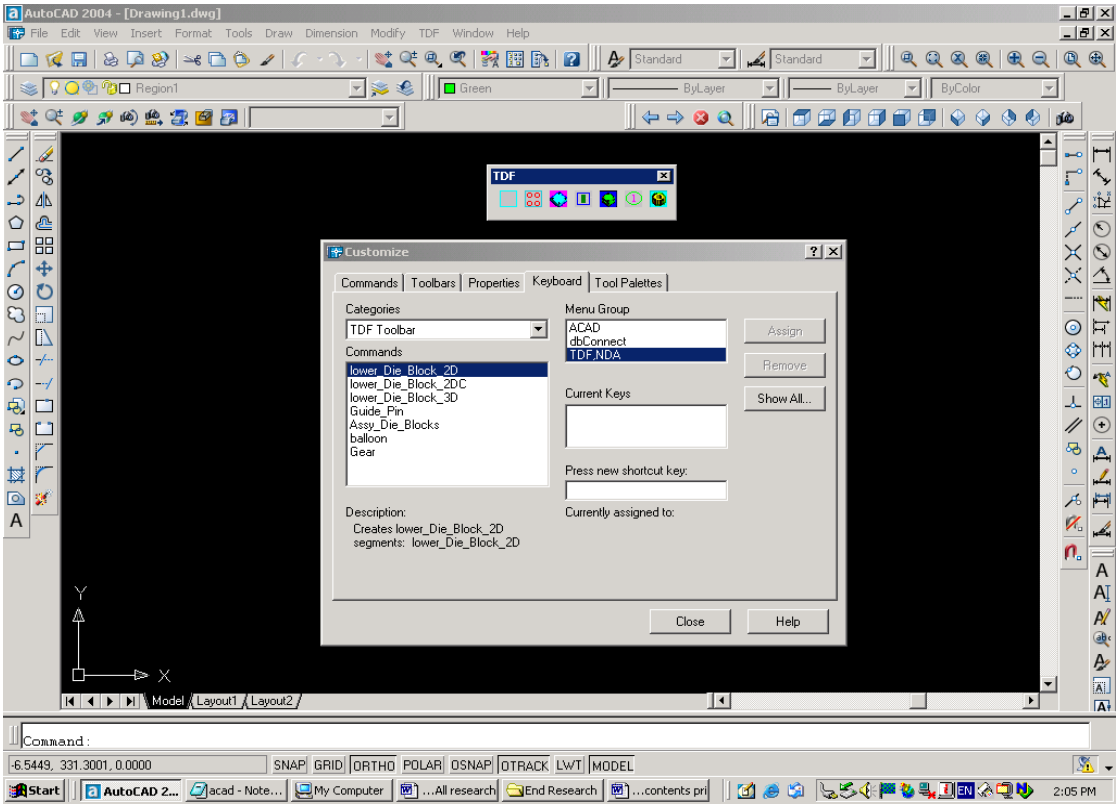


Fig. (3.7) The TDF tool bar commands

At last the TDF toolbar appear on the screen as shown in Fig. (3.8)

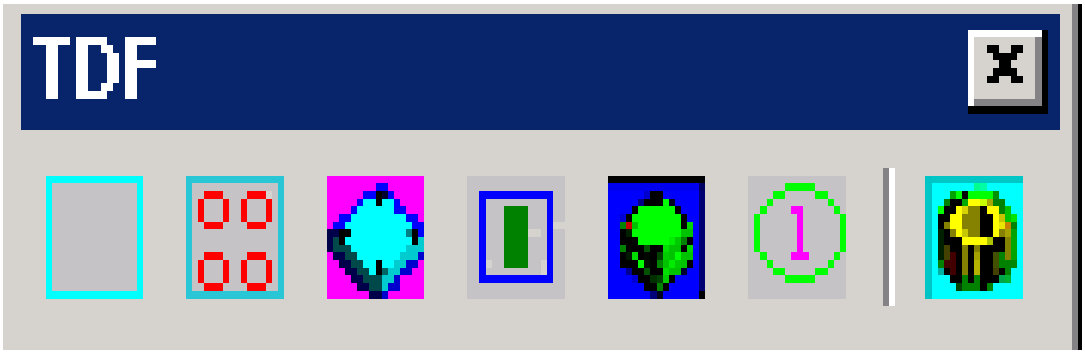


Fig. (3.8) The TDF tool bar

### 3.1.4 Loading Partial Menu's

Fire up AutoCAD and follow these instructions:

1. Select "Tools" in AutoCAD or “Assist” from mechanical desktop.
2. Select options. See Fig. (3.9)

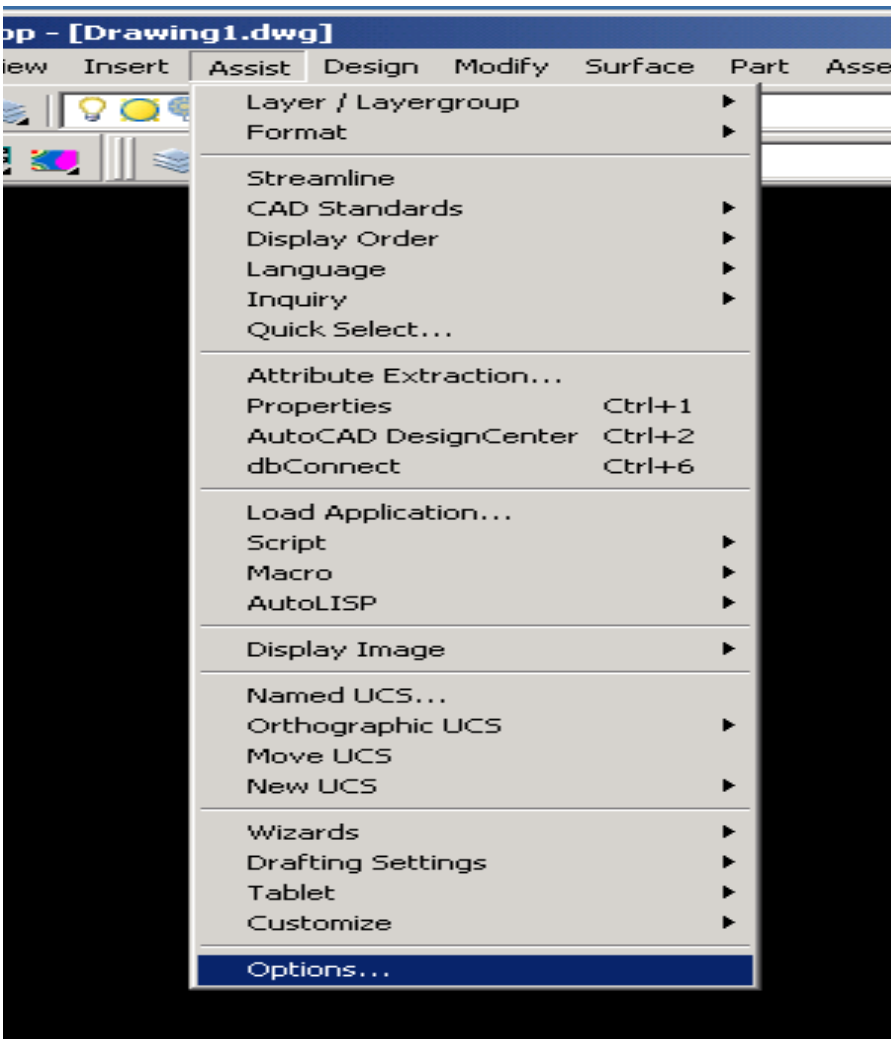


Fig. (3.9) Tool Menu or Assist Menu

3. From options dialog box Open file and from it open support files search path as appear in Fig. (3.10).

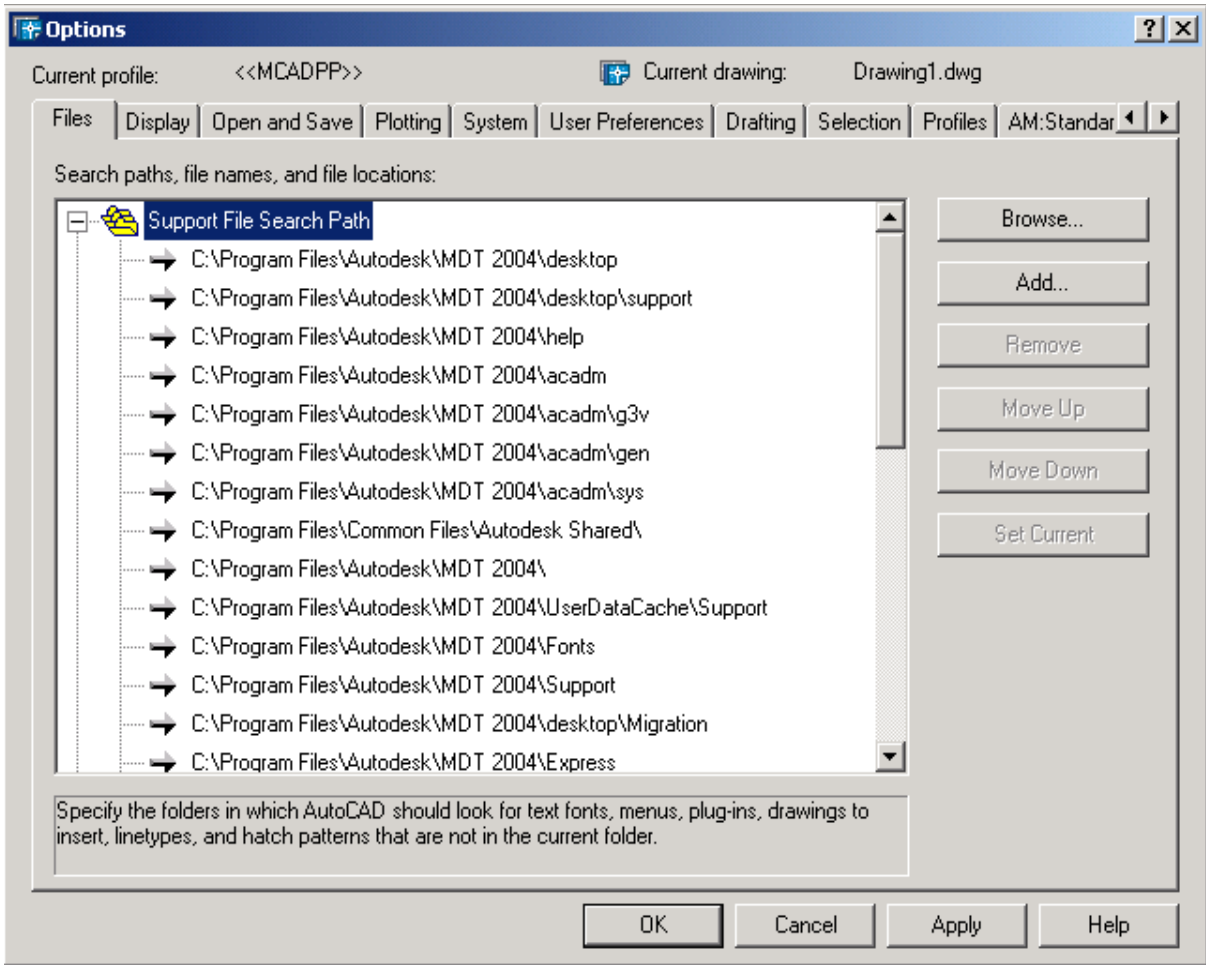


Fig. (3.10) Options dialog box

4. Select add the empty box appear this empty box fill with TDF folder see Fig. (3.11).

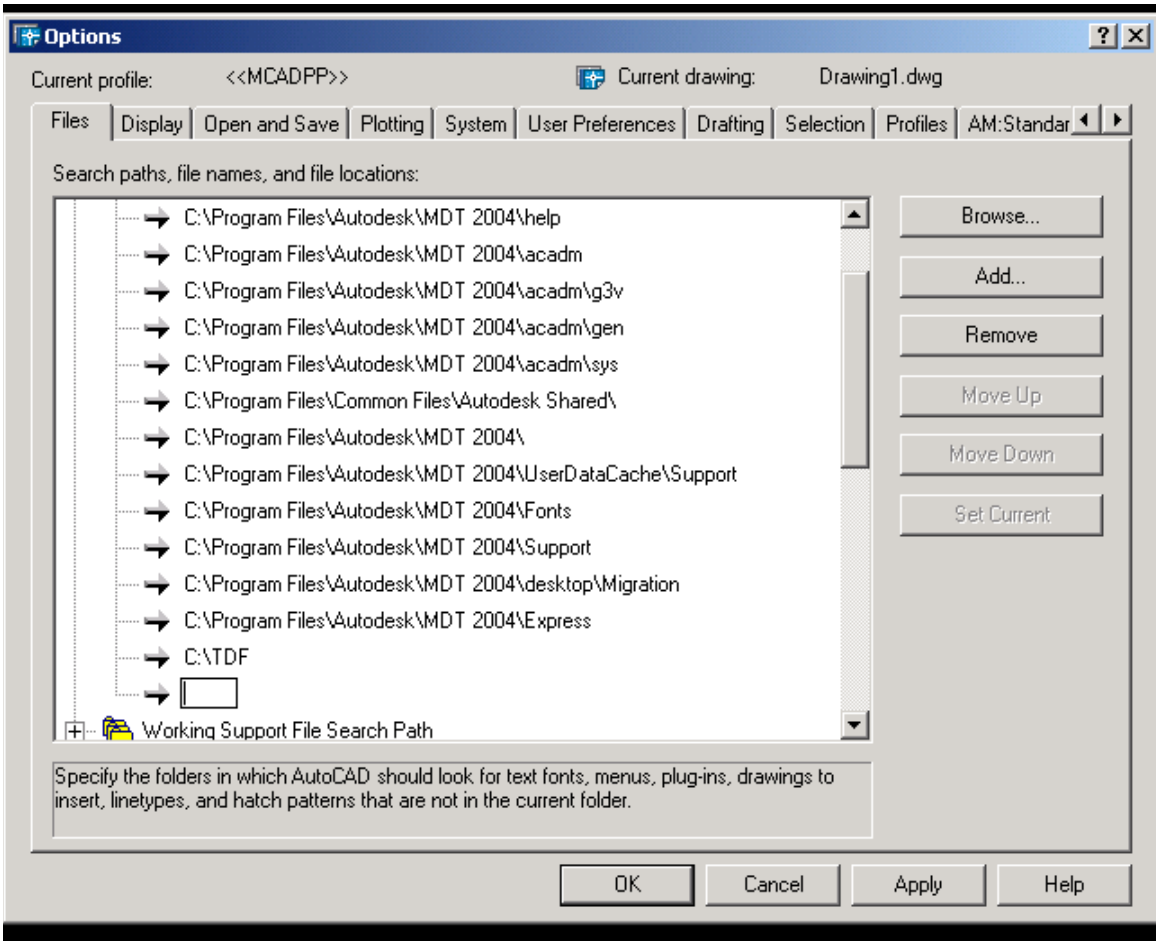


Fig. (3.11) The TDF folder

5. Select "Browse" the browse for folder dialog box appear and select from it the TDF folder and then press ok to add it in the support search file bath and press apply on options dialog box and press ok see Fig. (3.12).

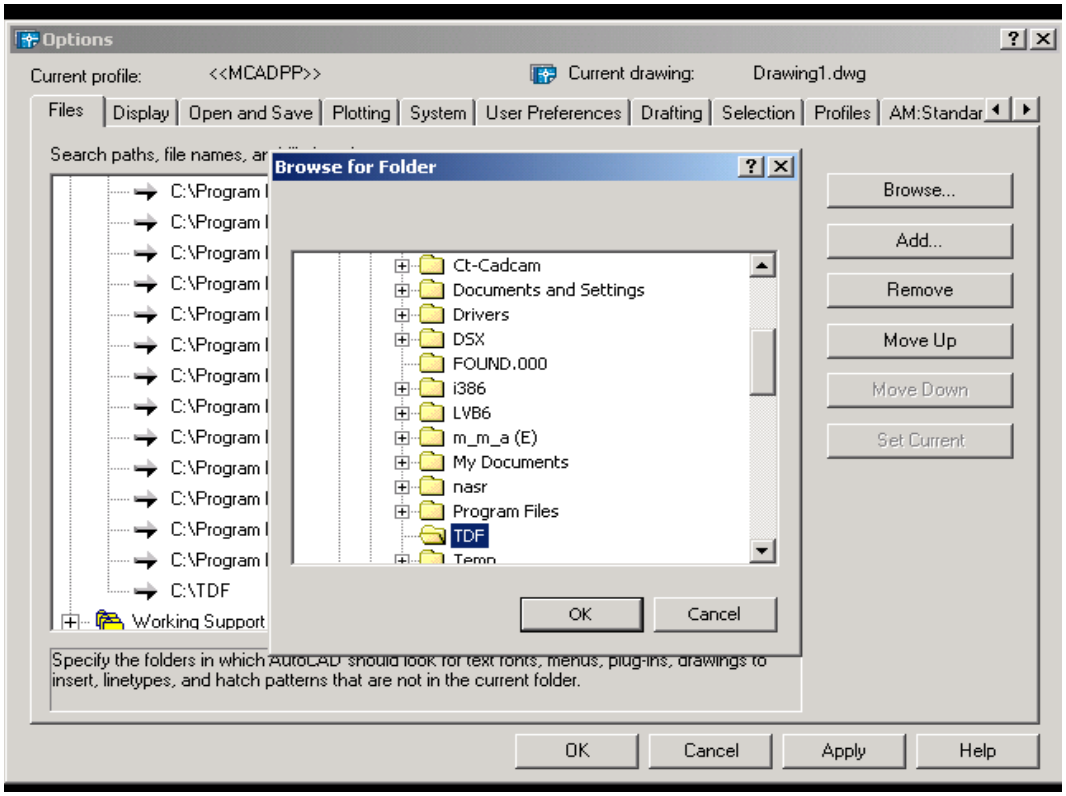


Fig. (3.12) Browse dialog box

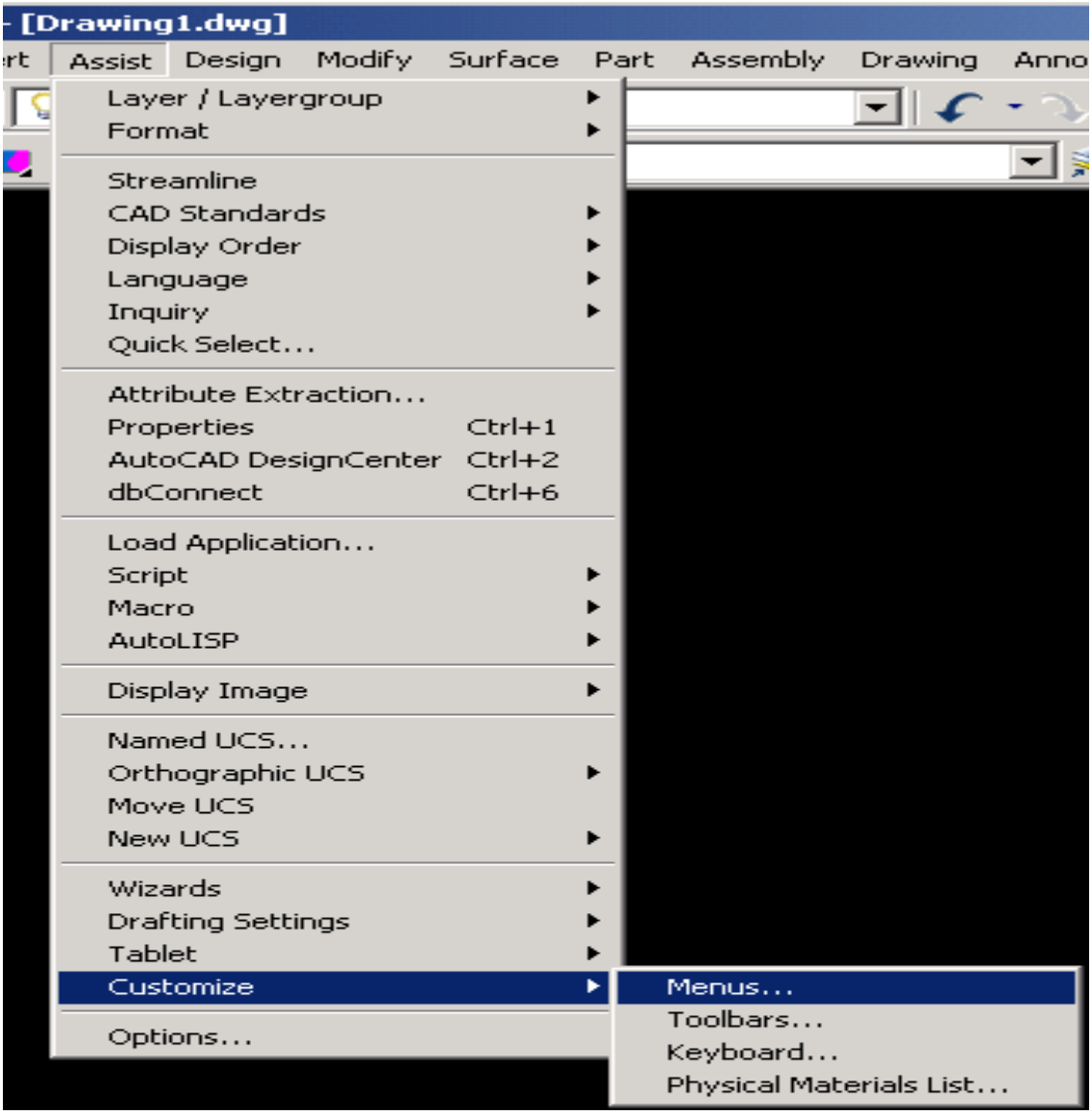


Fig. (3.13) customize menu

6. Select "Tools" "Customize and then select Menu's" the customize menu appear in Fig. (3.13)
7. From menu customization dialog box select browse and then select from browse for folder the TDF folder and open it and open TDF.NNS file and load it see in Fig. (3.14)

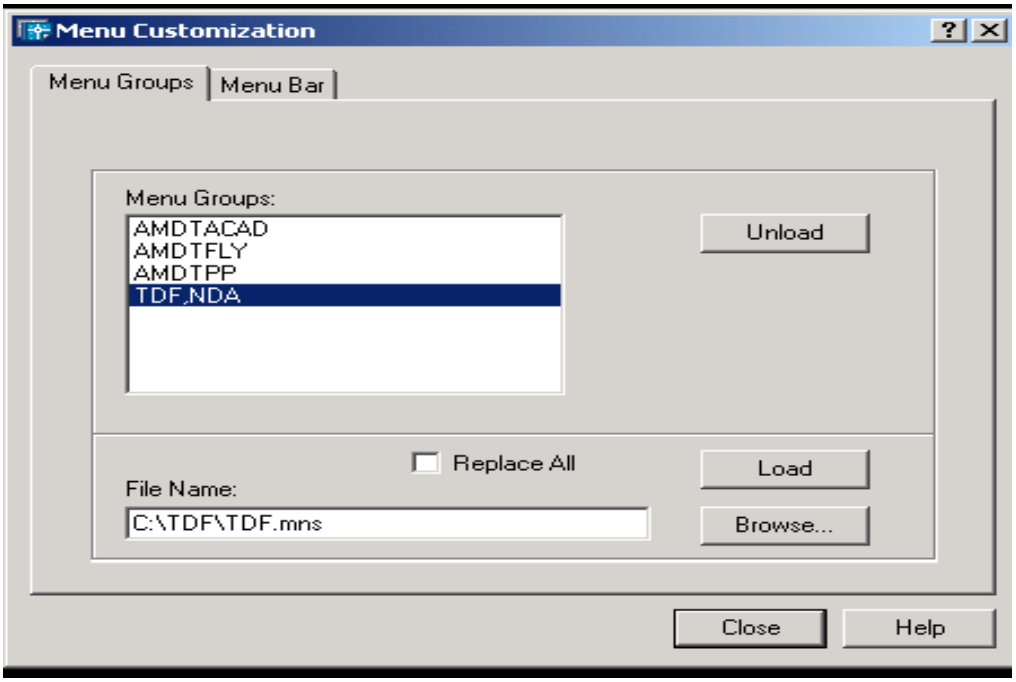


Fig. (3.14) menu customization

8. Select "Menu Bar" form customization dialog box and from menu group select the TDF,NDA and then appear menus TDF see Fig. (3.15)

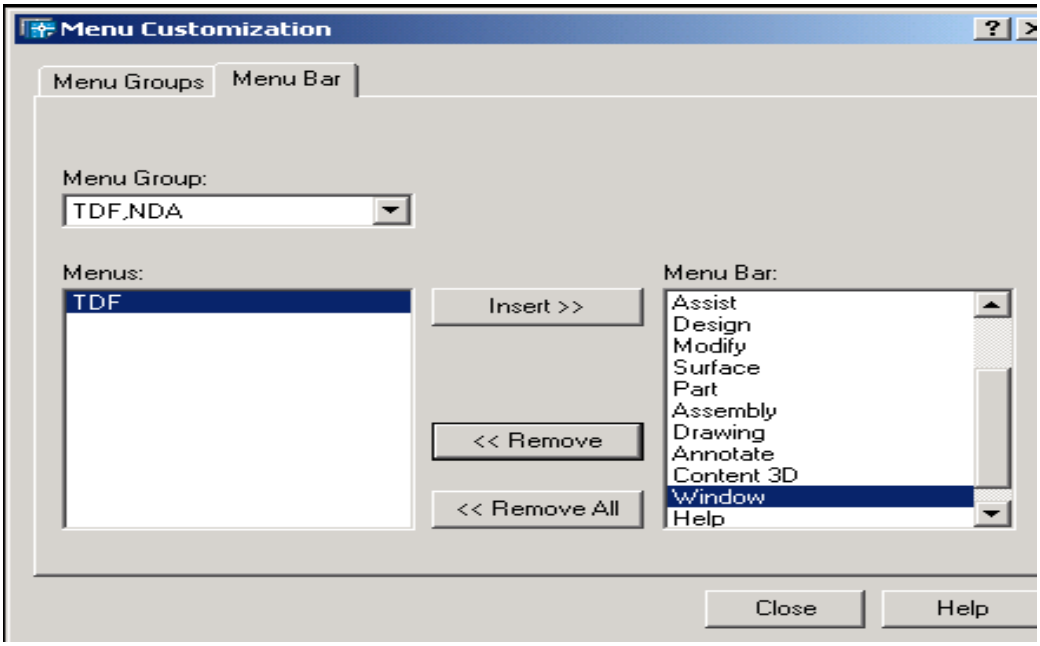


Fig. (3.15) menu bar

9. Press "Insert" to add the TDF to the Menu Bar see Fig. (3.16).

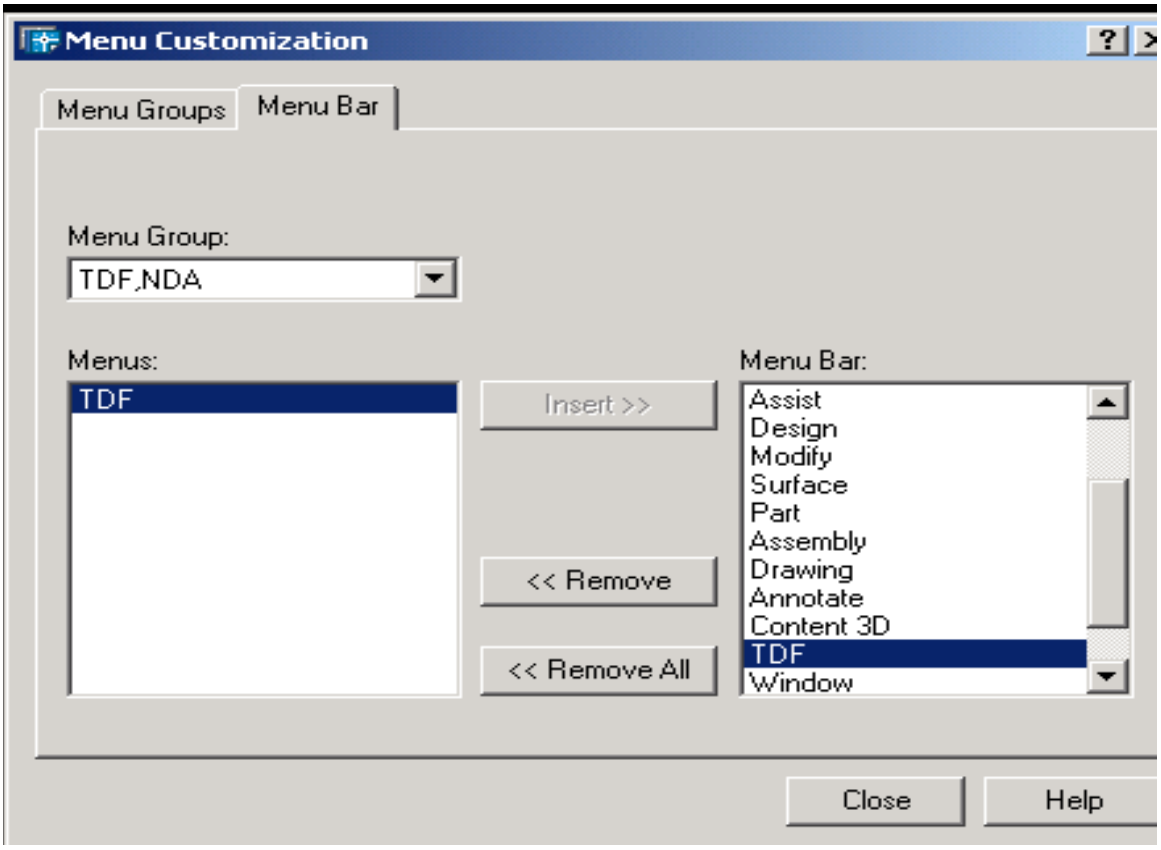


Fig. (3.16) TDF inserted in the menu bar

## 3.2 THE CUSTOMIZATION AUTOCAD PROGRAM

The function of the customized program is to draw the main standard parts for dies and mould in 2D and 3D drawing by using the AutoLISP language here bellow is the processes of the developments of the program:

### 3.2.1 Lower Die Block:

The lower die block is the main part of the die. Many companies are producing it as the standard part. The dies factory order the lower part as the entire part, in this part many operations can be done (drilling, tapping, grinding…etc) this operations depend on the requirements of the part to be manufactured.

The companies produce the standard parts e.g. (HASCO, FUTABA, VEBRO, MUSUMI…etc).

First the program creates to draw the lower die block in 2D drawing as square see Fig. (3.17) appendix (A) program (No.1).

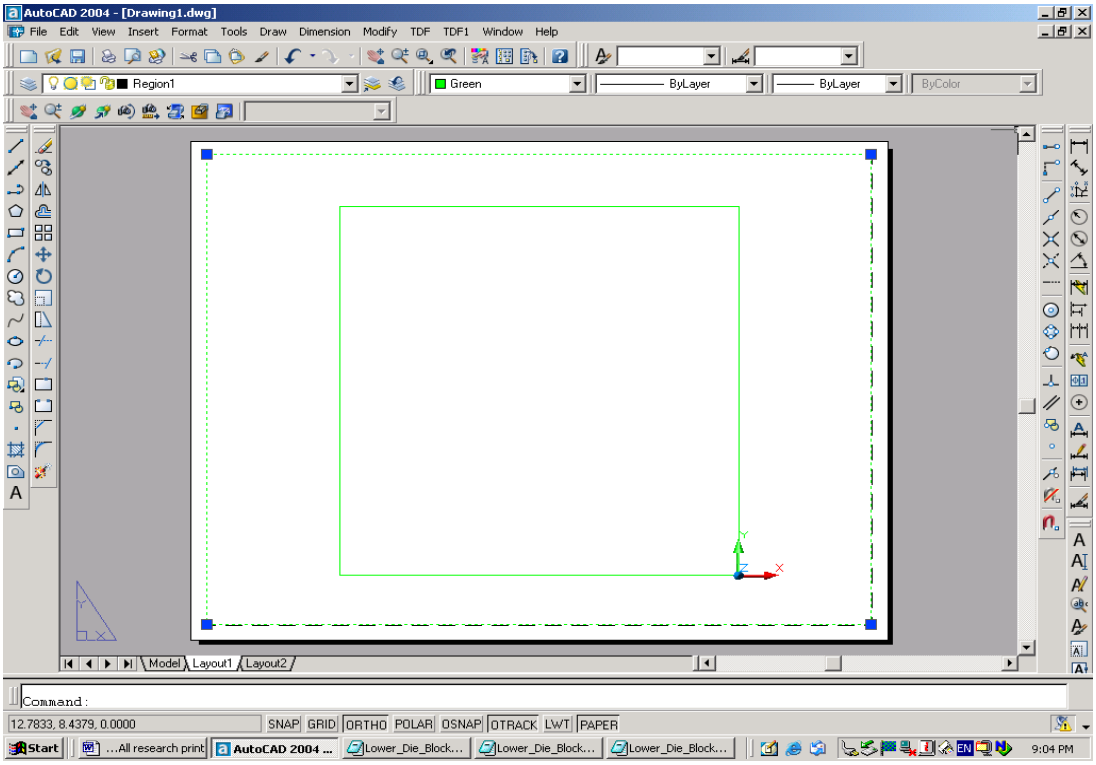


Fig. (3.17) Lower die block

Secondly the program creates to draw a hole in the lower die block in 2D drawing see Fig. (3.18) appendix (A) program (No.2)

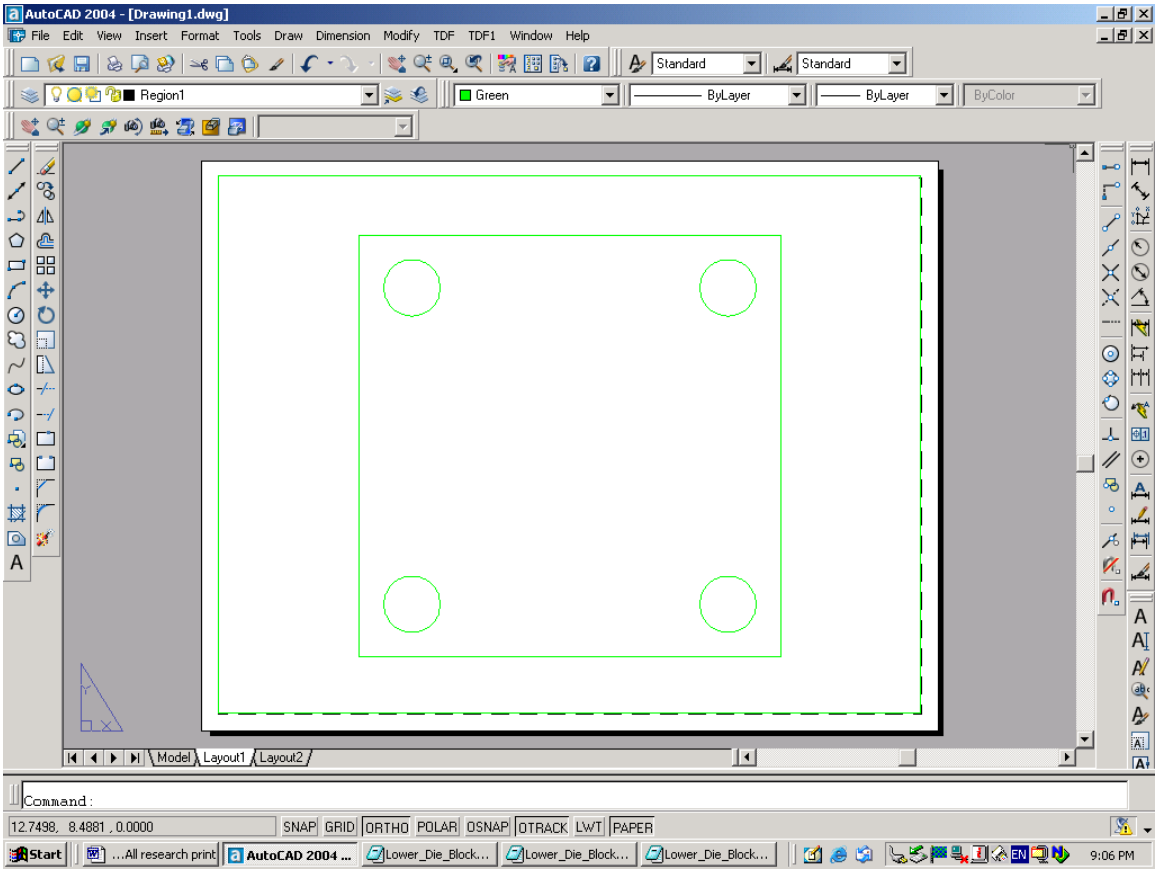


Fig. (3.18) Lower die block with holes

Thirdly the program developed to draw a lower die block in 3D drawing see appendix (A) program (No.3) Fig. (3.19)

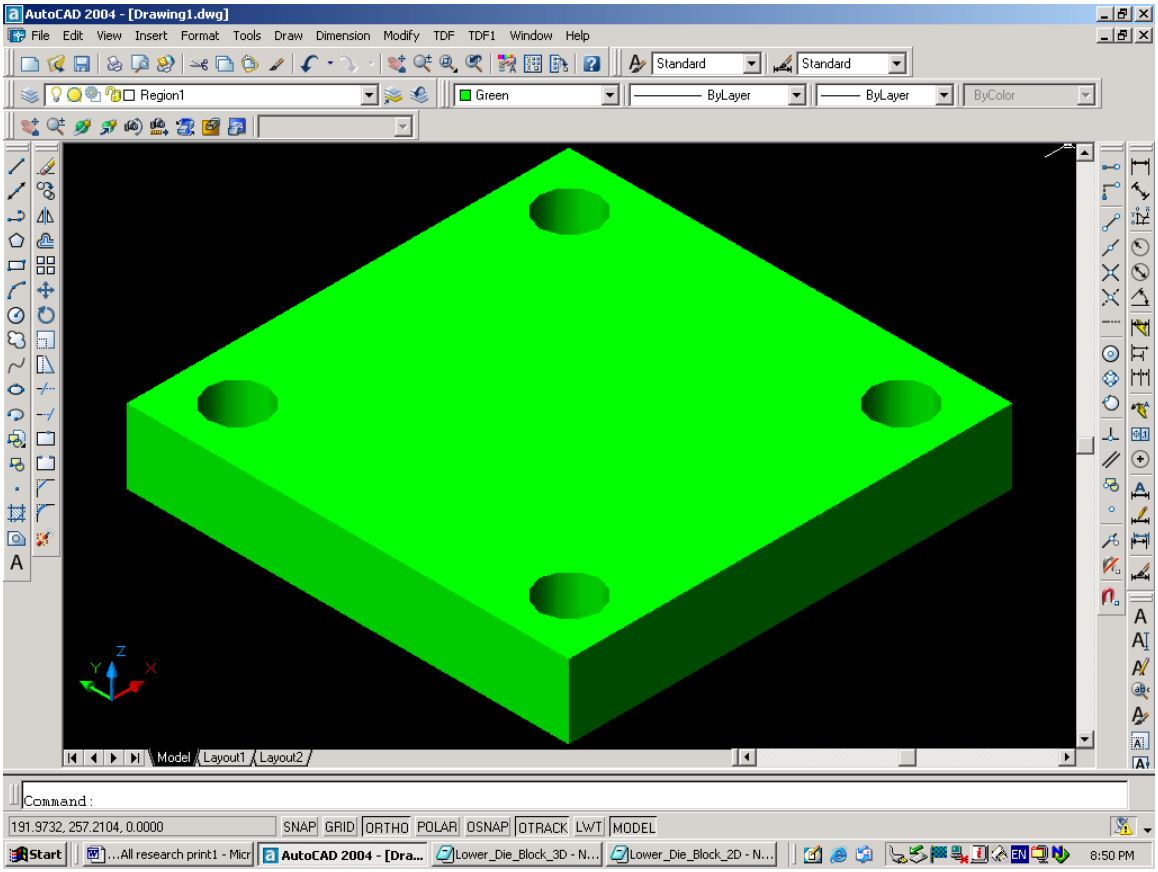


Fig. (3.19) Lower die block drawn in 3D

### 3.2.2 Guide Pin:

The guide pin is one of the main parts of the dies and also the companies produce it as the standard part here draw it in 3D as a shaft see appendix (A) program (No.4) Fig. (3.20)

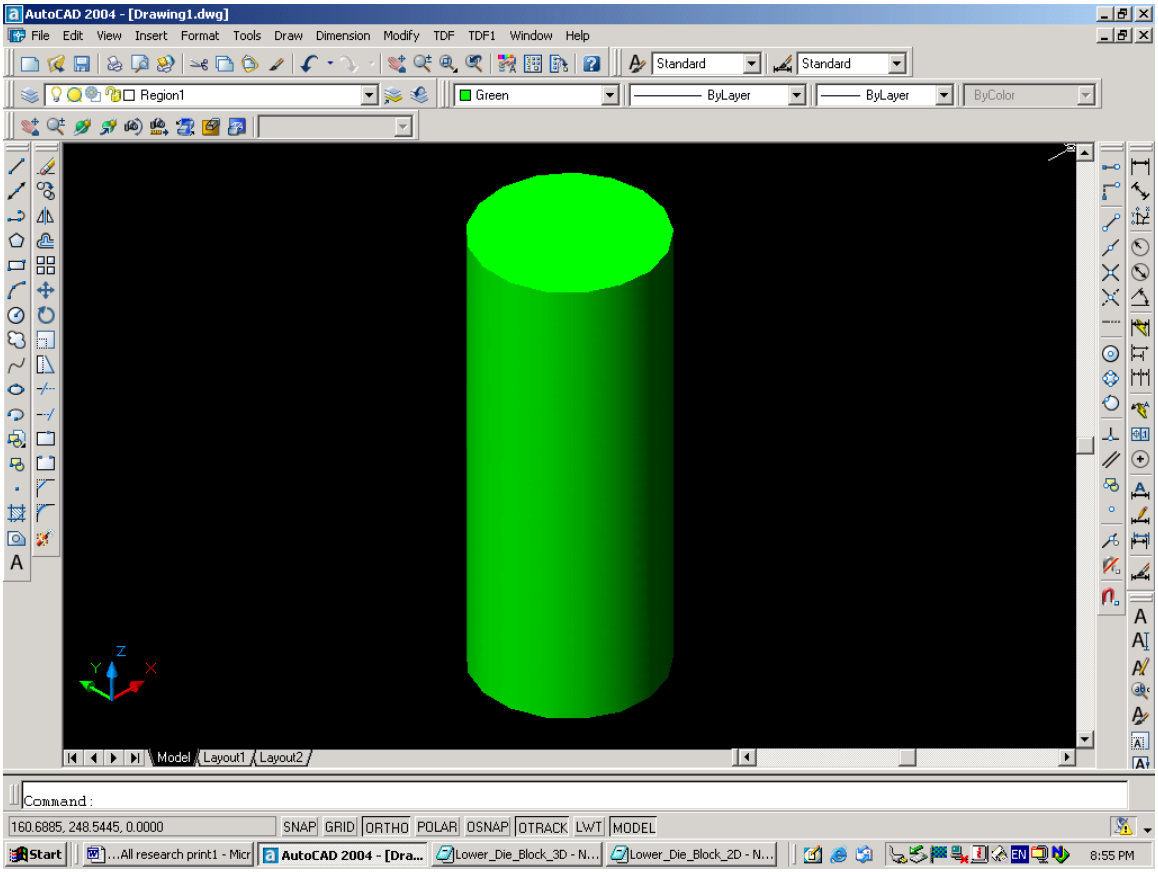


Fig. (3.20) The guide pin drawn in 3D

### 3.2.3 Assembly Die Blocks:

When drawn the main standard parts of the die here below can see the lower die block and the guide pin assembly together and make a copy from the lower die block and paste it on the top of the guide pin in the write position, for program see appendix (A) program No. (5) Fig. (3.21)

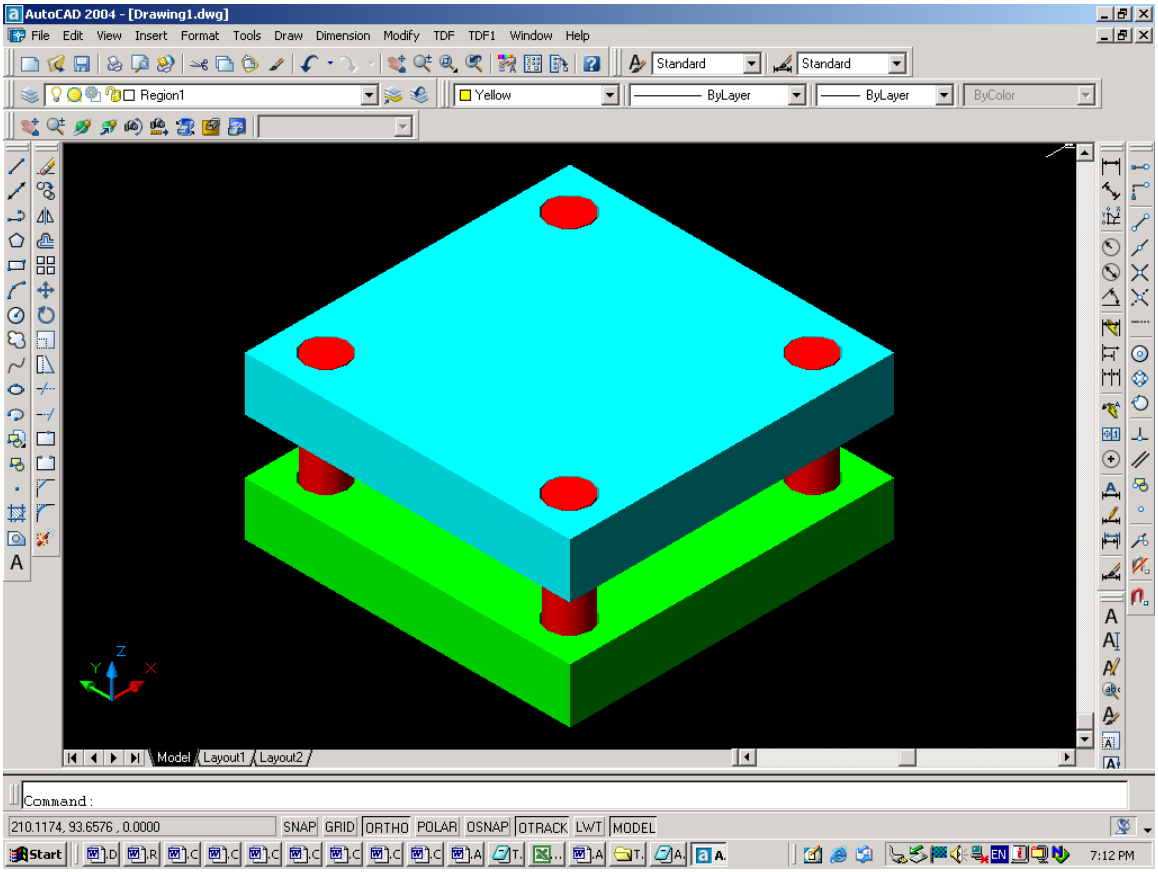


Fig. (3.21) The assembly die blocks

### 3.2.4 Drawing Of Gear:

The customize AutoCAD program can be easily draw the spur gear in different dimension just the program ask for the following:

- Center point of gear
- Number of teeth
- Module
- Height of gear
- Shaft diameter

Fig. (3.22) show the spur gear drawn by using the customize AutoCAD program, for the program code see appendix (A) program No. (6)

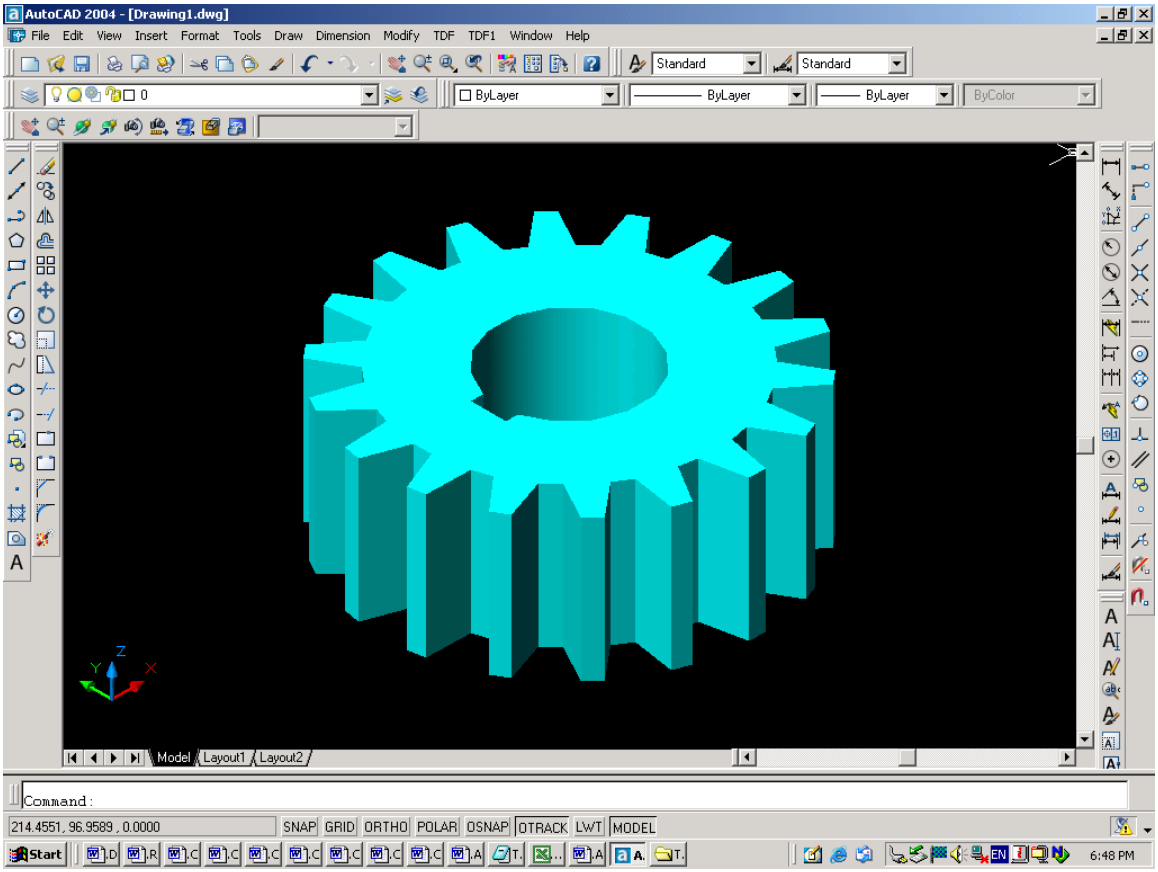


Fig. (3.22) The gear

CHAPTER FOUR

# RESULTS & DISCUSSION

## 4.1 RESULTS

Five technicians of good knowledge of AutoCAD were employed to produce drawings of dies and gears, using AutoCAD at a time and the customize AutoCAD at other time, the time recorded for each job is tabulated below:

Table (4.1) Drawings of die blocks assy. And gear by Draftsman (1)

| No. | Parts | AutoCAD Time/sec | Customize AutoCAD Time/sec | Difference time/sec | Time gaining by using Customize AutoCAD % |
|---|---|---|---|---|---|
| 1 | Die blocks Assy. | 334.8 | 51.84 | 282.96 | 84.52% |
| 2 | Gear | 548.4 | 20.84 | 527.56 | 96.20% |

Table (4.2) Drawings of die blocks assy. And gear by Draftsman (2)

| No. | Parts | AutoCAD Time/sec | Customize AutoCAD Time/sec | Difference time/sec | Time gaining by using Customize AutoCAD % |
|---|---|---|---|---|---|
| 1 | Die blocks Assy | 213 | 17.67 | 195.33 | 91.70% |
| 2 | Gear | 270 | 8.45 | 261.55 | 96.87% |

Table (4.3) Drawings of die blocks assy. And gear by Draftsman (3)

| No. | Parts | AutoCAD Time/sec | Customize AutoCAD Time/sec | Difference time/sec | Time gaining by using Customize AutoCAD % |
|---|---|---|---|---|---|
| 1 | Die blocks Assy | 240 | 20.74 | 219.26 | 91.36% |
| 2 | Gear | 273.6 | 10.77 | 262.83 | 96.06% |

Table (4.4) Drawings of die blocks assy. And gear by Draftsman (4)

| No. | Parts | AutoCAD Time/sec | Customize AutoCAD Time/sec | Difference time/sec | Time gaining by using Customize AutoCAD % |
|---|---|---|---|---|---|
| 1 | Die blocks Assy | 200.4 | 20.67 | 179.73 | 89.69% |
| 2 | Gear | 316.3 | 12.55 | 303.75 | 96.03% |

Table (4.5) Drawings of die blocks assy. And gear by Draftsman (5)

| No. | Parts | AutoCAD Time/sec | Customize AutoCAD Time/sec | Difference time/sec | Time gaining by using Customize AutoCAD % |
|---|---|---|---|---|---|
| 1 | Die blocks Assy | 88.8 | 17.67 | 71.13 | 80.10% |
| 2 | Gear | 200,4 | 8.45 | 191.95 | 95.78% |

A comparison between the number of commands require to draw the die blocks assembly by AutoCAD commands and Customized AutoCAD commands shown in table (4.6)

Table (4.6) Commands for AutoCAD and customized AutoCAD for die block

| No. | AutoCAD commands to draw die blocks assy. | Customize AutoCAD commands to draw die blocks assy. |
|---|---|---|

| | | |
|---|---|---|
| 1 | Draw rectangle | Select die block assy. From AutoCAD menu |
| 2 | Enter first corner | Enter first corner |
| 3 | Enter second corner | Enter second corner |
| 4 | Draw circle | Paste lower die block on upper guide pin |
| 5 | Enter diameter | Move upper die block to right place |
| 6 | Mirror circle | |
| 7 | Mirror two circle | |
| 8 | Extrude rectangle and circle | |
| 9 | Subtract circle | |
| 10 | Draw circle for guide pin | |
| 11 | Enter diameter of guide pin | |
| 12 | Extrude guide pin circle | |
| 13 | Copy guide pin multiple | |
| 14 | Copy lower die block | |
| 15 | Paste on upper guide pin | |

A comparison between the number of commands require to draw the gear by AutoCAD commands and Customized AutoCAD commands shown in table (4.7)

Table (4.7) Commands for AutoCAD and customized AutoCAD for gear

| **No.** | **AutoCAD commands to draw die blocks assy.** | **Customize AutoCAD commands to draw die blocks assy.** |
|---|---|---|

| | | |
|---|---|---|
| 1 | Draw circle | Select gear from AutoCAD menu |
| 2 | Enter center of outer circle | Enter center of gear |
| 3 | Enter diameter | Enter the number of teeth |
| 4 | Draw circle | Enter module number |
| 5 | Enter center of inner circle | Enter thickness of the gear |
| 6 | Enter diameter | Enter shaft diameter |
| 7 | Draw line from circle center | |
| 8 | Offset line | |
| 9 | Offset line | |
| 10 | Offset line | |
| 11 | Offset line | |
| 12 | Draw line | |
| 13 | Mirror line | |
| 14 | Trim to draw a tooth | |
| 15 | Array the tooth | |
| 16 | Draw circle for shaft | |
| 17 | Enter center of shaft circle | |
| 18 | Enter diameter | |
| 19 | Draw line center of shaft | |
| 20 | Offset line | |
| 21 | Offset line | |
| 22 | Trim to draw key way | |
| 23 | Modify lines to polyline | |
| 24 | Extrude gear and shaft | |
| 25 | Subtract shaft and key way | |

## 4.2 DISCUSSION

The preparation of the customized AutoCAD program it need a time to complete, but this time compare with time taken to draw many jobs it's negligible.

The results of using Customize AutoCAD program by different five Draftsman was detected and analyzed, and the conclusion obtained showing a large difference in results when using AutoCAD program and Customize AutoCAD program, that's the results show a big reduction in execution time

between (80.10 % to 91.70 % for Die blocks Assy. drawing) and (95.78 % to 96. 87 % for gear drawing).

In the assessment five technicians with good knowledge in using the computer and AutoCAD are employed. The two parts (Die blocks Assy. and gear) are draw on by these five technicians in using the two ways (AutoCAD program and Customize AutoCAD program).

The conclusions from the results obtained from the Customize AutoCAD program contribute with following:-

1. The program customized with the AutoCAD menu bar and now it's ready to setup with any computer built-up with AutoCAD program.
2. The time was saved due to the difference between customizing AutoCAD and AutoCAD.
3. The draftsman of this program will not need training, because the commands of customize AutoCAD program as the same of AutoCAD program commands.
4. The customized program minimized the cost of design by reducing the time required for drawing.
5. The number of basic commands in the customize AutoCAD are less than AutoCAD commands.

Here below histograms illustrate the difference in time obtain when drawing the two parts (Die blocks Assy. and gear) in the design section in Yarmouk Industrial Complex:

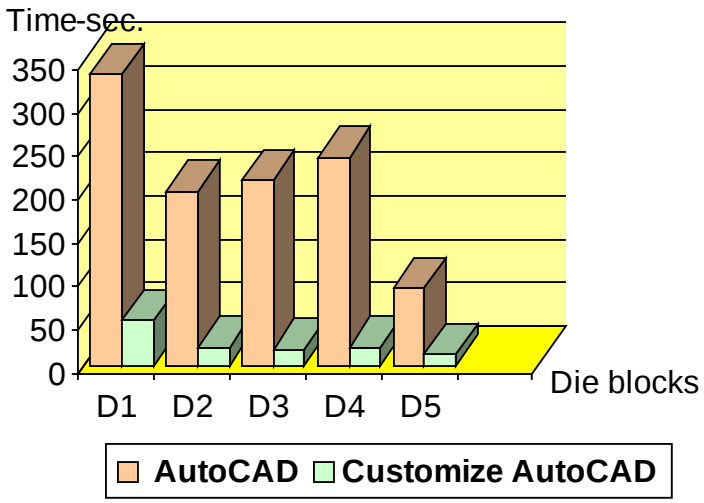


Fig. 4.1 The difference in time between AutoCAD and customize AutoCAD when draw a die blocks assy.

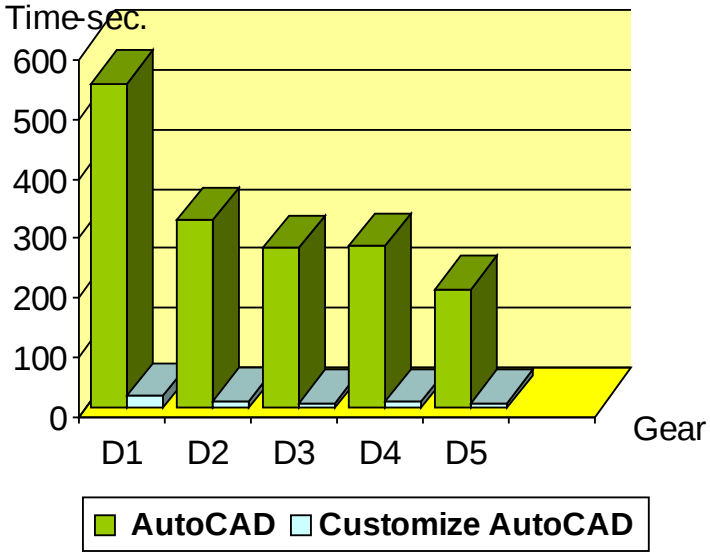


Fig. 4.2 The difference in time between AutoCAD and customize AutoCAD when draw a gear

# CHAPTER FIVE

# CONCLUSIONS AND RECOMMENDATION

## 5.1 CONCLUSIONS

The continual improvement of the computer programs dealing with design drawings leads to using simple and fast programs when customizing with in AutoCAD and MDT. These customizations offer more facilities for the designers of dies and moulds and gears…etc.

Customize AutoCAD program is one of the sequence of the continual improvement with which more complicated drawings are executed with somehow low effort, short time, and helps very much in solving problems that facing the users of the programs that can be interfaced with AutoCAD, the main findings can be listed as follows: -

The study reached to the point that the designs of customize AutoCAD program is by using AutoLISP language.

1- Install Customize AutoCAD program to any AutoCAD version..
2- Making applications for the customize AutoCAD program in design section of (Yarmouk Industrial Complex). The results obtained from this applications where discussed before in chapter four.

## 5.2 RECOMMENDATIONS: -

At the end of this research I know I am not arrive our proposal I am thinking about it in my mind because that there will be a following recommendation to whom they try to research in this subject: -

1- It is highly recommended to develop the work in (Yarmouk Industrial Complex) by introducing this program so as to carry out design drawing.
2- The researcher of this study recommends to prepare a programs for all the remaining standard parts in 3D drawing.

3- I am recommend to students to begin research in the subject of AutoCAD using the AutoLISP language because this language is the basic language using in AutoCAD programs.

4- The researcher of this study recommended to display the customize AutoCAD program to all engineering colleges in the Sudan for using and benefit from it especially the Department of mechanical.

5- Add the dimensions to 3D drawing obtained by customize AutoCAD program.

# REFERENCES

1. Kenny Ramage,**The AutoLISP Tutorials**, 2002, afralisp@mweb.com.na http://www.afralisp.com
2. Kevin Standiford, **AutoLISP to Visual LISP Design solutions for AutoCAD,** Thomson Learning, 2001, 1st Edition.
3. Terence M. Shumaker & David A. Madsen, **AutoCAD and its applications Advanced,** The Goodheart-Willcox Company,Inc, 1998, 1st Edition .
4. http://come.to/autolisppage Credit by Michiel de Bruijcker.
5. Joseph Edward Shigley & Charles R. Mischke, **Mechanical Engineering Design**, New York, McGraw-Hill,1989, 5th Edition.
6. Thomas E. French & Charles J. Vierck, **Graphic Science and Design,** New York, McGraw-Hill,1970,3rd Edition.
7. Ronald W. Leigh, **AutoLISP Programming**, 2002, 5th Edition, http://courses.home.att.net/autolisp/index.htm.
8. Habeeb Kattab habeeb, **Developing A Computer Program For Optimization Of The Manufacturing Process** (MSc Research).