

User Friendly Communication with CAD for the Disabled

By

Mohammed Nurul Hassan, B. Sc.

This thesis is submitted as the fulfilment of the
Requirement for the award of degree of
Master of Engineering (M.Eng.)

to


Dublin City University

July 2006

**Research Supervisor: Professor M. S. J. Hashmi and
Dr Bryan MacDonald**
School of Mechanical & Manufacturing Engineering

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Engineering is entirely my own work and has not been taken from the work of others save to the extent that such work has been cited and acknowledged within the text of my work


Signed: _____ (Candidate)

ID No.: 50162454

Date: 01-10-07

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Engineering is entirely my own work and has not been taken from the work of others save to the extent that such work has been cited and acknowledged within the text of my work

Signed: _____(Candidate)

ID No.: 5016245

Date: _____

Acknowledgements

First, I would like to express my sincere thanks and gratitude to Professor M.S.J. Hashmi for his constant encouragement and support. Without his guidance and continuous advice it would not have been possible to finish this thesis. I am honoured to have had him as my supervisor for this research.

I would also like to express my sincere thanks to Dr Brayn MacDonald for his invaluable contribution to the design of my system and to Mr.Keith Hicky, School IT system administrator, for his ongoing support.

Thanks also to my fellow graduate students Rudro, Tariq Chudory and Tutul for their helpful comments and joyfulness. All of them made the hard times easier to bear and the good times so menable.

Linguistic Communication with CAD

by

Mohammed Nurul Hassn, B.Sc.

ABSTRACT

In this research, by using an input device (keyboard, mouse) information may be transferred into the computer and drawings created in the Auto CAD can be edited using natural language. A system has been developed to generate mechanical objects, from basic primary drawings such as circle, arc, and line. Drawing commands are given in linguistic form and edit tool commands are given by using function keys from the keyboard or from the shortcut menu. The edit tools are fillet, chamfer, and trim. This project has the ability to magnify the image for better viewing. In this project a new technique has been introduced to select an object. Each object has been given a unique numerical name, thus allowing the object to be selected as many times as the name is chosen. The main goal of this project is to create a simple and easier way to draw objects without using a mouse. This will allow a physically disabled person or a person who is unable to use a mouse to draw a basic drawing by using a keyboard. This process is much more interactive and user friendly. The developed software can calculate intersection points and can write and read DXF files, which can be read by AutoCAD. After reading the DXF file it can redraw it on an AutoCAD interface. This software has been developed using Visual Basic 6, as part of the proposed system. It also provides a good foundation for a software operated by natural language communication.

Table of content

Title	Page
Declaration	ii
Acknowledgement	iii
Abstract	iv
Contents	v
List of Figures	vii
List of Tables	xi
Definitions	xii
Chapter 1: Introduction	1
Chapter 2: Literature Review	3
Chapter 3: Overview of the Developed System	12
3.1 Introduction	10
3.2 Software	11
3.3 Draw a Line	11
3.4 Draw a Circle	12
3.5 Draw an Arc	12
3.6 Fillet	14
3.7 Chamfer	16
3.8 Tangent	17
3.9 Save Drawing	18
3.10 Open a file	18
3.11 Mathematical Terms	19

Chapter 4. Algorithm	24
Chapter 5: Result and Discussion	26
5.1 Introduction	24
5.2 Line Line intersection	24
5.3 Line Circle Intersection	27
5.4 Line Arc Intersection	29
5.5 Circle Circle Intersection	33
5.6 Finding out Tangent Point	36
5.7 A Model of a Shaft	37
5.8 A Model of a joint Link	43
5.9 A Model of a Cover Lock	46
 Chapter 6: Conclusion and Suggested Future Work	 56
6.1 Conclusions	54
6.2 Suggested Future Work	54
References	58
Appendix: Source Code	62

List of Figures

<i>No.</i>	<i>Legend</i>	<i>Page</i>
1.(a)	Application Specific Keyboard.....	1
3.2	Main window (Interface of the software).....	13
3.3a,b	Start point of the line.....	13
3.3c,d	End point of the line.....	14
3.4a,b	Center coordinate of the circle.....	14
3.4c	Radius of the circle.....	14
3.5a,b	Center coordinate of the arc.....	15
3.5c	Radius of the Arc.....	15
3.5d,e	Start and End Angle of the Arc.....	15
3.2.1	An example on Auto CAD interface.....	16
3.6(a)	Two lines intersection	17
3.6(b)	Two lines intersect with intersection point... ..	17
3.6(c)	Asking for object number.....	18
3.6(d)	Asking for radius number.....	18
3.6(e)	After Fillet.....	18
3.7(a)	After Chamfer.....	19
3.8(a)	Tangent input box.....	19
3.8(b)	Before creating Tangent.....	19
3.8(c)	After creating Tangent.....	19
3.9(a)	DISCAD Save window.....	20
3.10(a)	DISCAD Open window.....	20
3.11(a)	Finding out intersection point between two lines.....	21
3.8(b)	Finding out Line Circle intersection point.....	22
3.8(c)	Finding out intersection point between two Circle.....	22
4	System Algorithm.....	24
5.2(a,b)	Line line intersection.....	26
5.2 (c)	New generated 4 lines.....	27

5.2 (d)	Line line intersection Block diagram.....	28
5.3(a,b)	Line Circle Intersection.....	29
5.3(c,d)	Line intersect Circle.....	30
5.4(a,b)	Line Arc intersection.....	31
5.4(c)	Line Arc intersection point.....	32
5.4(d)	After dividing Arc and Line at one point.....	32
5.5	Line Arc intersection Block diagram.....	33
5.4(e)	Normal Arc.....	34
5.4(f)	Abnormal Arc.....	35
5.5(a)	Circle circle intersection.....	35
5.5(b)	After intersection between two circle.....	36
5.5	Two Circle's intersection block diagram.....	37
5.6(a)	Finding out Tangent point.....	38
5.7(j)	Final drawn model of a shaft.....	45
5.8(g)	Model of drawn Joint link.....	48
5.9(h)	Model of a cover lock.....	53
5.10(g)	Model of a Link.....	55

Definitions

Active X technology: ActiveX is a standard that enables software components to interact with one another in a networked environment, regardless of the language(s) used to create them. Most World Wide Web (WWW) users will experience ActiveX technology in the form of ActiveX controls, ActiveX documents, and ActiveX scripts.

DXF format: The DXF format is a tagged data representation of all the information contained in an AutoCAD® drawing file. *Tagged data* means that each data element in the file is preceded by an integer number that is called a *group code*. A group code's value indicates what type of data element follows. This value also indicates the meaning of a data element for a given object (or record) type. Virtually all user-specified information in a drawing file can be represented in DXF format.

Fillet: Filletting connects two objects with a smoothly fitted arc of a specified radius. Although an inside corner is called a fillet and an outside corner is called a *round*, AutoCAD treats both as fillet.

Chamfer: Chamfering connects two nonparallel objects by extending or trimming them to intersect or to join with a beveled line. Lines, polylines, xlines, and rays can chamfer. With the distance method, specify the amount that each line should be trimmed or extended. With the angle method, also specify the length of the chamfer and the angle it forms with the first line.

Chapter One

Introduction

By using a keyboard and mouse one can draw an object on different types of drawing applications. By using an input device (keyboard, mouse) information can be transferred into the computer, which is the basic way one communicates with a computer. With a combination of keyboard and mouse a drawing can be edited or modified much more easily and quickly by a skilled designer. This is the way it has been done since drawing applications were first used. A customised keyboard has been introduced in the market, which has limited operation. The name of that keyboard is Application Specific Keyboards (ASK). The purpose of this keyboard is enhancing AutoCAD, an industry-leading product by Autodesk, Inc. This keyboard has been made for veteran, casual and student users. Figure 1. shows a customised keyboard for the AutoCAD. Still, physically disabled people will face difficulties to operate this type of keyboard.



Figure 1. Application Specific Keyboards

To edit or modify an object after it has been drawn it must first be selected. There are several existing methods to select an object. The easiest method is to select the object by using a mouse. Although an object can also be selected through the keyboard, this method is more complicated and thus more time consuming. Natural language command in verbal mode would be even better.

There are very few software for the physically disabled people so that they can operate the CAD and can draw design inside the Auto CAD interface. In this project a new technique has been introduced to select an object. Each object has been given a unique numerical identity thus allowing the object to be selected as many times as required.

The main goal of the present project is to create a simple and easy way to draw objects without using a mouse inside the Auto CAD interface. This will allow a physically disabled person or a person who is unable to use a mouse to draw a basic drawing by using a keyboard which is much more interactive and user friendly. The developed system provides a good foundation for a software to be operated by natural language communication.

The objective of the present project is to develop a very simple and less expensive system for people with physical coordination difficulties and for the people with learning difficulties, so that they can draw design without using a mouse, just using the Keyboard and object numbers inside the Auto CAD interface. This thesis has been organized as follows: In chapter two related works in the literature are briefly reviewed. Chapter three presents the overviews of the developed software. While in chapter four the algorithm developed has been described, chapter five describes the results and discussion, and chapter six presents conclusion and suggested future work. The listing of the software code is given in the Appendix.

Chapter Two

Literature Review

2.0 Introduction:

In this chapter a review is presented on research in the area of technological and computational aids to overcome the constraints encountered by intellectually as well as physically disabled people.

2.1 Technical Aid for Handicapped people:

Springer, et al [1] have tested practical suitability of the foot-mouse. The experimental result of the foot-mouse has been compared with the conventional mouse. Testing of the foot-mouse was performed either barefoot or with shoes on. Both handicapped and able-bodied subjects took part in their experimental laboratory studies. Experimental result has showed that handicapped subjects with the foot-mouse produced the same number of mistakes in a given time period as did able-bodied subjects with a conventional, hand-operated mouse. The experimental result has also showed that speed and accuracy problem has been reduced by using foot-mouse among hand and arm handicapped people.

Kawamura, et al [2] have presented a design philosophy for service robotics research and development, and described the current efforts. The role of service robotics and features has been elaborated in their design philosophy. Intelligent robotic-aid system, based on ISAC and HERO have been implemented based on such design philosophy. A voice command operated robot called HERO has been designed to feed physically handicapped people. They have also designed to solve robot navigation problems. Their goal was to improve the performance of a useful service at a reasonable cost through close robot-user interaction.

Noyes, et al [3] reviewed the development in the field of automatic speech recognition with particular reference to voice control of robotic arms and environmental control units. They evaluated and described a voice activated domestic appliance system and concluded that speech recognition applications for disabled people are well within the capacity of available technology. They also noted that it is primarily the lack of human factors work which is impeding developments in this field. Several human factors issues

have been identified. The most important of these being the need to increase the reliability of present speech recognisers, before they can confidently be incorporated into the lives of the disabled population.

Nicoud [4] analysed and proposed to develop voice synthesizer by using microcomputer devices for badly handicapped people. Voice can be synthesized, and the voice synthesizer can be controlled by a full or specialized keyboard. Three categories of devices were proposed for the handicapped to alleviate part of their mobility communicational and education problems. A motion problem can be eliminated by aid of wheelchair and prosthesis. Communication problem can be eliminated through sounds, signs, characters; special devices. Education problem can be eradicated through developing cognitive process and specific devices to replace what normal young children learn by themselves.

Miralles, et al [5] presented a case studying reengineering process starting from individual workplace where certain workers were capable of assembling the entire product and finishing with an assembly line implementation. The authors also analysed how the traditional division of work in single tasks, typical in assembly lines, becomes a perfect tool for making certain workers' disabilities invisible, and providing new jobs for disabled people while always taking into account certain special constraints

Chen, et al [6] approached to establish a functional assessment to measure the physical ability of handicapped people in response to specific tasks and environmental demands, so that visually impaired people can be placed in types of jobs that are compatible by rehabilitation and appropriate training. The objective of their study is to develop and integrate a computerized system which is called Vision Impaired Task and Assignment Lexicon (VITAL). This can measure the vision impaired workers' residual capabilities in order to provide necessary recommendations for job accommodations. VITAL includes two major modules: the disability index, and the ergonomics consultation module. The Disability Index (DI) represents the capacities of vision impaired. DI can be used in identifying the functional deficits and limitations of the visually impaired

workers. The ergonomic consultation module can provide recommendations regarding job and workplace design for the visually impaired workers.

2.2 Software for Handicapped people review:

Srinivasan, et al [7] developed a computer interface for visually handicapped people. A paperless Braille display, speech converter and IBM/compatible PC is the part of the interface. The system has been made for the visually handicapped people. The system has been used to teach Braille codes, programming, word processing, etc.

Oakey [8] developed Rapidtext software especially for the handicapped people. . The translated output of steno writer's have been presented into English words and displayed on a computer monitor by Rapidtext software. This system has allowed the deaf to receive equal access to the full text of spoken communication. This is the first time that Microcomputer steno-interpreting technology has been applied to allow the handicapped an equal access to the spoken world. Voice recognition is one of the concerns of DISCAD to draw in CAD interface from current developed software.

Coldwell [9] developed artificial intelligent software for satisfying the need of autistic people. Autistic children and, perhaps, other intellectually handicapped children sometimes have seemingly inherent skills that are difficult to understand. Autistic children have an inherent ability in mathematics but often can not use of symbols to benefit from it. The developed software will help autistic people to communicate and exchange their thoughts among the non autistic people.

Hawley, et al [10] developed an integrated control system for the disabled people. The Function of that system is sending multiple commands from a single input device, so that multiply handicapped users can switch efficiently between wheelchair control, communication, computer access and control of their environment without third-party help. The design philosophy has concentrated upon utilizing, wherever possible, commercially available assistance devices and remotely controlling these via logic-based integrated control systems tailored to the needs and abilities of the individual client. The system has the facility to utilize software based communications, keyboard emulation and environmental control packages and business and education software.

These authors have also found that it is easier to setup this system for the physically disabled people.

2.3 Handicapped people review:

Seeland, et al [11] undertook a survey of the Isle of Mainau located on Lake Constance in Southern Germany. They have found that people with officially recognised disabilities feel stigmatised by green space that is specially designed for visitors with handicaps. They have also noticed that people with lighter handicaps liked to have more attention and services rendered to them. 'Standard users', particularly those of higher income with better education, are reluctant to concede the entire island park's design and infrastructure to accommodate the needs of disabled visitors.

2.4 Use of CAD:

Park, et al [12] developed the outer body parts of a mid-size humanoid robot by focusing on the use of an integrated application of CAD/CAM/CAE and rapid prototype (RP). Most parts were three-dimensionally designed with 3D CAD, which enables effective connection with CAE analyses, the basis of which lays in kinematic simulation and structural analysis. To reduce the lead time and investment cost of developing parts, RP and CAM are selectively used to manufacture master parts for the body. They have successfully developed a prototype of Bonobo with a relatively low time and investment cost. This system can be integrated with DISCAD so that physically disabled people can operate CAM to manipulate the robotic body.

Nagata, et al [13] robotic sanding system. They have described two features of the robot sander. One is that the polishing force acting between the tool and wooden work piece is delicately controlled to track a desired value, e.g., 2 kgf. The polishing force is defined as the resultant force of the contact force and kinetic friction force. The other is that no complicated teaching operation is required to obtain a desired trajectory of the tool. Cutter location (CL) data, which are tool paths generated by a CAD/CAM system.

They have done few experiments to show the effectiveness. The robot sander can be useful for the disabled people by using the current developed software “DISCAD”.

Chen, et al [14] developed intelligent software prototype. Using feature extraction this software can find out dimension of the cylindrical surfaces in mechanical parts for their two-dimensional drawing automatically from their three-dimensional part models. They have successfully demonstrated to locate the dimensions of all the holes in the multi-spindle headstock of a modular machine tool by using intelligent dimensioning software prototype. Dimension is important in mathematics and drawing because it gives a precise parameterization of the conceptual or visual complexity of any geometric object. In fact, the concept can even be applied to abstract objects which cannot be directly visualized. For example, the notion of time can be considered as one-dimensional, since it can be thought of as consisting of only "now," "before" and "after." Since "before" and "after," regardless of how far back or how far into the future they are, are extensions, time is like a line, a one-dimensional object.

Chen, et al [15] developed an intelligent prototype software for generating mechanical product assembly drawings automatically from their 3-D assembly model made with existing CAD systems to reduce the time of product design and ensure the high quality of assembly drawings. In the current developed software assemble model drawing can be applying for the disabled people so that disable people can draw assemble drawing by using keyboard.

Prabhu, et al[16] developed an intelligent system which can extract features from engineering drawings created in CAD format. Natural language and syntactic pattern recognition are used as techniques. They apply this system to detect a generic class of features like hole, pockets, steps, slots, bosses, etc. In this project features can be extract from user after verifying data. Current project can draw in the Auto CAD interface from the Keyboard command.

2.5 Artificial intelligent or AI:

Varró [17] presented a general framework for an automated model transformation system. This method starts with a uniform visual description and a formal proof concept of the particular transformations by integrating the powerful computational paradigm of graph transformation, planner algorithms of artificial Intelligence and various concepts of computer engineering. Modelling concept is the most important in the proposed work. Daniel Varro has given a conceptual idea to process a complex systems that requires a precise checking of the functionality and dependability attributes of the target design.

Hauser, et al [18] found that artificial intelligent methods are necessary to draw complex models and they have also found that it is not necessary to cover all cognitive aspects for all potential design tasks and domains with one single specific problem solving method or architecture. They introduce the level of cognitive architectures as a level on which experience in the implementation of knowledge-based design support for dedicated design tasks in structural engineering. In this project AI has been applied which generate complex drawing and by using AI complex intersection point has been found out.

Su, et al [19] developed an intelligent hybrid system to integrate design, and manufacture product design specification, conceptual design, detail design, process planning, costing and CNC manufacturing. They have also found that production cost and time can be reduced by using this system. Generating drawing in such a quick and short time and less effort is important now a days. In their work two types of applications are being used - one is front end where user gives basic data. Then their software checks the validity of those data, and sends those data inside the Auto CAD for drawing via active X technology.

Taraban [20] The author has conducted five experiments using artificial grammar with gender-like noun subcategories to test the hypothesis that syntactic context models are sufficient for category induction. The first experiment has validated a computer based paradigm for artificial language learning. The second has shown that direct instruction

is one way to draw a learner's attention to the defining morphemes, and bring about category induction. The remaining experiments have shown that blocking learning trials by using nouns as the blocking factor draws the learner's attention to the correlated subsets of grammatical morphemes, and leads to category induction.

Bemden, et al [21] have presented a paper dealing with the way to train deaf to lip-read and to be able to use cued-speech. They have introduced a computer - aided learning method which is based on a phonetic transcription of word or sentences. In this method the phonetic transcription of words or sentences are compared to the student's response from the keyboard. They therefore, have described a matrix- based algorithm. However, the pedagogical decision module is yet to be designed

Celano, et al [22] investigated the problem of optimal pass schedule design in multi-pass wire drawing processes. They have proposed an automatic design procedure based on an effective artificial intelligence technique, called Simulated Annealing (SA). They have developed an algorithm designed to reduce stress on the material balance drawing. They have tested their algorithm with a set of industrial sequences for wires of different materials.

Nassehi, et al [23] examined the application of distributed artificial intelligent methods. They have collaborated multi-agent systems in designing an object-oriented process planning system for prismatic components in a STEP-NC compliant environment. Multi-Agent System for Computer Aided Process Planning (MASCAPP) is designed and specified. They have completed two design components and process plans. They have done simulation on machine control.

Schleiffer [24] explained the fundamental issues of agent intelligence. He has elaborated general agent architecture, emphasizing aspects of perception, interpretation of natural language and learning. Particular agent behaviour is partitioned in the subsequent phases of performing and interpreting observations, selection strategies for action execution and evaluating the usefulness of interpretations and executed actions. Each of these phases is modelled as a sequence of pattern recognition, evaluation of the usefulness of known patterns and modification of open and distributed environments.

He discusses the aspect of intelligence with regard to the agent model that has been developed.

Yan, et al [25] developed a database management system, computer-aided wear particle image processing system and an artificial intelligent system for oil monitoring. Their results show that a combination of information technology and oil monitoring increases the speed of oil analysis, and conveniently and precisely manage the information.

Yang, et al [26] developed an artificial intelligence system to help online guide learners with similar interests among reasonably sized learning communities. They use a multi-agent mechanism to organize and reorganize supportive communities based on learners' learning interests, experiences and behaviours. Simulated experimental results show that algorithms can improve the speed and efficiency in identifying and grouping homogeneous learners.

Sugisaka, et al [27] developed an artificial brain for a life robot. From voice and vision they have developed to control the wheel, head control, camera control, and speech, touch screen and LED. They have developed a robot, named Tarou that can perceive an environment, interact with humans, make intelligent decisions and learn new skills. By using speech recognition physically disabled can draw by voice command in the AutoCAD interface.

Leon, et al [28] developed a system to detect changes in the emotional status of a subject. This method has been used to detect Artificial Neural networks, statistical mechanisms, and Physiological measures. Their results show that it can distinguish changes from neutral to non- neutral emotional states.

2.6 DXF:

Mansour et al [29] developed an automated and interactive procedure, and demonstrated a new concept for generating bills of quantities. A user friendly interface, dynamic linking to the structural drawing and tracking of bill of quantities modification system has developed. Interactive automation has been used to compute the cost of the

structural skeleton. The area and volume for any structural shape, including circles and polygons have been determined after finding out coordinates from the DXF file format. The extracting method is a new technique for structural engineers and quantity surveyors. The same technique has been used in the DISCAD during extracting information from the DXF file in the current project.

Yamaguchi, et al [30] developed 2D finite element system from DXF file format. They have proposed the usefulness of the system and clarified through the magnetic field analysis of an electromagnet and a permanent magnet motor. DXF file format of this project is one of the basic elements for future improvement which could be useful in case of other software to read and modify any drawing.

2.7 Language Communication:

Kirby, et al [31] studied lexico-syntactic aspects of language and found fertile ground for artificial life modeling. They argue against a model of survey, and demonstrated that Artificial Life techniques have a lot to offer an explanatory theory of language. Using Computational simulation they concluded that theoretical linguistics is an appropriate response to the challenge of explaining real linguistic data in terms of the processes that underpin human language.

2.8 Summary

The review of literature shows that considerable research effort has been devoted to develop various software and hardware tools to assist disabled population to one or more aspect of their disability whether intellectual or physical. However, the best knowledge of this researcher these is very little evidence of research to develop a software for physically disabled people to operate CAD system through natural language based typed or voice operated instruction. The present research was thus aimed at developing such a software as an initial step to use linguistic instruction to operate CAD system without the use of a mouse.

Chapter Three

Overview of the Developed System

3.1 Introduction:

The software system developed in this research project has the capability to produce a design by drawing using only the keyboard. The system can be used to draw moderately complex object. The software is called DISCAD and it has been developed by the author in Visual Basic 6.

Though the system can use either the keyboard or the mouse, the software has been developed with particular attention to disabled people, so that they can use this software using only the keyboard.

The whole system can be divided into two distinct parts. The first concerns with the Drawing, in which a Line, Circle or Arc can be drawn. The second part is “Editing” in which the object can be modified or edited by using Chamfer, Fillet and Trim so that the main drawing can turn into a meaningful shape or design.

The first section of this chapter will explain how to draw all the basic objects. Whilst the second section will explain how to edit them.

3.2 Software:

When DISCAD software starts to execute it provides a single document graphical user interface (Figure 3.1). This software focuses mainly on drawing the object on the Auto CAD interface.

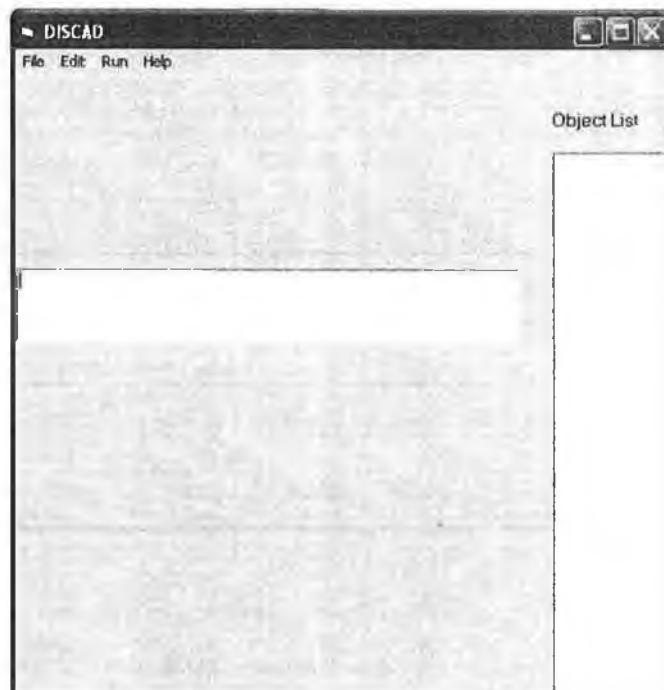


Figure 3.1: Main Window

A basic drawing can be inserted by the object name.

3.3 To draw a line:

Place the word “line” inside the text box then press the “F5” key from the keyboard to execute the code. Four input boxes will pop up for the coordinates.

Figure 3.2(a) shows the box for the X coordinate of the start line. Figure 3.2(b) shows the box for the Y coordinate of the line.

Start point of the line:

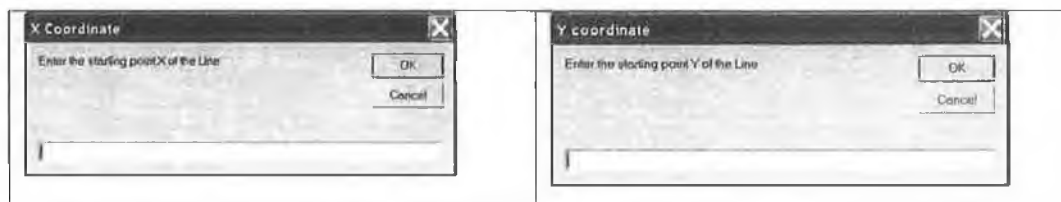


Figure 3.2(a) window for X coordinates Figure 3.2(b) Window for Y coordinates

Figure 3.2(a) Shows the box for the X coordinate of the endpoint of the line. Figure 3.2(d) shows the box for the Y coordinate of the endpoint of the line.

End point of the line:

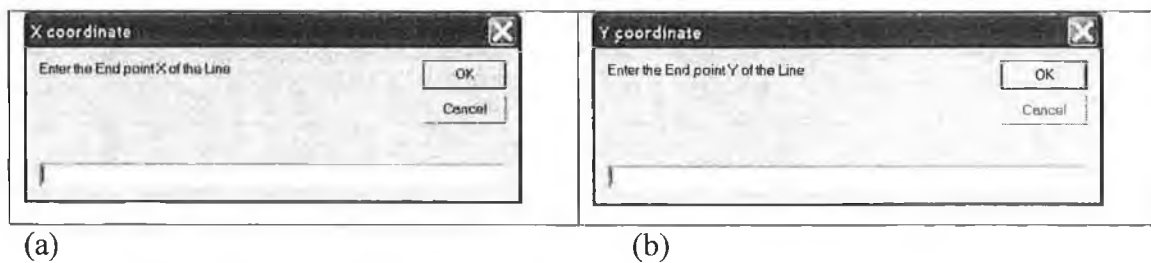


Figure 3.2 Windows for the endpoint X and Y coordinates.

3.4 To draw a circle:

Write “circle” inside the text box then press “F5” to execute the code. Three input boxes will appear to accept two-centre coordinates and one length of the radius data. Figure 3.3(a) shows the box for the X coordinate of the centre of the circle. Figure 3.3(b) shows the Y coordinate of the circle and figure 3.3(c) shows the box for the length of the radius of the circle.

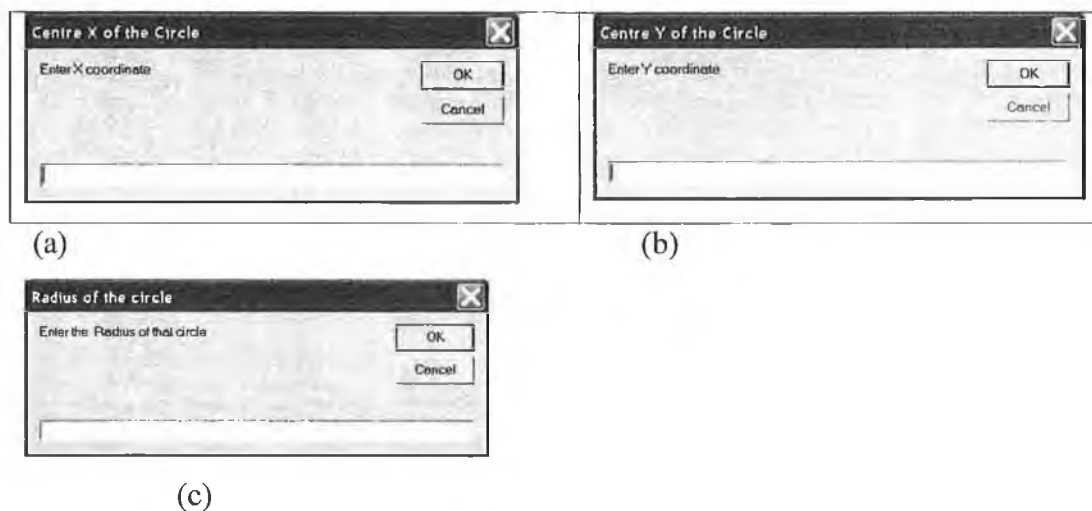


Figure 3.3(c) windows for the X and Y coordinates and length of the radius

3.5 To draw an ARC :

Write “arc” inside the text box then press “F5” to execute the code. Five input boxes will appear to accept data from the user. The function of each input box is described In Figure 3.4 (a) to (c)

Figure 3.4(a) and (b) shows two input boxes for the centre coordinate of the arc and Figure 3.4 (c) input box shows the length of the radius. While Figure 3.4 (d) and (e) input boxes shows the start and end angles of the arc.

Centre coordinate input box of the Arc:

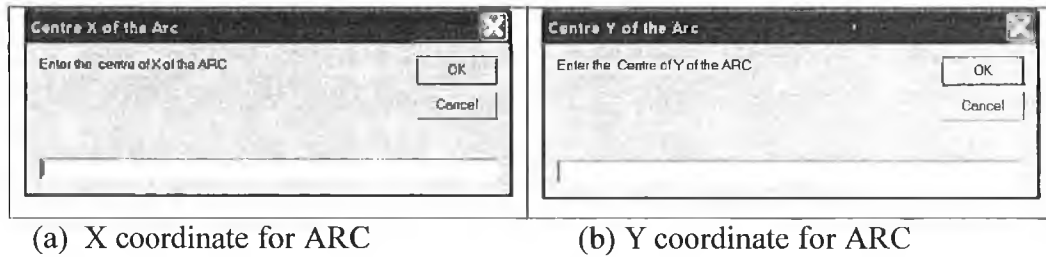
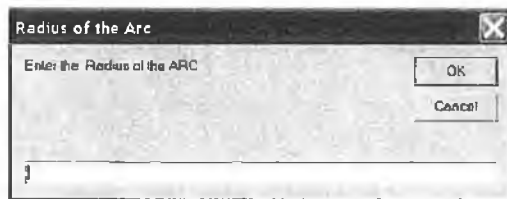


Figure 3.5(c) input box shows the length of the radius.

Radius input box of the arc:



Angles of the ARC:

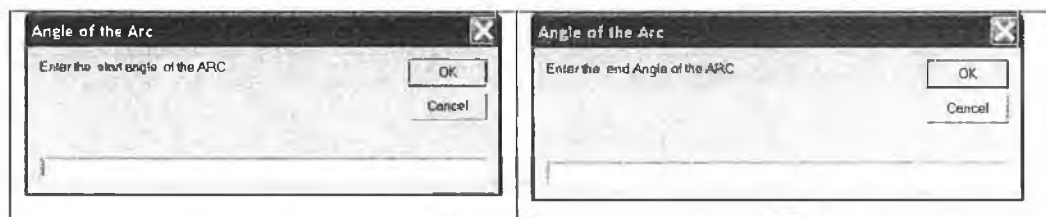


Figure 3.4 Windows for drawing an ARC

In order to quit from the middle of the program simply press the “cancel” button. After taking all the valid data from the user, the program will draw the basic drawing into the Auto CAD interface with the object numbers. Figure 3.5 is shown as an example.

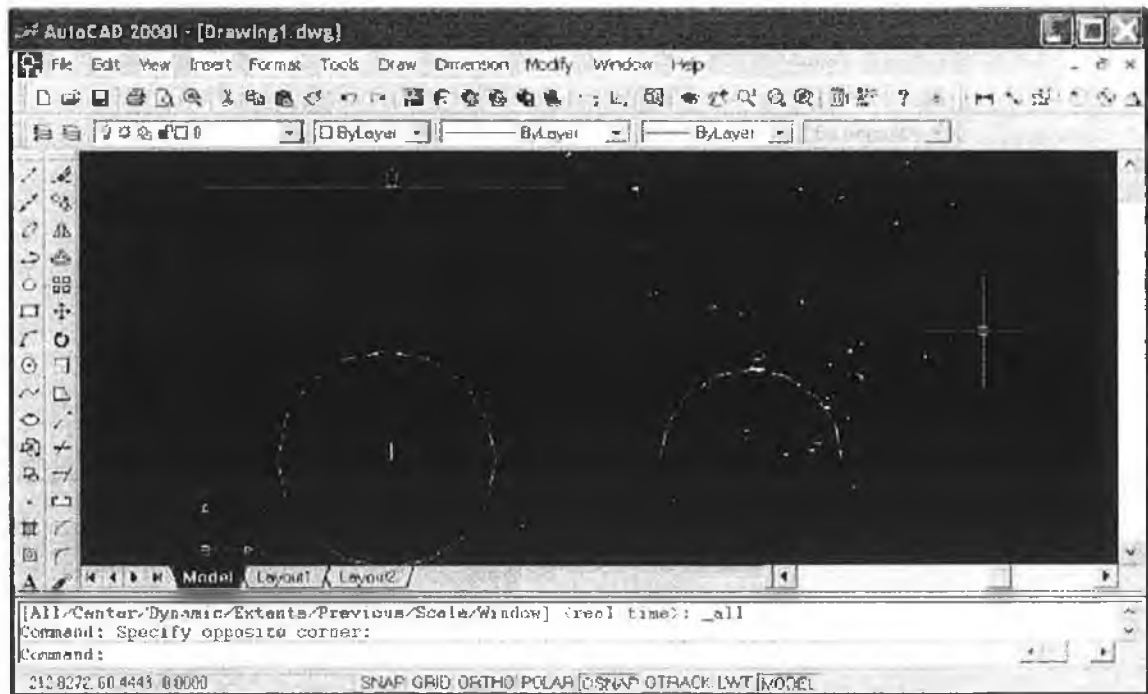


Figure 3.5 On AutoCAD Interface Line, Circle and Arc have been drawn.

After drawing on the AutoCAD interface then press Ctrl+D and the object will be divided by the intersection point. If it finds a new intersection it will give it a new object name for further manipulation or editing.

3.6 To do Fillet:

Two intersected lines have been drawn on the AutoCAD interface. Figure 3.6(a) shows an example.



Figure 3.6(a) Shows two intersecting lines.

Now to get the intersection point press Ctrl + D. Figure 3.6(b) shows 4 distinct lines based on the intersection point. Each line now can be edited by using the object name. i.e. 0, 1, 2, 3, 4

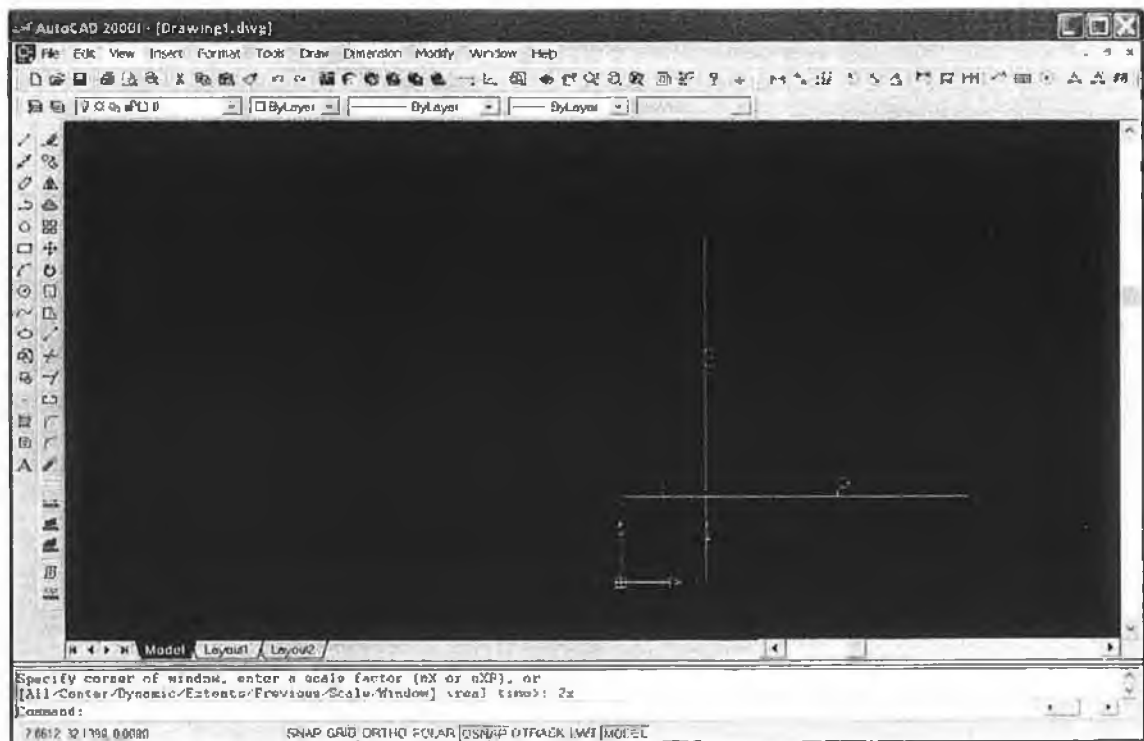


Figure 3.6(b) Window showing 4 lines.

To fillet press Ctrl+f from the keyboard. DISCAD will prompt two input boxes to take object name.

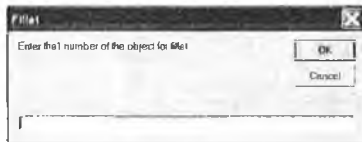


Figure 3.6(c) Asking for object name of the fillet.

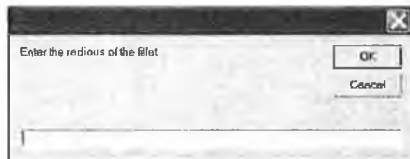


Figure 3.6(d) Asking for the radius of the fillet.

Figure 3.6(e) is showing the fillet:

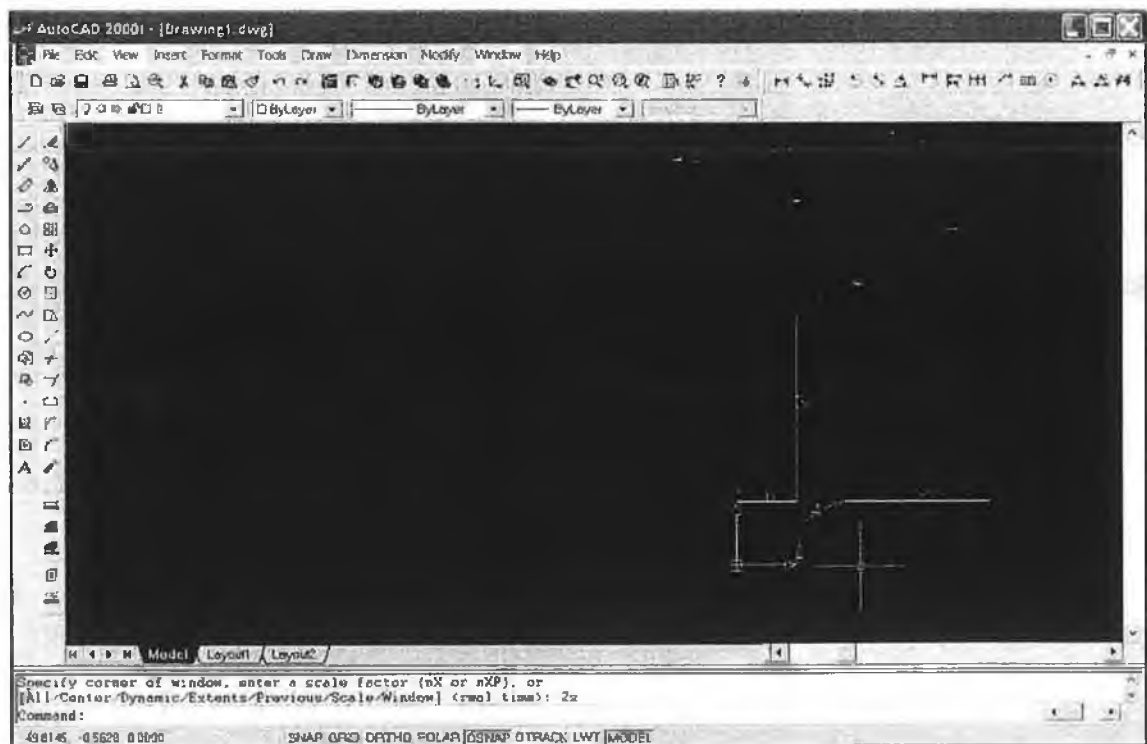


Figure 3.6(e) Showing the drawing after fillet.

3.7 To Do chamfer:

Just press Ctrl+C and two input boxes will be prompted to accept object name. Then it will ask for the length of the chamfer. Figure 3.7 shows the chamfer.

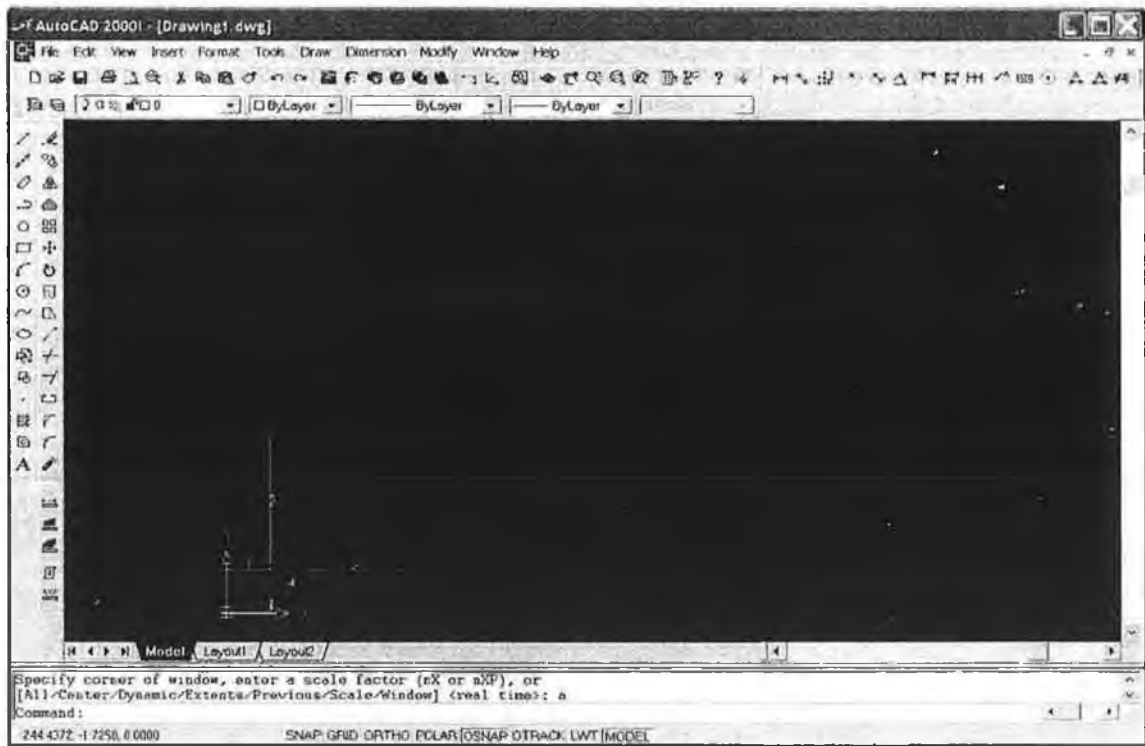


Figure 3.7 Shows the drawing after chamfer

To trim press Ctrl+t and DISCAD will prompt an input box to trim the object. After getting the object number it will delete that object from AutoCAD interface.

3.8 To draw a tangent line between two-spheres:

Press Ctrl+g and DISCAD will prompt two input boxes for the object name of two spheres. Figure 3.8(a) is showing the dialogue box while Figure 3.8 (b) shows the dream spheres.

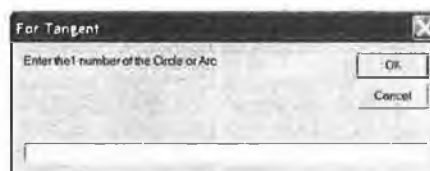


Figure 3.8(a) dialogue box.

After getting the object names, it will draw a tangent line between two spheres. Figure 3.8(c) shows the tangent line between two spheres.



Figure 3.8(b) Before creating tangent. Figure 3.8(c) After creating tangent.

To Zoom In press Ctrl+ ↑ To Zoom out Press Ctrl+ ↓

3.9 To save a drawing:

Press Ctrl+S from the keyboard and DISCAD will open a save window and prompt the user to supply a name. If the user presses the Enter key by mistake without giving any name at that time, the DISCAD save menu will not disappear until it receives a name from the user.

The save Window is shown in figure 3.9.



Figure 3.9 DISCAD Save Window

3.10 To open a file:

Just press Ctrl+O and DISCAD will open an open window. It will prompt the user to select a file name inside the File name text box. Press the Enter key and it will draw on an Auto CAD interface.

Figure 3.10 Shows the window used to open a file.



Figure10 DISCAD Open Window

3.11 Mathematical Formulation:

To determine the Intersection points between two lines are two parametric equations have been used. Where $(x11, y11)$ and $(x12, y12)$ are the start point and end point of first line and $(x21, y21)$ and $(x22, y22)$ start point and end point of the second line as shown in Figure 3.11 (a)

$$\text{Intesectionpoint_X} = x11 + dx1 * t1$$

$$\text{Intersectionpoint_Y} = y11 + dy1 * t1$$

where

$$t1 = ((x11 - x21) * dy2 + (y21 - y11) * dx2) / (dy1 * dx2 - dx1 * dy2)$$

$$dx1 = x12 - x11$$

$$dy1 = y12 - y11$$

$$dx2 = x22 - x21$$

$$dy2 = y22 - y21$$

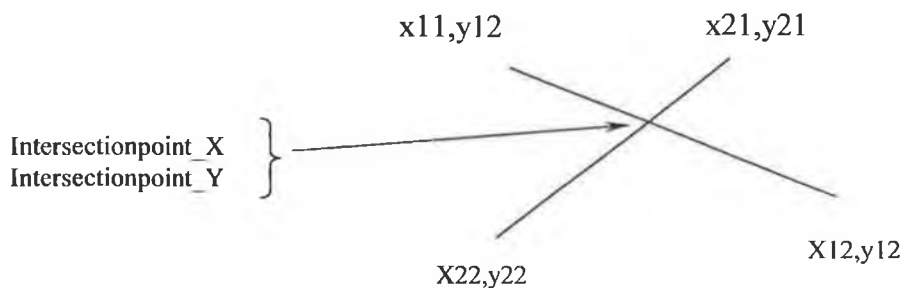


Figure 3.11(a) determining intersection point between two lines.

Line circle intersection:

As described below

$$P = P1 + u (P2 - P1)$$

where P expresses the (x, y) intersection point and two points of the line P1 (start point) and P2 (end point) where P1 is $(x1, y1)$ and P2 is $(x2, y2)$ as shown in Figure 3.11 (b)
Considering all the coordinates $x = x1 + u (x2 - x1)$; $y = y1 + u (y2 - y1)$

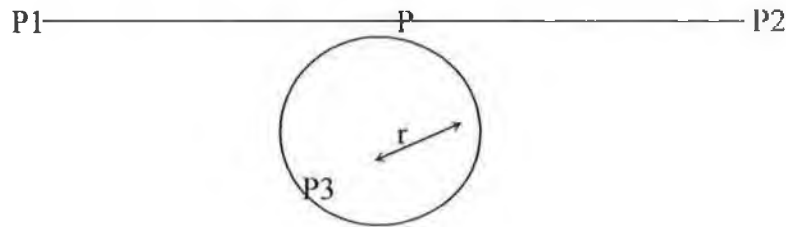


Figure 3.11(b) Line Circle Intersection

where

$$u = (-b \pm \sqrt{b^2 - 4 * a * c}) / 2a$$

$$a = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

$$b = 2[(x_2 - x_1)(x_1 - \text{centreX}) + (y_2 - y_1)(y_1 - \text{centreY})]$$

$$c = \text{centreX}^2 + \text{centreY}^2 + x_1^2 + y_1^2 - 2[\text{centreX} * x_1 + \text{centreY} * y_1] - r^2$$

The characteristics is determined by the expression with the square root of $(b^2 - 4 * a * c)$

If it is less than 0 then the line does not intersect the circle

If it is equal to 0 then the line is tangent to the circle

If it is greater than 0 then the line intersects the circle.

To determine the intersection point between two circles one parametric equation has been used. It is $p_3 = p_0 + a(p_1 - p_0)/d$

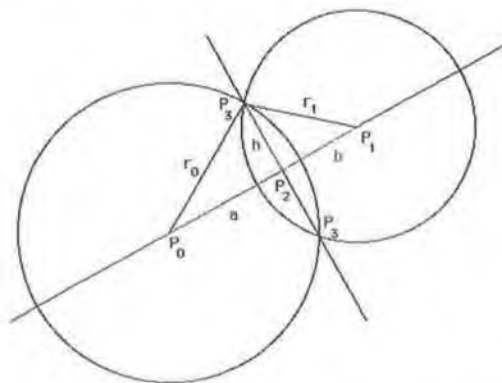


Figure 3.8(c) Circle to Circle intersection

where

$$p_0 = (x_0, y_0), p_1 = (x_1, y_1) \text{ and } p_3 = (x_3, y_3)$$

$$a=(r_0^2+r_1^2+d^2)/(2d) \text{ and } d=\text{abs}(P_1-P_0)$$

as shown in Figure 3.11 (c).

Chapter Four

Algorithm

There are three steps which need to be followed in order to draw a model using DISCAD software.

1. Draw Basic object by Interactive method.
2. Divide the basic object by the intersection point.
3. Edit them by using edit tools name.

The System flow chart can be described as shown in Figure 4.1 below.

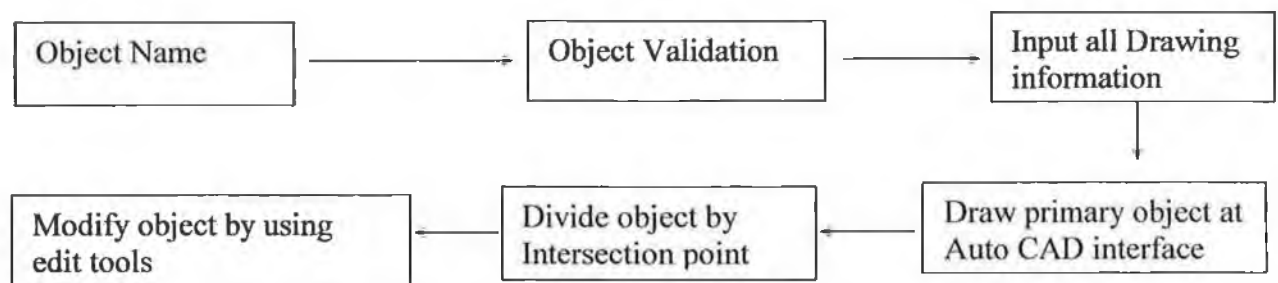


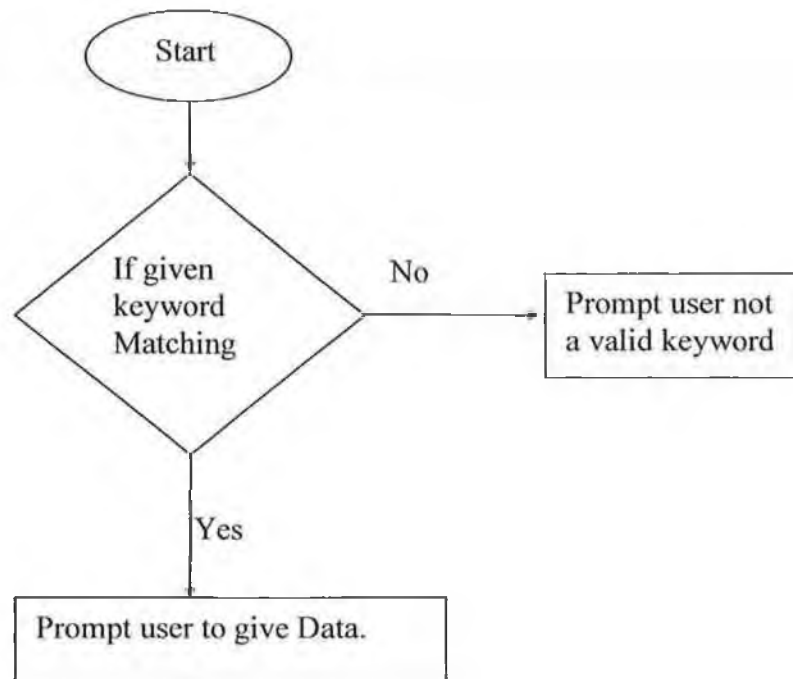
Figure 4.1 Algorithm Flow Chart.

Object Name:

Chapter three described how to draw the basic drawing into the Auto CAD interface.

When DISCAD recognizes the keyword from the user it will move on to the second step for the basic coordinates of the object.

The IsCommandOk class is responsible for matching the keyword that has been given by the user. In this section it compares each keyword, like line, circle, arc etc. If the user gives a wrong key that is not similar to those keywords, DISCAD will inform the user and request a valid input key as shown in Figure 4.2 below.



Input all information:

DISCAD will prompt the user to input numerical data through an input box. After receiving the data DISCAD will validate it, but will take only the numeric portion.

The full listing of the DISCAD coding is given in Appendix. A.

Chapter Five

Results and Discussion

5.1 Introduction:

This chapter is devoted to the analysis and discussion of the performance of the developed software. It describes the basic drawing model of the target object and modify the basic object by editing into a meaningful object. A number of different models have been drawn, and the step-by-step process of drawing an object, modifying or editing into a meaningful object and then saving it to a DXF file have been shown.

Section 5.2 Describes Line with Line intersection.

5.2 Line with Line intersection:

If DISCAD finds more than two lines then it looks for an intersection point between them. When DISCAD finds the intersection point then it divides the line into two lines. To divide each line DISCAD takes each line at a time and then compares their start point and end point with the intersection point. If the intersection point matches with the start point or end point DISCAD will not divide the line, as to do so it would divide the line and create two new lines.

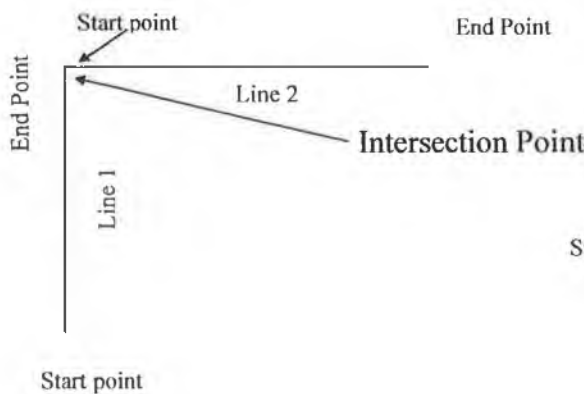


Figure 5.1(a)

This intersection will not generate a

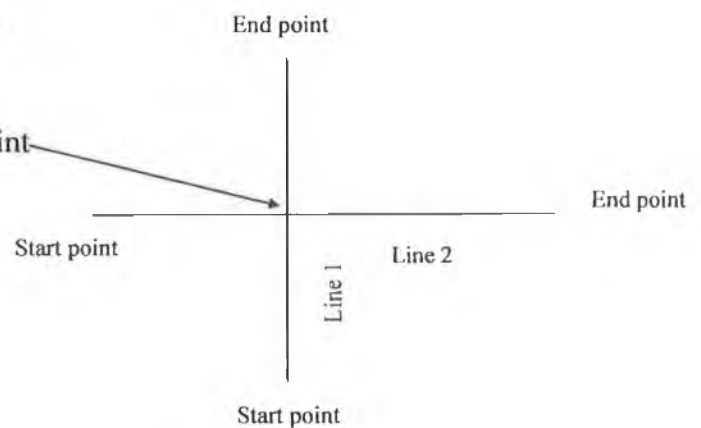


Figure 5.1(b)

This intersection will draw

new line.

4 more lines.

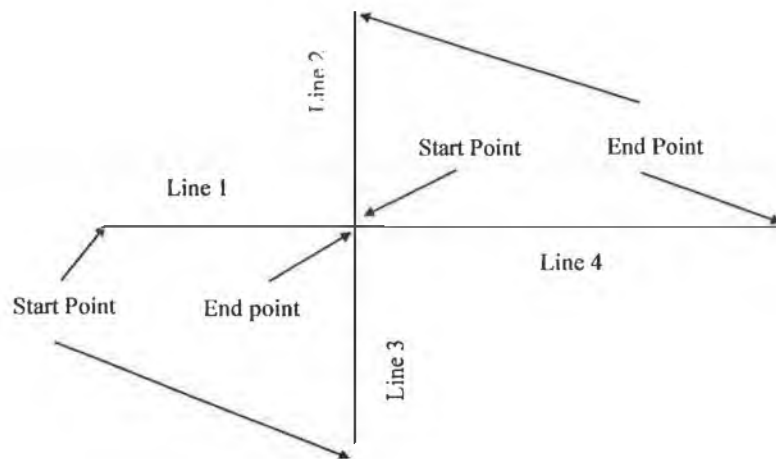
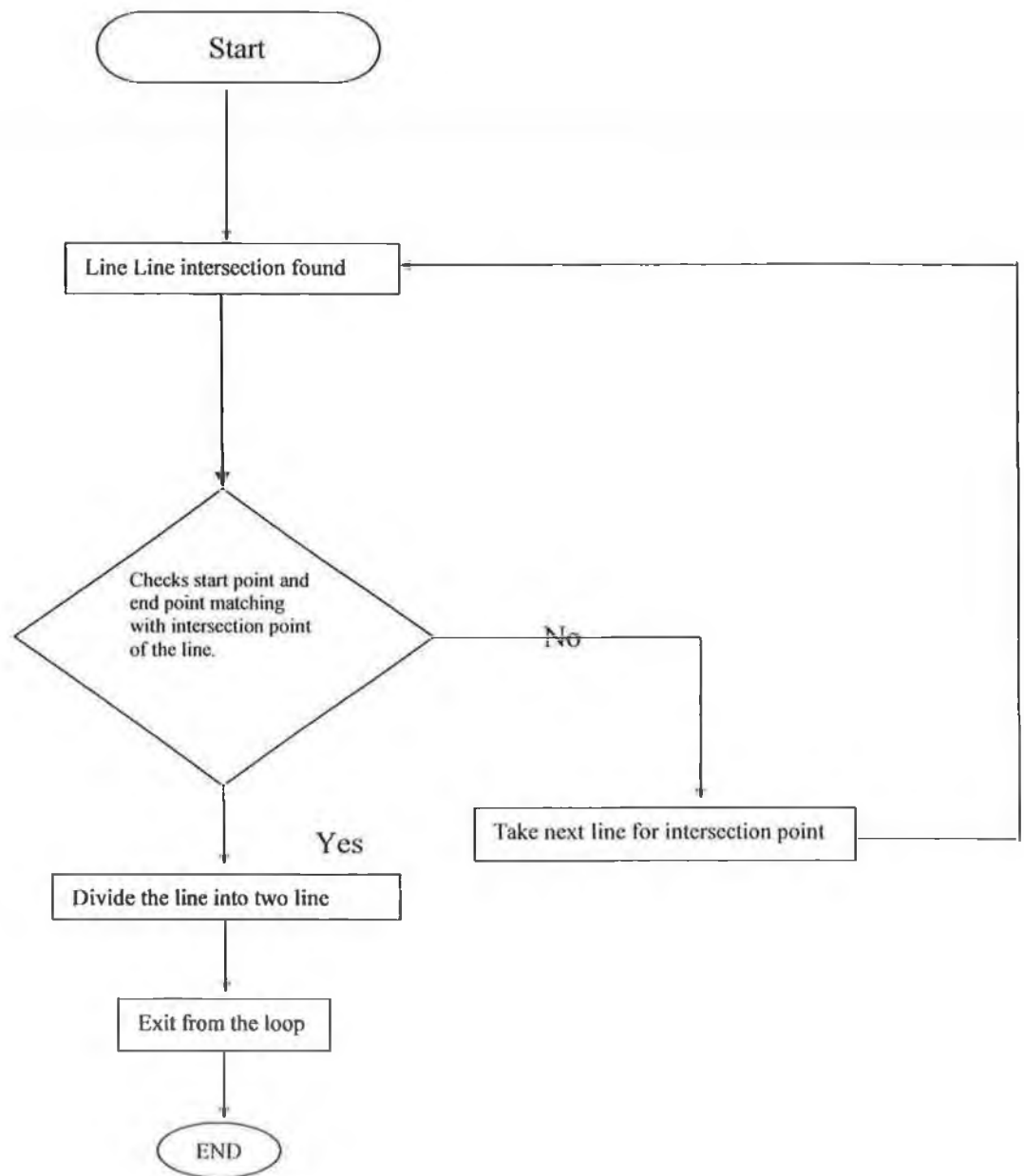


Figure 5.1(c) New generated 4 lines.

Block diagram of two line intersection



Block diagram 5.1(d)

5.3 Line with Circle Intersection:

When a line intersects a circle it always creates two intersection points. It is not necessary that the line will cross both sides of the circle. Sometimes an end point or start point may not be on the perimeter. Sometimes the line may create two intersection points, but sometimes the line will only create one intersection point. When a line creates two intersection points DISCAD will convert the circle into two Arcs and convert the line into three new lines. Figure 5.2(a) shows two arcs and three lines.

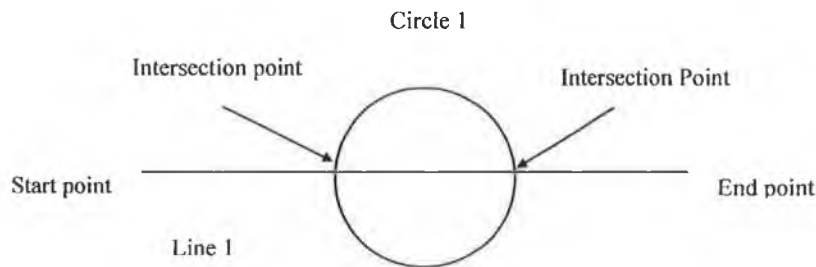


Figure 5.2(a) The line has created two intersection points

The Line has created two intersection points

After dividing the line and circle by the intersection point it will look like the diagram in Figure 5.2 (b).

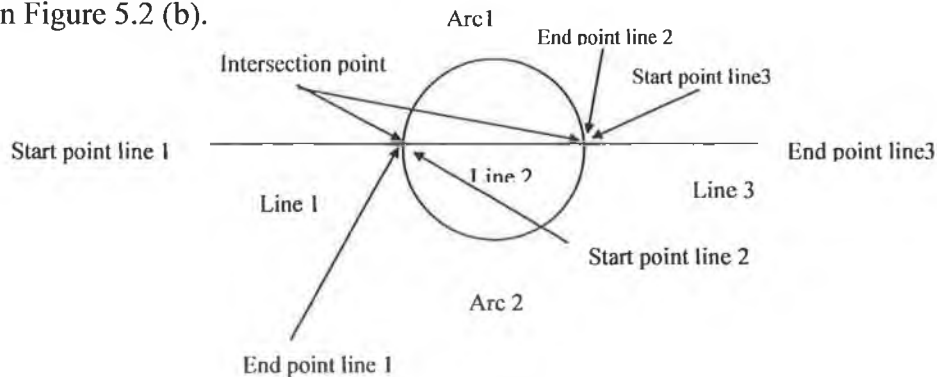


Figure 5.2(b) Single line produces three lines.

When the line creates only one intersection point the circle is converted into one Arc, whilst the line is converted into two lines. The diagram below in Figure 5.2 (c) displays this process.

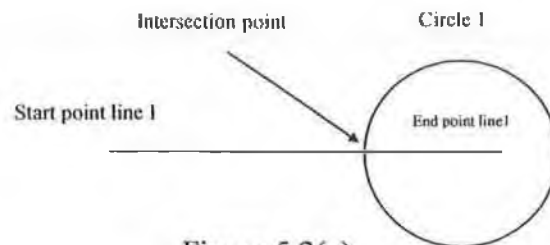


Figure 5.2(c)

Figure 5.2(c) Shows one line creating one intersection point

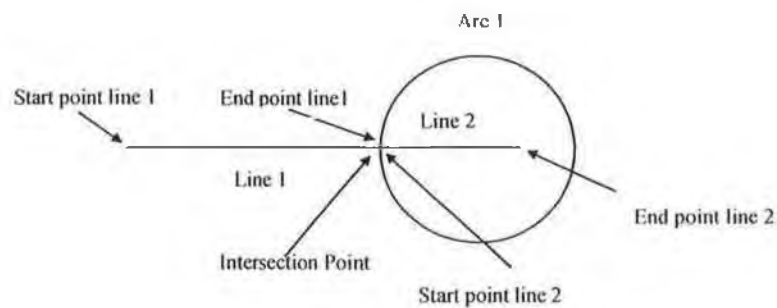


Figure 5.2(d) Single line converted into two lines.

This leads to the conversion of the line and circle into two lines and an arc as shown in Figure 5.2 (d).

5.4 Line with Arc intersection:

When a line intersects an arc it may create two intersection points. If so, the line can be divided into three segments and the arc can be divided into three arcs. The diagram in Figure 5.3 (a) below displays this.

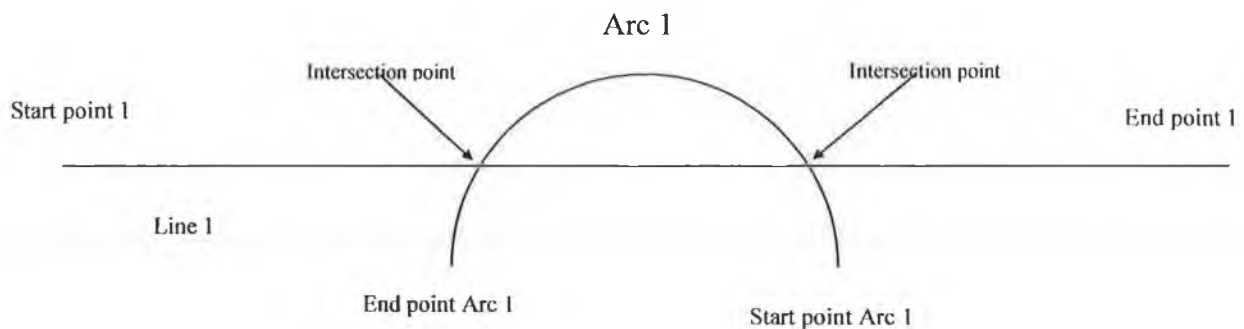


Figure 5.3(a) Shows line with arc intersection.

Figure 5.3 (a) shows a line that has created two-intersection points and an arc that has also been crossed at two intersections points. In this case, after converting the line, it will create three lines. Also, after converting the Arc1 it will create three more Arcs. Figure 5.3(b) shows these three new arcs and three new lines.

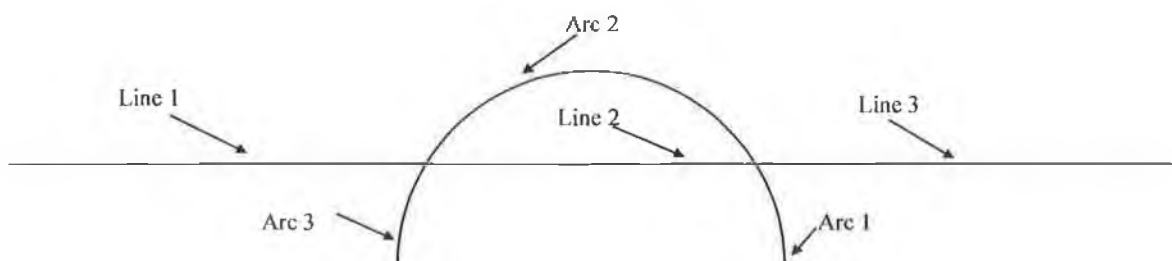


Figure 5.3(b) After converting line 1 and Arc 1 by their intersection point

When the line creates only one intersection point, the line will then create two more lines by dividing the arc in two arcs. The diagram in Figure 5.3 (c) and (d) describes this process.

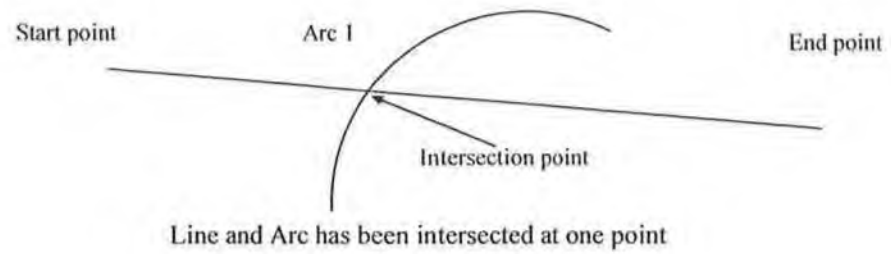


Figure 5.4(c) A line with one intersection point with an arc.

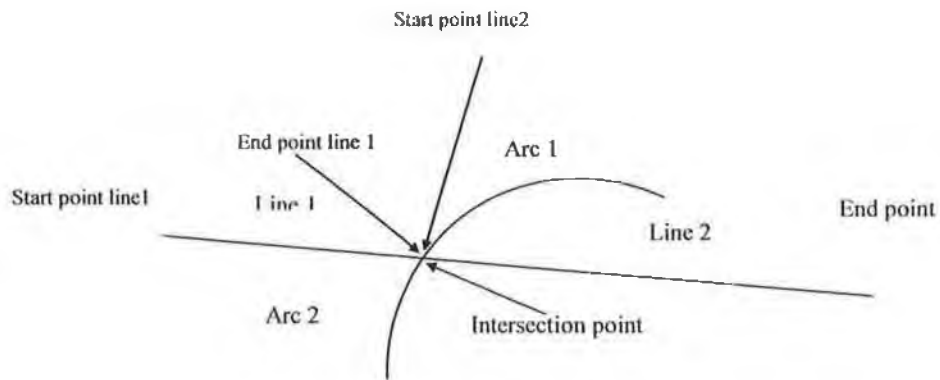


Figure 5.3 (d) Configurations after dividing the Arc and the line at one point

To save an arc as a DXF file it is necessary to know the start point and the end point of an Arc. One can determine the start point and end point by knowing their angle. To determine the start angle and the end angle of an Arc a formula has been used that is given by equation 5.1. The start point of an Arc can be determined by knowing the start angle and the end point and the end angle. One can calculate the smaller angle by comparing both angles. The diagram in Figure 5.3 (e) demonstrates this. In Auto CAD the arc is always drawn counter clockwise.

Figure 5.3(e) shows a normal arc because the start angle is smaller than the end angle.

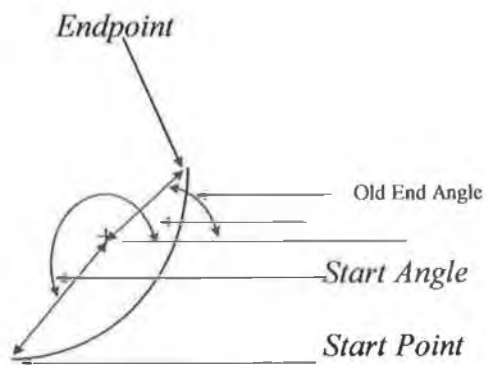
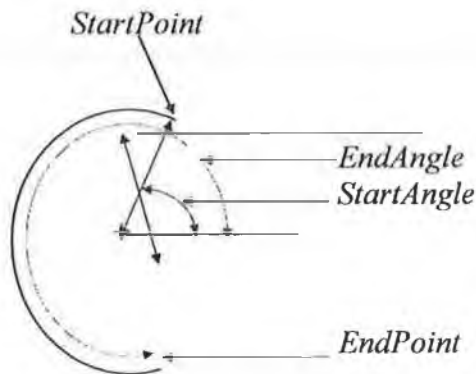


Figure 5.3(f) shows an unusual arc because the start angle is bigger than the end angle.

When an unusual arc is found the start angle and end angle are calculated by using equation (5-1). This equation always considers the start point as a zero degree angle and the end angle is changed accordingly.

$$Endangle = 360 - StartAngle + OldEndAngle \quad (5-1)$$

Figure 5.4 below is a block diagram showing a line and an Arc intersection options.

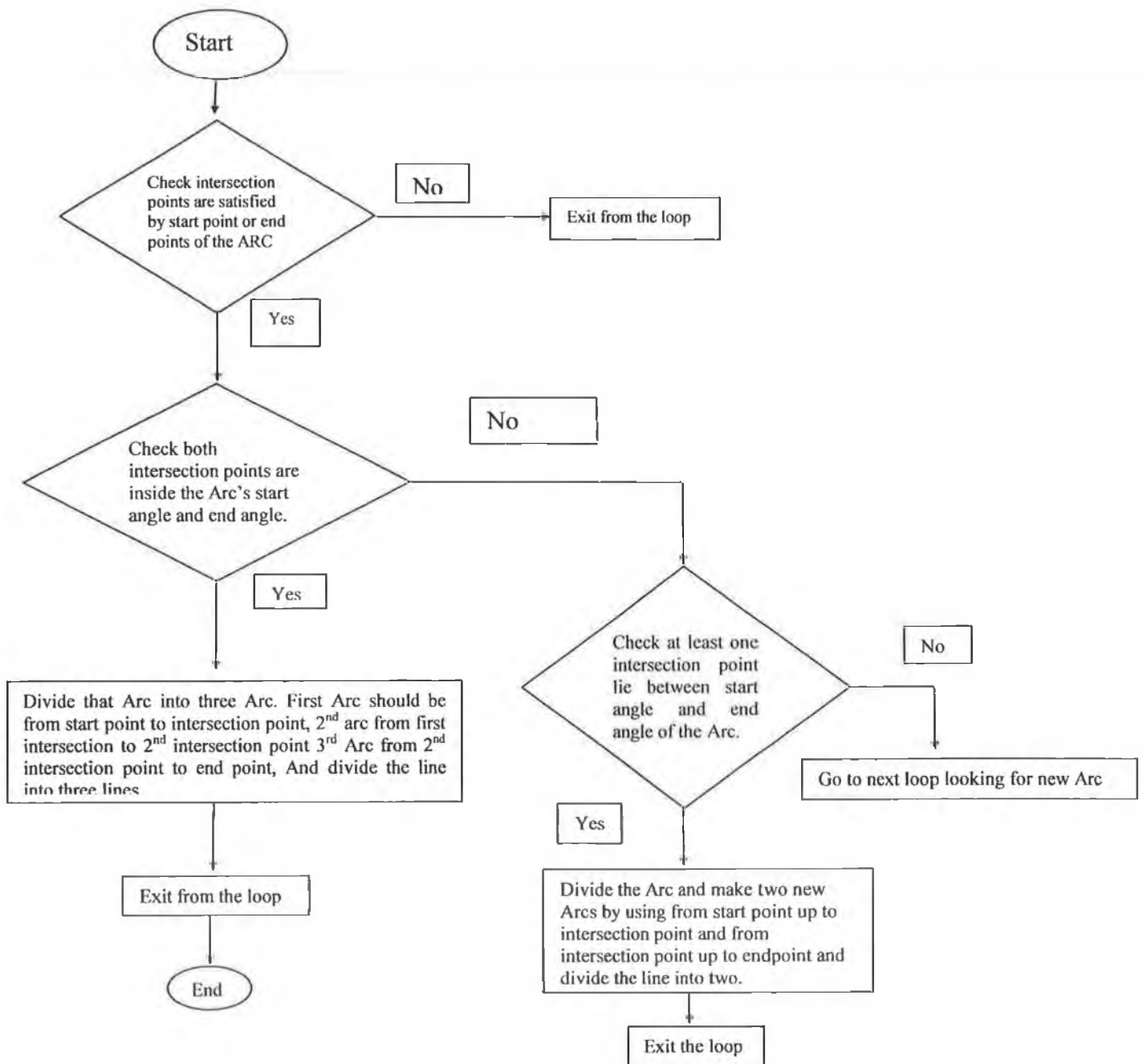


Figure 5.4 Various types of intersection of a line with an arc.

5.5 Circle with circle intersection.

When two circles intersect each other, each circle is converted into two arcs by their intersection point.

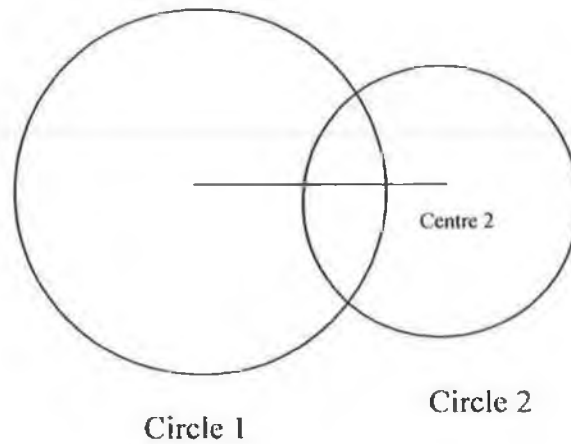


Figure 5.4(a) Two circles before being divided by their intersection points.

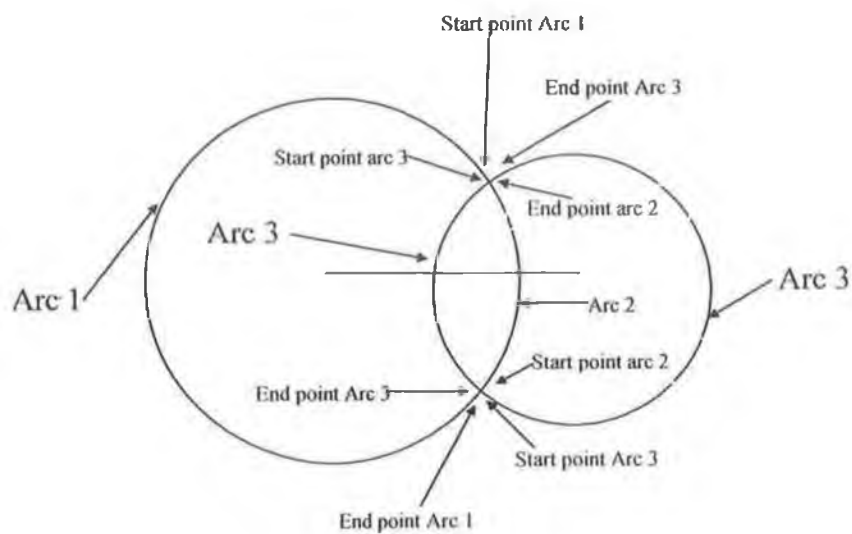


Figure 5.4(b) Two circles after being divided by their intersection points.

The block diagram in Figure 5.4 (c) is for various options in of two circles intersections.

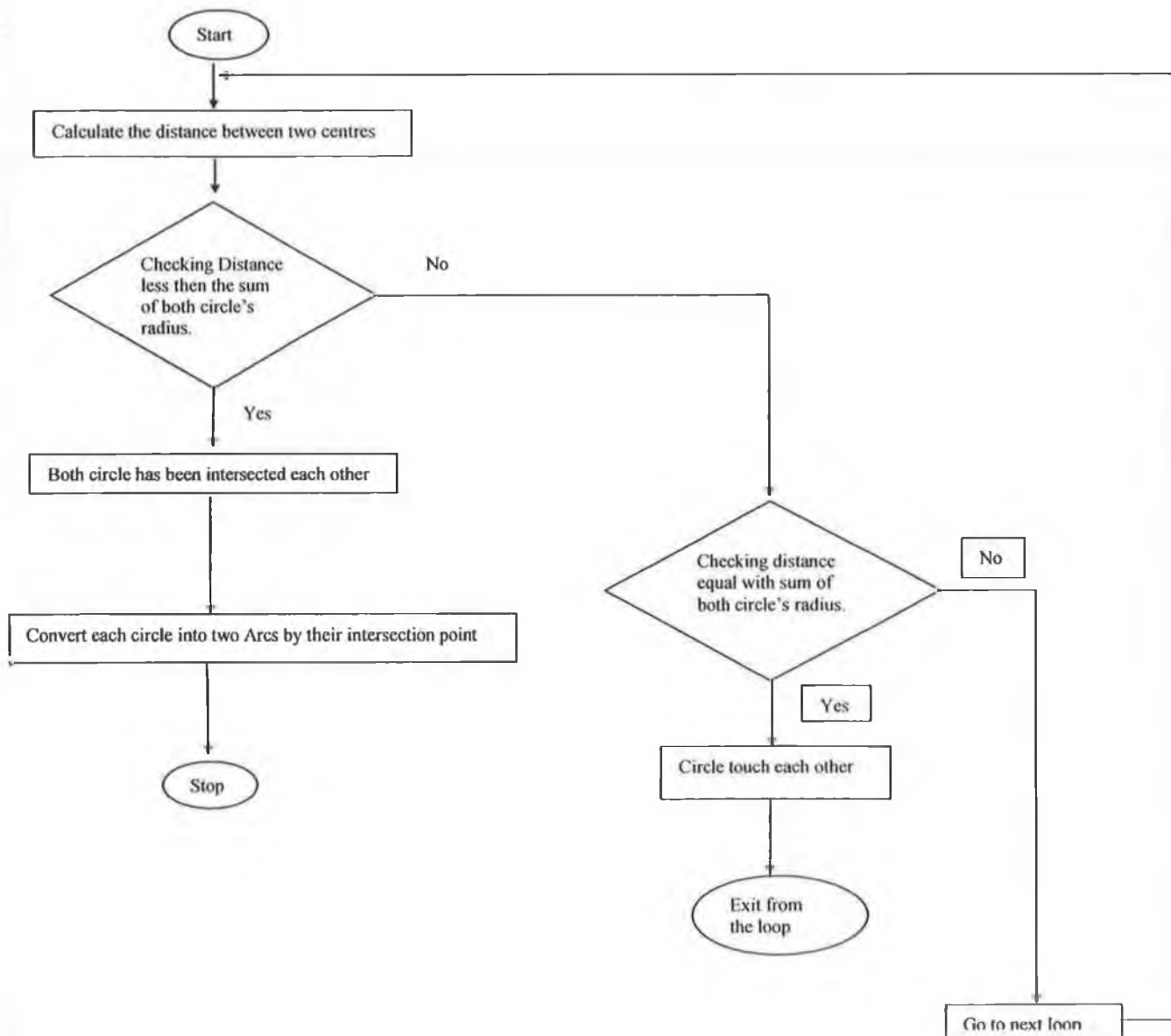


Figure 5.4 (c) Various option routes for two intersecting circles.

When a circle intersects an arc, and if two intersection points are found the arc will divide into three arcs and the circle will convert into two arcs. If the circle intersects the arc at one point the arc is converted into two arcs and the circle is converted into one Arc. It works like a circle circle intersection.

5.6 Determining the tangent point of a circle or Arc.

Pressing Ctrl+g from the keyboard DISCAD will prompt the user twice to input the object name. When DISCAD gets two Circles or two Arcs or one circle and one arc it will begin searching for a tangent point for both the circle and the arc. Figure 5.5 illustrates how it calculates the tangent point.

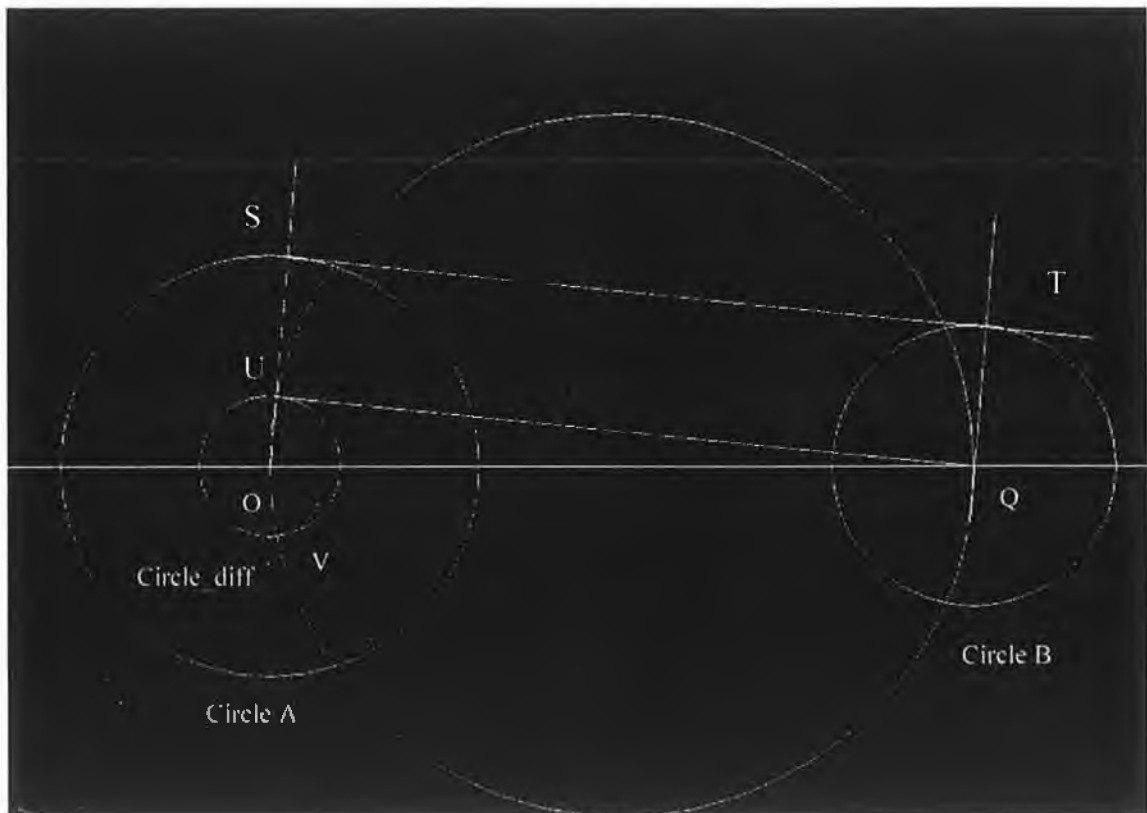


Figure 5.5

To determine the tangent point between two circles a procedure is applied. The first step is to identify the circle which has larger radius. For example in figure 5.5 circle A has a larger radius than circle B. The second step is to find the difference between the two radii. Let in figure 5.5 the difference in the radii $R_{diff} = R_a - R_b$, where R_a is the radius of circle A and R_b is the radius of circle B. The third step is to draw a circle whose radius is R_{diff} in the centre of larger circle. It is clearly demonstrated in the figure 5.5 that circle A is larger one and circle diff is the circle with radius R_{diff} . The fourth step is to draw a circle with diameter OQ and centre will be the midpoint of OQ. The new

circle will intersect circle diff at point's u and v. The fifth step is to draw a straight line through o and u which will intersect the circle A at point S. The sixth step is to draw a parallel line through the centre of Q (circle B). That will intersect at the circumference at point T. In the figure 5.5 S and T are the expected tangent points of those circles.

5.7 A Model of a Shaft connector:

A shaft connector has been drawn as an example, which will have a total of 11 objects at the final stage. The step-by-step procedure has been described below. Figure 5.6-(a) shows the model of a shaft connector that has been drawn using the software.

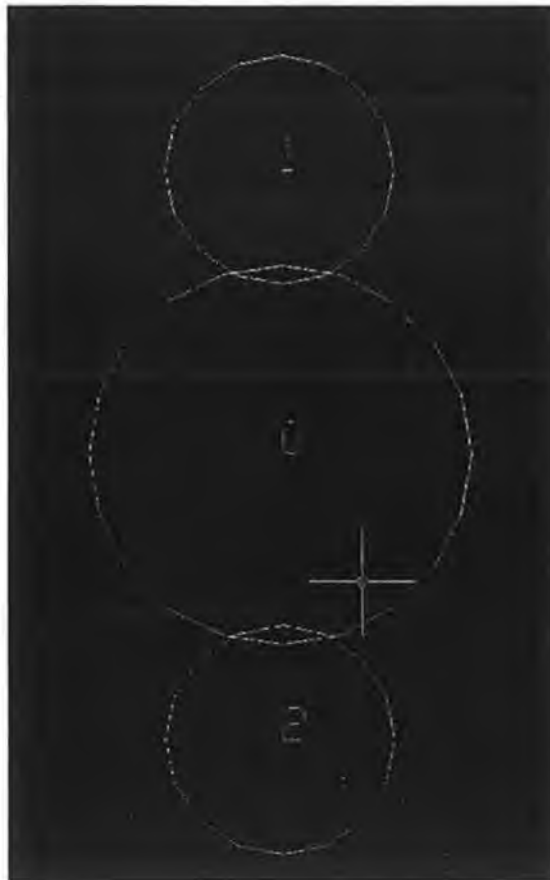


Figure 5.6-(a): Final drawn model of a Shaft connector.

To draw a shaft connector it needs 7 circles and 4 straight lines. Circles are drawn one by one. Circle-1 of centre coordinate (x_c, y_c) and radius r_c was drawn. Similarly Circle-2 to Circle-7 were drawn using corresponding centre point and radius. Next,

Line-1 of coordinates (x1, y1) and (x2, y2) were drawn. Similarly Line-2, Line-3 and Line-4 were drawn through corresponding coordinate pairs.

Figure 5.6-(b) shows the first step in drawing the shaft. The diagram in Figure 5.6 (b) shows what it looks like when three Circles have been drawn on the AutoCAD interface by using DISCAD.



(b)

Figure 5.6 (b): Three circles of the Shaft connector.

To draw a tangent line Ctrl + g is pressed. DISCAD pops up two input boxes for the object numbers. DISCAD then draws two tangent lines of two circles (circle 0 and circle 1) both of which have been selected by the user from (Figure 5.6 (b)). Figure 5.6(c) shows the two new tangents. Similarly DISCAD draws two more tangents by using object No: 2 and object No: 4 of Figure 5.6 (c). The new tangent is drawn as object 2 and object 3. Figure 5.6 (d) shows these.



(c)

Figure 5.6(c): Three circles with 2 tangent lines.



(d)

Figure 5.6(d): Three circles with 4 tangent lines.

The intersection point among circles and lines has been shown by pressing Ctrl + D in the interface of the DISCAD software. This helps the user to edit the drawing.



(e)

Figure 5.6(e): All objects named after intersection.

By pressing Ctrl+t DISCAD will pop up the Trim edit tool for trimming or deleting unexpected objects. After trimming the unexpected object from Figure 5.6(e) the new drawing shown in Figure 5.6(f) is taking the shape of a Shaft.



(f)

Figure 5.6(f): After trimming.

By pressing Ctrl+F1 figure.



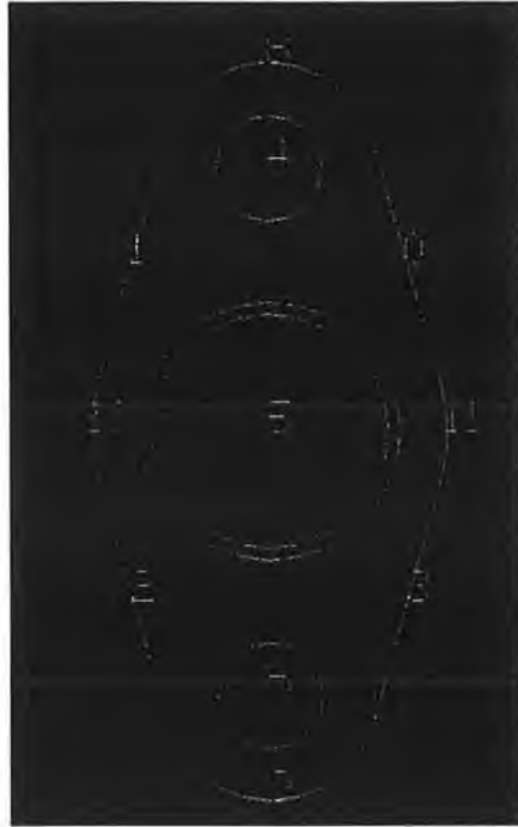
(g)

Figure 5.6(g): Pre shaft connector without an object name.



(h)

Figure 5.6(h): Shaft connector with two holes that has been drawn by two circles.
At the end of edit (j) Final Drawing of a shaft connector.



(i)

Figure 5.6(i): Shaft connector with three holes.



(j)

Figure 5.7(j): Final figure of a shaft connector without object number.

5.8 Model of a Joint link

Three circles and a straight line have been drawn. Press Ctrl+d to show the intersection point. Figure 5.7(a).



Figure 5.7(a)

Delete the unexpected object by pressing Ctrl + t. Figure 5.7(b) Shows after trimming.



Figure 5.7(b)

Shows after fillet at line and circle intersection point. Figure 5.8(c)

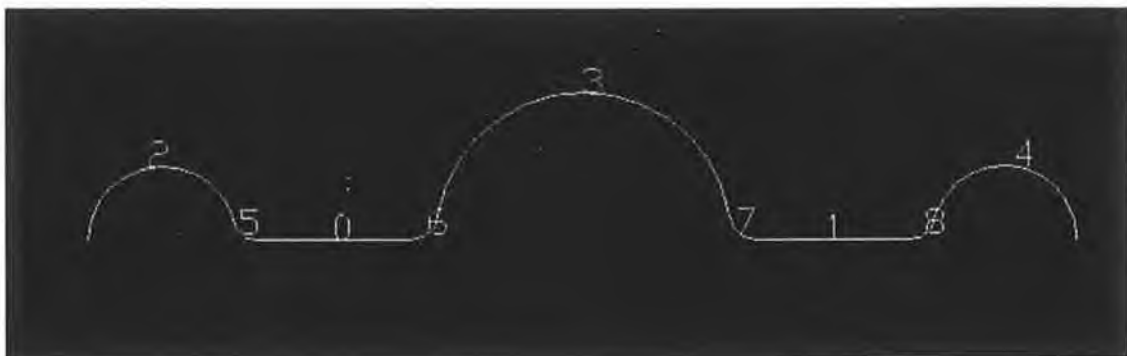


Figure 5.7(c)

Joining start and end point of a straight line with arc's start or end point.

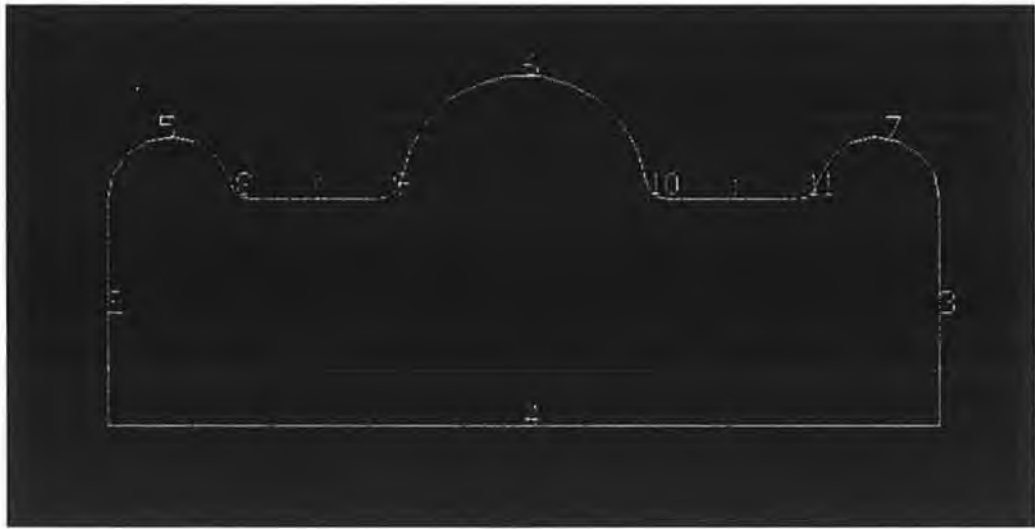


Figure 5.7(d)

Figure 5.7(e) with a small circle.

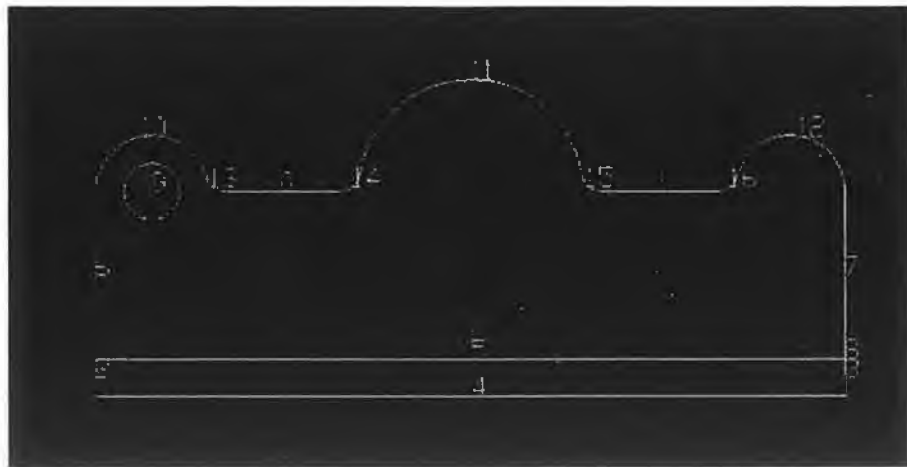


Figure 5.7(e)

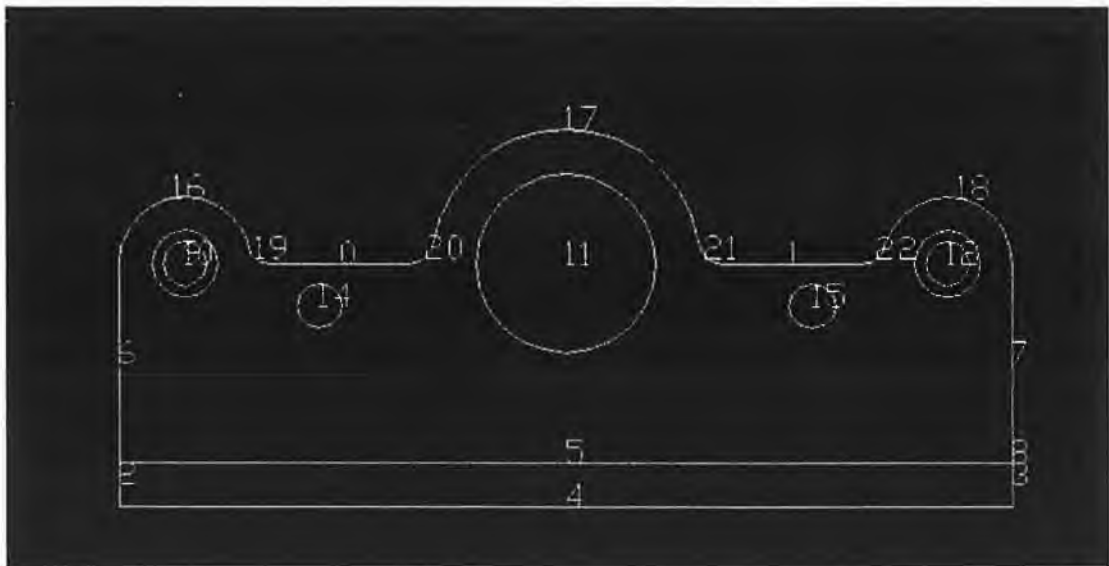


Figure 5.7(f): A Joint link with all object names.

Figure 5.7(f): A final joint Link. Press Ctrl+F1 or toggleswitch off.



Figure 5.7(g)

5.9 Model of a cover lock:

Draw two arcs, with the same centre. The start Angle is 90 degrees and the end angle is 240 degrees for the arc, the object number is 0. Similarly for the object number 1 the start angle is 90 and the end angle is 270.

Figure 5.8(a): Two Arcs.

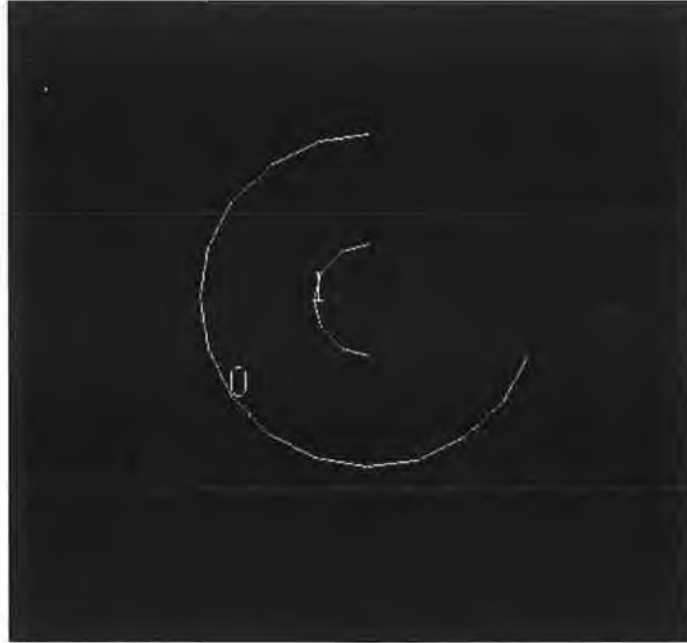


Figure 5.8(a)

Figure 5.8(b): Draw a straight line by pressing Ctrl+J between the end point of object no:0 and the end point of object No:1.



Figure 5.8(b)

Figure 5.8(c): Draw another Arc according to centre and start angle and end angle.



Figure 5.8(c)

Pressing Ctrl+g and DISCAD will draw a tangent line between object 3 and 1 and Figure 5.8(d) shows the tangent line.



Figure 5.8(d)

Figure 5.8(e) after unexpected object is deleted by pressing Ctrl+t.



Figure 5.8(e)

Draw two straight lines. Take the start coordinate from the object list box in the main DISCAD window.



Figure 5.8(f)

Figure 5.8(g) After trimming the unexpected straight line.



Figure 5.8(g)

Figure 5.8(g) After fillet



Figure 5.8(g)

Figure 5.8(h) Final picture of a cover lock without object name.



Figure 5.8(h)

5.10 Model of a Link:

To draw a link places two arcs according to measurement distance.



Figure 5.9(a)

Join the end point of the first object with the start point of the second object by pressing Ctrl+j.



Figure 5.9(b)

Similarly, join the first object start point with the second object end point.



Figure 5.9(c)



Figure 5.9(d)

Figure 5.9(e) After fillet.



Figure 5.9(e)

Chapter Six

Conclusions and Suggested Future Work

6.1 Conclusions:

In traditional Auto CAD one can select an object by using either the mouse or the keyboard. For some physically handicapped users it may be difficult to select/pinpoint objects by using the mouse because of their shaky hands. On the other hand, selecting /pinpointing object by keyboard is at times may also be tedious and too complicated for them.

By using the keyboard drawings can be generated and can be manipulated by using object name. The DISCAD software makes drawing easier for a person with shaky hands. This interactive software enables a disabled person or a person with learning difficulties to draw, shape and edit objects through manipulated object names.

DISCAD, a user friendly software, has been developed by the author as a part of the research project to create drawings of mechanical part which work inside the Auto CAD and interface with the same. A model of object can be drawn from basic drawings like line, circle, and arc. By pressing particular keys on the keyboard, the software can detect the intersection point of the drawing and it divides an object by the intersection point. It considers the divided object as an individual whole object. To edit and manipulate these objects each object is given a unique numerical identity, and thus the object is allowed to be selected by number keys.

This software can draw a line, circle and arc, and the drawing of objects is comprised of these elements. It can calculate the intersection point among them. It can also edit by using editing tools such as Fillet, Chamfer and Trim. It can select the objects by using object's name that is easier to execute than by using the mouse. For better visualisation DISCAD can zoom in and zoom out by pressing particular keys on the keyboard. The software developed by the author can write and read DXF files, which can also be read by Auto CAD.

The proposed system has some limitations. If the length of the object is short or if there are too many objects to be drawn which are very close to each other, DISCAD finds it difficult to identify the object by the corresponding object name in numbers.

6.2 Suggested Future Work:

- Each object and corresponding object's name should be identified by a unique colour.
- Visualisation of dimensions should be implemented.
- 3D modelling system may be implemented.
- Speech recognition can be implemented.

References

1. J. Springer, C. Siebes, "International Journal of Industrial Ergonomics", Volume 17, Issue 2, February 1996, Pages 135-152.
2. K. Kawamura, R. T. Pack, M. Bishay and M. Iskarous, "Robotics and Autonomous Systems", Volume 18, Issues 1-2, July 1996, Pages 109-116.
3. J. M. Noyes, R. Haigh and A. F. Starr, "Applied Ergonomics", Volume 20, Issue 4, December 1989, Pages 293-298.
4. J. D. Nicoud, "Education and Computing", Volume 2, Issues 1-2, 1986, Pages 107-111.
5. C. Miralles, J. P. García-Sabater, Carlos Andrés and Manuel Cardos, "International Journal of Production Economics", Article in Press.
6. J. J. Chen and M. Ko, "International Journal of Industrial Ergonomics", Volume 13, Issue 4, June 1994, Pages 317-335.
7. M. P. Srinivasan, C. R. Venugopal and N. Kaulgud, "Journal of Microcomputer Applications", Volume 13, Issue 3, July 1990, Pages 261-272.
8. J. E. Oakey, "Journal of Microcomputer Applications", Volume 16, Issue 3, July 1993, Pages 271-276.
9. Dr Coldwell, "Journal of Microcomputer Applications", Volume 18, Issue 4, October 1995, Pages 305-311.

10. M. S. Hawley, P.A. Cudd, J. H. Wells, A. J. Wilson and P. L. Judd, "Journal of Biomedical Engineering", Volume 14, Issue 3, May 1992, Pages 193-198.
11. K. Seeland and S. Nicolè, "Urban Forestry & Urban Greening", Volume , Issue 1, 13 June 2006, Pages 29-34.
12. K. Park, Y. S. Kim, C. S. Kim and H. J. Park, "Journal of Materials Processing Technology ", Volumes 187-188, 12 June 2007, Pages 609-613.
13. F. Nagata, Y. Kusumoto, Y. Fujimoto and K. Watanabe, "Robotics and Computer-Integrated Manufacturing", Volume 23, Issue 4, August 2007, Pages 371-379.
14. K. Chen, X. Feng, Q. Lu, "Intelligent location-dimensioning of cylindrical surfaces in mechanical parts", Computer-Aided Design, Volume 34 2002, Pages 185-194.
15. K. Z. Chen, X. Feng, L. Ding, "Intelligent approaches for generating assembly drawings from 3-D computer models of mechanical products", Computer-Aided Design, Volume 34, 2002 Pages 347-355.
16. B.S.Prabhu, S.S.Pande, "Intelligent interpretation of CADD drawings", Computer & Graphics, Volume 23, 1999, Pages 25-44.
17. D. Varró, G. Varró and A. Pataricza, "Designing the automatic transformation of visual languages Science of Computer Programming", Volume 44, Issue 2, August 2002, Pages 205-227.
18. M. Hauser, R. J. Scherer, "A cognitive architecture to support structural design tasks", Computer and Structures, Volume 67 1998 Pages 339-346.

19. D. Su, M. Wakelam, "Intelligent hybrid system for integration in design and manufacture", *Journal of materials processing technology*, Volume 76 1998 Pages 23-28.
20. R. Taraban, "Drawing learners attention to syntactic context aids gender-like category induction", *Journal of Memory and Language*, Volume 51, 2004, Pages 202-216.
21. G. V. Bemden, P. Dufour and C. Marco, "French lip-reading and cued-speech training by interactive video", *Journal of Microcomputer Applications*, Volume 13, Issue 2, April 1990, Pages 193-200.
22. G. Celano, S. Fichera, L. Fratini, F. Micari, "The application of AI techniques in the optimal design of multi-pass cold drawing processes", *Journal of Materials Processing Technology*, Volume 113, 2001, Pages 680-685.
23. A. Nassehi, S. T. Newman, R. D. Allen, "The application of multi-agent systems for STEP-NC computer aided process planning of prismatic components", *International Journal of Machine Tools & Manufacture*, Volume 46, 2006, Pages 559-574.
24. R. Schleiffer, "An intelligent agent model", *European journal of Operational Research*, Volume 166, 2005, Pages 666-693.
25. X. P. Yan, C. H. Zhao, Z. Y. Lu, X. C. Zhou, H. L. Xiao, "A study of information technology used in oil monitoring", *Tribology International*, Volume 38, 2005, Pages 879-886.
26. F. yang, M. Wang, R. Shen, P. Han, "Community-organizing agent: An Artificial intelligent system for building learning communities among large numbers of learners", *Computers & Education*, 2005, Article in press.

27. M. Sugisaka, A. Loukianov, F. Xiongfeng, T. Kubik and K. B. Kubik, "Development of an artificial brain for lifeRobot", *Applied Mathematics and Computation*, Volume 164, 2005, pages 507-521.
28. E. Leon, G. Clarke, V. Callaghan, F. Sepulveda, "Real time detection of emotional changes for inhabited environments", *Computer & Graphics*, Volume 28, 2004, Pages 635-642.
29. M. N. Jadid and M. M. Idrees, "Automation in Construction", Volume 16, Issue 6, September 2007, Pages 797-805.
30. T. Yamaguchi, Y. Kawase, T. Nishimura and H. Naito "Journal of Materials Processing Technology", Volume 161, Issues 1-2, 10 April 2005, Pages 311-314.
31. S. Kirby, "Natural language from artificial life", *Artificial Life* Volume 8, Issue 2, 2002, Pages 185-215.

Appendix - A

Source Code

```

Public acadApp As Object
Public acadDoc As Object
Public mspace As Object
Dim Mopen As Object
Dim DOC As Object
Dim boxObj As Object
Dim cylinderObj As Object
'Dim Circleobj() As Object
'Dim LineObj As Object
Dim Line_Array() As Double
Dim Line_Array1() As Double
Dim block_Array() As Double
Dim Cylinder_Array() As Double
Dim Circle_Array() As Double
Dim Circle_Array1() As Double
'Dim Circle_Array() As Double
Dim Arc_Array() As Double
Dim Arc_Array1() As Double
Dim Arcobj() As Object
Dim ssetObj_ARC As Object
Dim ssObjs_ARC() As Object
Dim stvalue As String
Private m_blnCloseEnabled As Boolean
Dim LineEditArray() As Double
Dim CirDy() As Double
Dim LinDy() As Double
Dim ArcDy() As Double
Dim FindLineOb As Integer
'for selecat circle object
'circle
Dim ssetObj_Cir As Object
Dim Circleobj() As Object
Dim ssobjs() As Object
'line
Dim ssetobj_Line As Object
Dim Lineobj_1() As Object
Dim ssobjs_line() As Object
'Text or Numbering *****
Dim SsetObj_text As Object
Dim TextObj() As Object
Dim Ssobjs_text() As Object

''

Dim Total_Circle As Integer
Dim Total_Line As Integer
Dim Total_ARC As Integer
'Dim Total_Object As Integer
'kjhj

Dim Check_Block As Boolean, check_circle As Boolean
Dim check_cylinder As Boolean
Dim check_line As Boolean, check_arc As Boolean

Dim textString As String
    Dim insertionPoint(0 To 2) As Double
    Dim height_C As Double
    Dim Object_Counter As Integer
    Dim LineObj_counter As Integer
    Dim CircleObj_counter As Integer
    Dim Total_Object As Integer
    Dim ARCObj_counter As Integer
Dim Repeat_Object_Number As Boolean
Dim stop_Repeat_objectnumber As Boolean, toggle_on As Boolean
Dim InterSec As Boolean
Dim Edit_Circle As Boolean
Dim SellectionSet_TotalCircle As Integer, Selectionset_text_Access As Boolean, TEST_SELECTIONSET_Arc_Access As
Boolean, TEST_SELECTIONSET_Line_Access As Boolean

```

```
Dim TEST_SELECTIONSET_circle_Access As Boolean
Dim AllobjectName() As String
Dim Line1(4) As Single, Line2(4) As Single, Circle1(3) As Single, Circle2(3) As Single, Arc1(7) As Single, Arc2(7) As Single,
Position(1) As String
Dim radiouTrueorFalse As Boolean, RadiusFillet As Single
Dim int_X1 As Single, int_Y1 As Single, int_x2 As Single, int_Y2 As Single, oneintersec As Boolean, twointersec As Boolean,
CircLe_Counter As Integer, Comp As Single
Dim Compare As Single
Dim scaleFactor As Integer
```

```
' This example creates a new viewport and makes it active.
' Then it splits the viewport into 4 windows.
' It then takes finds the lower left corner of each of the
' windows.
```

```
-----
Set acadApp = GetObject(, "AutoCAD.Application")
Set acadDoc = acadApp.ActiveDocument
```

```
Dim newViewport As Object
```

```
' Create a new viewport and make it active
Set newViewport = acadDoc.Viewports.Add("TESTVIEWPORT")
acadDoc.ActiveViewport = newViewport
```

```
' Split the viewport in 4 windows
newViewport.Split acViewport4
```

```
' Make the newly split viewport active
acadDoc.ActiveViewport = newViewport
```

```
' Iterate through the viewports. For each viewport,
' make that viewport active and display the coordinates
' of the lower left corner.
```

```
Dim entry As Object
Dim lowerLeft As Variant
For Each entry In acadDoc.Viewports
    entry.GridOn = True
    acadDoc.ActiveViewport = entry
    lowerLeft = entry.LowerLeftCorner
    MsgBox "The lower left corner of this viewport is " & lowerLeft(0) & ", " & lowerLeft(1), , "LowerLeftCorner Example"
    entry.GridOn = False
Next
```

```
End Sub
```

```
Public Function GetParamValue_Delete(ParamStr As String)
Dim char As String, TempStr As String, valStr As String
Dim NoDot As Boolean
NoDot = True
TempStr = Trim$(ParamStr)
valStr = ""
While (Not (Len(TempStr) = 0))
    char = Left$(TempStr, 1)
    If (StrComp(char, "-") = 0 And Len(valStr) = 0) Then
        valStr = valStr & char
    ElseIf (StrComp(char, ".") = 0 And NoDot) Then
        valStr = valStr & char
        NoDot = False
    ElseIf (StrComp(char, "0") = 0 Or StrComp(char, "1") = 0 Or
        StrComp(char, "2") = 0 Or StrComp(char, "3") = 0 Or
        StrComp(char, "4") = 0 Or StrComp(char, "5") = 0 Or
```

```

        StrComp(char, "6") = 0 Or StrComp(char, "7") = 0 Or _
        StrComp(char, "8") = 0 Or StrComp(char, "9") = 0) Then
            valStr = valStr & char
        End If
        TempStr = Right(TempStr, Len(TempStr) - 1)
    Wend
    If NoDot = False Then
        GetParamValue_Delete = False
        value_1 = 1
    ElseIf (IsNumeric(valStr)) Then
        ParamStr = Val(valStr)
        GetParamValue_Delete = True
    Else
        GetParamValue_Delete = False
        value_1 = 1
    End If
End Function

Public Function GetParamValue(ParamStr As String)
    Dim char As String, TempStr As String, valStr As String
    Dim NoDot As Boolean
    NoDot = True
    TempStr = Trim$(ParamStr)
    valStr = ""
    While (Not (Len(TempStr) = 0))
        char = Left$(TempStr, 1)
        If (StrComp(char, "-") = 0 And Len(valStr) = 0) Then
            valStr = valStr & char
        ElseIf (StrComp(char, ".") = 0 And NoDot) Then
            valStr = valStr & char
            NoDot = False
        ElseIf (StrComp(char, "0") = 0 Or StrComp(char, "1") = 0 Or _
            StrComp(char, "2") = 0 Or StrComp(char, "3") = 0 Or _
            StrComp(char, "4") = 0 Or StrComp(char, "5") = 0 Or _
            StrComp(char, "6") = 0 Or StrComp(char, "7") = 0 Or _
            StrComp(char, "8") = 0 Or StrComp(char, "9") = 0) Then
            valStr = valStr & char
        End If
        TempStr = Right(TempStr, Len(TempStr) - 1)
    Wend
    If (IsNumeric(valStr)) Then
        ParamStr = Val(valStr)
        GetParamValue_Delete = True
    Else
        GetParamValue_Delete = False
        value_1 = 1
    End If
End Function

Private Function IsCommandOk(ByVal ComStr As String) As Boolean
    Dim KeyWords(8) As String, TempStr As String, index As Integer, ComCount As Integer
    KeyWords(0) = "line"
    KeyWords(1) = "circle"
    KeyWords(2) = "rectangle"
    KeyWords(3) = "triangle"
    KeyWords(4) = "arc"
    KeyWords(5) = "ellipse"
    KeyWords(6) = "cylinder"
    KeyWords(7) = "line"
    ComCount = 0
    For i = 0 To 6
        index = 1
        TempStr = ComStr
        While (Not (index = 0))
            index = InStr(1, TempStr, KeyWords(i))
            If (Not (index = 0)) Then
                ComCount = ComCount + 1
                TempStr = Trim$(Right$(LCase$(TempStr), Len(TempStr) - (index + Len(KeyWords(i)) - 1)))
            End If
        Wend
    Next i
    If (ComCount > 1) Then
        IsCommandOk = False
    Else
        IsCommandOk = True
    End If
End Function

```

```

Private Sub Command10_Click()
Dim storenumber As Integer, giveObjectnumber As String
Call LookforObjectNumber
giveObjectnumber = InputBox("Enter the number of the object")
If AllobjectName(giveObjectnumber) Like "line*" Then
    storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 5))
    Call LineSearch(storenumber)
ElseIf AllobjectName(giveObjectnumber) Like "circle*" Then
    storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 7))
ElseIf AllobjectName(giveObjectnumber) Like "arc*" Then
    storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 4))
End If

End Sub

Private Sub Command11_Click()
Call SaveAll
End Sub

Function SaveAll()
Dim LineX1 As Single, LineY1 As Single, LineX2 As Single, LineY2 As Single
Dim CenterX As Single, CenterY As Single, Radius As Single
Dim FirstAngleX As Single, FirstAngleY As Single, SAngle As Single, SecondAngleX As Single, SecondAngleY As Single,
EAngle As Single
Dim FileName As String

    Cdlg1.Filter = "DXF files(*.dxf)|*.dxf|"
    Cdlg1.ShowSave
    On Error GoTo SaveProblems
    FileName = Cdlg1.FileName

    Open FileName For Output As #1

DXFBeginHeader

'Call DXFLimits(0, 0, 100, 55)

DXFEndHeader
DXFBeginTables
Call DXFBeginLayerTable(1)
Call DXFLayer("Layer1", 1, "")

DXFEndTable

DXFEndTables
DXFBeginEntities
If Total_Line > 0 Then
    For i = 0 To UBound(Line_Array) Step 4
        LineX1 = Line_Array(i)
        LineY1 = Line_Array(i + 1)
        LineX2 = Line_Array(i + 2)
        LineY2 = Line_Array(i + 3)
        Call DXFLine(LineX1, LineY1, 0, LineX2, LineY2, 0, "Layer1", 0, "", 1)
    Next
End If
If Total_Circle > 0 Then
    For i = 0 To UBound(Circle_Array) Step 3
        CenterX = Circle_Array(i)
        CenterY = Circle_Array(i + 1)
        Radius = Circle_Array(i + 2)
        Call DXFCircle(CenterX, CenterY, 0, Radius, "Layer1", 0, "", 1)
    Next i
End If
If Total_ARC > 0 Then
    For i = 0 To UBound(Arc_Array) Step 7
        CenterX = Arc_Array(i)
        CenterY = Arc_Array(i + 1)
        Radius = Arc_Array(i + 2)
        FirstAngleX = Arc_Array(i + 3) - Arc_Array(i)
        FirstAngleY = Arc_Array(i + 4) - Arc_Array(i + 1)
        SAngle = Val(FormatNumber(atan2(FirstAngleX, FirstAngleY), 3))

        SecondAngleX = Arc_Array(i + 5) - Arc_Array(i)
        SecondAngleY = Arc_Array(i + 6) - Arc_Array(i + 1)
        EAngle = Val(FormatNumber(atan2(SecondAngleX, SecondAngleY), 3))
    Next i
End If

```

```
        Call DXFArc(CenterX, CenterY, 0, Radius, SAngle, EAngle, "Layer1", 0, "", 1)
    Next i
End If

DXFEndEntities
DXFEndFile
Close #1
MsgBox ("It is done.")

Exit Function

If Cdg1.FilterIndex = 1 Then
    Cdg1.DefaultExt = "DXF"
    rtbDisplay.SaveFile Cdg1, DXF
Else
    Cdg1.DefaultExt = "DXF"
End If
Exit Function
SaveProblems:
    MsgBox "Can't save the file, try again.", vbCritical

Close #1

End Function

Private Sub Command2_Click()
    Call Main
End Sub
Function Main()
    Dim keyword_1(5), paramStr_1 As String, Tokens() As String, ParamCount As Integer, ParamValue As Double
    Dim count_5 As Boolean, paramvalue_1 As Boolean, toKen_1() As String, For_cen As Integer, Paramstr_2 As String,
    Paramstr_3 As String
    Dim Find_p As Integer
    Dim Lineobj As Object
    Dim Out As String
    Dim oObjectCount As Boolean
    Dim ArcStartX As Single, ArcStartY As Single, ArcEndX As Single, ArcEndY As Single
    Dim Edit_ARC As Boolean
    Dim Edit_line As Boolean
    Edit_line = False
    Edit_ARC = False
    keyword_1(0) = "block"
    keyword_1(1) = "cylinder"
    keyword_1(2) = "line"
    keyword_1(3) = "circle"
    keyword_1(4) = "arc"
    Dim count_2 As Integer, countOFcenter As Integer

    Check_Block = False
    check_circle = False
    check_arc = False
    check_cylinder = False
    check_line = False
    countOFcenter = 0
    count_2 = 0
    count_5 = 0
    Dim permission As Boolean
    Dim CircleVariable(2) As Single, i_circle As Integer, ArcVariable(4) As Single, i_Arc As Integer, LineVariable(3) As Single,
    l_line As Integer
    permission = True

    If toggle_on = True Then
        MsgBox "you have to press Toggle Switch"
        Exit Function
    End If
    toggle_on = False
    If Trim(LCase$(Text1.Text)) = "open" Then
        block_list.Clear
    End If
```

```

If block_list.ListCount = 0 Then
    oBjectCount = True
    ComStr = Split(Text1.Text)
    If Trim(LCase$(Text1.Text)) = "open" Then
        block_list.Text = " "
        ObCount.Text = " "
        Cdg1.FileName = ""
        Cdg1.Filter = "DXF Files (*.dxf)|*.dxf"
        Cdg1.ShowOpen
        dxffile = Cdg1.FileName

        Out = ReadDXF(dxffile, "ENTITIES", "LINE", ",10,20,11,21,")

        Data = Split(Out, ",")
        permission = True
        For ii = 1 To UBound(Data) Step 2
            block_list.AddItem "line"
            block_list.AddItem (Data(ii))
            Debug.Print Data(ii)

        check_line = True
        Next ii
    Else
        For j = 0 To 4
            For i = LBound(ComStr) To UBound(ComStr)
                endcount = 0
                indx = InStr(1, LCase$(ComStr(i)), keyword_1(j))

                " BLOCK
                If (keyword_1(j) = "block") And (indx > 0) Then
                    Check_Block = True

                For block_X = 1 To 100
                    stvalue = InputBox("Enter starting Corner of the block of X")

                    If GetParamValue(stvalue) Then
                        block_list.AddItem "block"
                        block_list.AddItem (stvalue)

                    Exit For
                Else
                    w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
                    If w_do = 2 Then
                        block_list.Clear
                        Exit Function
                    End If

                End If

                Next block_X

                For block_Y = 1 To 100
                    stvalue = InputBox("Enter the starting Corner of the Block of Y")
                    If GetParamValue(stvalue) Then
                        block_list.AddItem "block"
                        block_list.AddItem (stvalue)

                    Exit For
                Else
                    w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
                    If w_do = 2 Then
                        block_list.Clear
                        Exit Function
                    End If

                End If

                Next block_Y

```

```

For block_L = 1 To 100
    stvalue = InputBox("Enter the Length")
    If GetParamValue(stvalue) Then
        block_list.AddItem "block"
        block_list.AddItem (stvalue)

        Exit For
    Else
        w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
        If w_do = 2 Then
            block_list.Clear
            Exit Function
        End If
    End If
Next block_L
For block_W = 1 To 100
    stvalue = InputBox("Enter the Right Width")

    If GetParamValue(stvalue) Then
        block_list.AddItem "block"
        block_list.AddItem (stvalue)

        Exit For
    Else
        w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
        If w_do = 2 Then
            block_list.Clear
            Exit Function
        End If
    End If
Next block_W
For block_H = 1 To 100
    stvalue = InputBox("Enter the Right Height")
    If GetParamValue(stvalue) Then
        block_list.AddItem "block"
        block_list.AddItem (stvalue)
        Exit For
    Else
        w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
        If w_do = 2 Then
            block_list.Clear
            Exit Function
        End If
    End If
Next block_H
''' CYLINDER
ElseIf (keyword_L(j) = "cylinder") And (indx > 0) Then
    check_cylinder = True
    For cylin_x = 1 To 100
        stvalue = InputBox("Enter the center X of the cylinder")
        If stvalue = "" Then
            block_list.Clear
            Exit Function
        End If
        If GetParamValue(stvalue) Then
            block_list.AddItem "cylinder"
            block_list.AddItem (stvalue)

            Exit For
        Else
            w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
            If w_do = 2 Then
                block_list.Clear
                Exit Function
            End If
        End If
    Next cylin_x
    For cylin_Y = 1 To 100
        stvalue = InputBox("Enter the center Y of the cylinder")
        If GetParamValue(stvalue) Then
            block_list.AddItem "cylinder"
            block_list.AddItem (stvalue)

```

```

Exit For
Else
    w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
    If w_do = 2 Then
        block_list.Clear
        Exit Function
    End If

End If
Next cylin_Y

For cylin_D = 1 To 100
    stvalue = InputBox("Enter the diameter")
    If GetParamValue(stvalue) Then
        block_list.AddItem "cylinder"
        block_list.AddItem (stvalue)

        Exit For
    Else
        w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
        If w_do = 2 Then
            block_list.Clear
            Exit Function
        End If
    End If
Next cylin_D
For cylin_H = 1 To 100
    stvalue = InputBox("Enter the Height of the cylinder")
    If GetParamValue(stvalue) Then
        block_list.AddItem "cylinder"
        block_list.AddItem (stvalue)
        Exit For
    Else
        w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
        If w_do = 2 Then
            block_list.Clear
            Exit Function
        End If
    End If
Next cylin_H

''' CIRCLE
ElseIf (keyword_1(j) = "circle") And (indx > 0) Then
    check_circle = True
    For circle_x = 1 To 100
        stvalue = InputBox("Enter X coordinate", "Centre X of the Circle", , 0, 0)

        If GetParamValue(stvalue) Then
            CircleVariable(0) = Val(stvalue)
            Exit For
        Else
            w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
            If w_do = 2 Then
                block_list.Clear
                Exit Function
            End If
        End If
    Next circle_x

    For cylin_Y = 1 To 100
        stvalue = InputBox("Enter Y coordinate", "Centre Y of the Circle", , 0, 0)
        If GetParamValue(stvalue) Then
            CircleVariable(1) = Val(stvalue)
            Exit For
        Else
            w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
            If w_do = 2 Then
                block_list.Clear
                Exit Function
            End If
        End If
    Next cylin_Y

```



```

        End If
    Next cylin_Y
    For cylin_R = 1 To 100
        stvalue = InputBox("Enter the Radius of that circle ", "Radius of the circle", , 0, 0)
        If GetParamValue(stvalue) Then
            CircleVariable(2) = Val(stvalue)
            Exit For
        Else
            w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
            If w_do = 2 Then
                block_list.Clear
                Exit Function
            End If
        End If
    Next cylin_R
    LineInter.AddItem "circle"
    For i_circle = 0 To UBound(CircleVariable())
        LineInter.AddItem CircleVariable(i_circle)
    Next i_circle
    Edit_Circle = True
'ARC*****

    ElseIf (keyword_1(j) = "arc") And (indx > 0) Then
        check_arc = True
        For arc_x = 1 To 100
            stvalue = InputBox("Enter the centre of X of the ARC", "Centre X of the Arc", , 0, 0)
            If GetParamValue(stvalue) Then
                ArcVariable(0) = Val(stvalue)
                Exit For
            Else
                w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
                If w_do = 2 Then
                    block_list.Clear
                    Exit Function
                End If
            End If
        Next arc_x
        For arc_Y = 1 To 100
            stvalue = InputBox("Enter the Centre of Y of the ARC", "Centre Y of the Arc", , 0, 0)
            If GetParamValue(stvalue) Then
                ArcVariable(1) = Val(stvalue)
                Exit For
            Else
                w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
                If w_do = 2 Then
                    block_list.Clear
                    Exit Function
                End If
            End If
        Next arc_Y
        For Arc_R = 1 To 100
            stvalue = InputBox("Enter the Radius of the ARC ", "Radius of the Arc", , 0, 0)
            If GetParamValue(stvalue) Then
                ArcVariable(2) = Val(stvalue)
                Exit For
            Else
                w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
                If w_do = 2 Then
                    block_list.Clear
                    Exit Function
                End If
            End If
        Next Arc_R

        For S_D = 1 To 100
            stvalue = InputBox("Enter the start angle of the ARC ", "Angle of the Arc", , 0, 0)
            If GetParamValue(stvalue) Then
                ArcVariable(3) = Val(stvalue)
                Exit For
            Else
                w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
                If w_do = 2 Then
                    block_list.Clear
                    Exit Function
                End If
            End If
        Next S_D
    End If
End Sub

```

```

End If
Next S_D

For E_A = 1 To 100
    stvalue = InputBox("Enter the end Angle of the ARC ", "Angle of the Arc", , 0, 0)
    If GetParamValue(stvalue) Then
        ArcVariable(4) = Val(stvalue)
        Exit For
    Else
        w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
        If w_do = 2 Then
            block_list.Clear
            Exit Function
        End If
    End If
Next E_A
LineInter.AddItem "arc"
LineInter.AddItem ArcVariable(0)
LineInter.AddItem ArcVariable(1)
LineInter.AddItem ArcVariable(2)
Call findXYfromAngle(ArcVariable(3), ArcVariable(0), ArcVariable(1), ArcVariable(2), ArcStartX, ArcStartY)
Call findXYfromAngle(ArcVariable(4), ArcVariable(0), ArcVariable(1), ArcVariable(2), ArcEndX, ArcEndY)
LineInter.AddItem ArcStartX
LineInter.AddItem ArcStartY
LineInter.AddItem ArcEndX
LineInter.AddItem ArcEndY

'Line
ElseIf (keyword_1(j) = "line") And (indx > 0) Then
    check_line = True
    For line_Sx = 1 To 100
        stvalue = InputBox("Enter the starting point X of the Line", "X Coordinate", , 0, 0)
        If GetParamValue(stvalue) Then
            LineVariable(0) = Val(stvalue)
            Exit For
        Else
            w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
            If w_do = 2 Then
                block_list.Clear
                Exit Function
            End If
        End If
    Next line_Sx

    For line_Sy = 1 To 100
        stvalue = InputBox("Enter the starting point Y of the Line", "Y coordinate", , 0, 0)
        If GetParamValue(stvalue) Then
            LineVariable(1) = Val(stvalue)
            Exit For
        Else
            w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
            If w_do = 2 Then
                block_list.Clear
                Exit Function
            End If
        End If
    Next line_Sy

    For line_Ex = 1 To 100
        stvalue = InputBox("Enter the End point X of the Line", "X coordinate", , 0, 0)
        If GetParamValue(stvalue) Then
            LineVariable(2) = Val(stvalue)
            Exit For
        Else
            w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
            If w_do = 2 Then
                block_list.Clear
                Exit Function
            End If
        End If
    Next line_Ex

    For line_Ey = 1 To 100
        stvalue = InputBox("Enter the End point Y of the Line", "Y coordinate", , 0, 0)
        If GetParamValue(stvalue) Then
            LineVariable(3) = Val(stvalue)
            Exit For
        End If
    Next line_Ey

```

```

Else
    w_do = MsgBox("Did not give right value press OK for continue cancel for Quit", vbOKCancel, "Be Confirm")
    If w_do = 2 Then
        block_list.Clear
        Exit Function
    End If

    End If
Next line_Ey
Call Linerearrange(LineVariable(0), LineVariable(1), LineVariable(2), LineVariable(3))
LineInter.AddItem "line"
For l_line = 0 To UBound(LineVariable())
    LineInter.AddItem LineVariable(l_line)
Next l_line

End If
Next i

Next j

End If
End If

Call Delete_Selectionset
'BLOCK DYNAMIC ARRAY
Dim count As Integer
Dim count_1 As Integer
count = 0
count_1 = 0
For i = 0 To block_list.ListCount
    If block_list.List(i) = "block" Then
        count = count + 1
    End If
Next i
If Check_Block = True Then
    count = count - 1
    ReDim block_Array(count)
    For ii_1 = 0 To block_list.ListCount
        If block_list.List(ii_1) = "block" Then
            block_Array(count_1) = block_list.List(ii_1 + 1)
            count_1 = count_1 + 1
        End If
    Next ii_1
End If

'Circle Dynamic Array As Double
Call ReloadCircle_Array

'checking for ARC
Call ReloadArc_Array
'Cylinder Dynamic Array

Dim Cylinder_count As Integer
Cylinder_count = 0
count_1 = 0
For i = 0 To block_list.ListCount
    If block_list.List(i) = "cylinder" Then
        Cylinder_count = Cylinder_count + 1
    End If
Next i

If check_cylinder = True Then
    Cylinder_count = Cylinder_count - 1
    ReDim Cylinder_Array(Cylinder_count)
    For ii_1 = 0 To block_list.ListCount
        If block_list.List(ii_1) = "cylinder" Then
            Cylinder_Array(count_1) = block_list.List(ii_1 + 1)
            count_1 = count_1 + 1
        End If
    Next ii_1
End If

```

```
'Line Dynamic Array
Call ReloadLine_Array

If (Edit_Circle = True) Or (Edit_ARC = True) Or (Edit_line = True) Then
' acadDoc.SelectionSets.Item("SELECTIONSET_text").Delete
End If
```

```
'Call the close function for disable the close button
'm_blnCloseEnabled = False
'EnableCloseButton Me.hWnd, m_blnCloseEnabled
'StartAnimatedCursor ("c:\WINNT\Cursors\globe.ani")
```

```
On Error Resume Next
'Get the AutoCAD Application object if AutoCAD is running
Set acadApp = GetObject(, "AutoCAD.Application")
If Err Then
Err.Clear
' Start AutoCAD if it is not running.
Set acadApp = CreateObject("AutoCAD.Application")

acadApp.Visible = True
If Err Then
MsgBox Err.Description
Exit Function
End If
End If
```

```
Set acadDoc = acadApp.ActiveDocument
Set mspace = acadDoc.ModelSpace
frmDraw.Show
```

```
'Bringing the last cursor when the application was not busy
'RestoreLastCursor
```

```
'Call the close function for enable the close button
m_blnCloseEnabled = True

'EnableCloseButton Me.hWnd, m_blnCloseEnabled
```

```
"Draw Block in Auto cad interface
If block_Array(0) >= 0 Then
```

```
i = 0
```

```
For i = 0 To UBound(block_Array()) Step 5
```

```
Dim length As Double
Dim width As Double
Dim Height As Double
Dim center(0 To 2) As Double
' Define the box
center(0) = block_Array(i)
center(1) = block_Array(i + 1)
center(2) = 0
length = block_Array(i + 2)
width = block_Array(i + 3)
Height = block_Array(i + 4)
```

```
' Create the box object in model space
```

```
AppActivate acadApp.Caption
```

```
Next i
End If
```

' Create a cylinder in model space.

```
Dim centerPoint(0 To 2) As Double
Dim Radius As Double
Dim height_Cylin As Double
i = 0
For i = LBound(Cylinder_Array()) To UBound(Cylinder_Array()) Step 4

    ' Define the circle
    centerPoint(0) = Cylinder_Array(i)
    centerPoint(1) = Cylinder_Array(i + 1)
    centerPoint(2) = 0
    Radius = Cylinder_Array(i + 2)
    height_Cylin = Cylinder_Array(i + 3)
    ' Create the Cylinder object in model space
    'Set cylinderObj = mspace.AddCylinder(centerPoint, radius, height_Cylin)
```

```
Next i
Circle_count = Circle_count + 1
'Total_Circle = ((UBound(Circle_Array) + 1) / 3)
ReDim Circleobj(Total_Circle - 1) As Object
'Define the circle
'Call Draw_Circle_selectionset
Call Draw_Circle_selectionset
'Call Draw_Arc_selectionset
Call Draw_Arc_selectionset
Call Draw_Line_selectionset
Call Draw_ObjectNumber
Call Objectlist
'Sending commnad to Auto CAD command line
'acadDoc.sendcommand " _zoom a "
'Me.MousePointer = 0
'Call Edit
Form1.Show
End Function
```

```
Private Sub Command3_Click()
Call OpenFile
End Sub
Function OpenFile()
Dim ArcStartX As Single, ArcStartY As Single, ArcEndX As Single, ArcEndY As Single
Dim ArcCenterX As Single, ArcCenterY As Single, ArcRadius As Single, ArcSangle As Single, ArcEangle As Single
```

```
Dim NX() As String
Dim X() As String
Dim each_c As String
Dim xx() As Double
Dim temp As Double
LineInter.Clear
Dim Result(3) As String
' CancelError is True.
On Error GoTo ErrHandler
```

```
Cdgl.FileName = ""
Cdgl.Filter = "DXF Files (*.dxf)*.dxf"
Cdgl.ShowOpen
dxfFile = Cdgl.FileName
Dim Out As String
LineInter.Clear
Out = ReadDXF(dxfFile, "ENTITIES", "LINE", ",10,20,11,21,")
```

```
Dim counter As Integer, ObjectCounte As Integer, Store As Integer
Dim Storestring As String, Lenthstring As Integer, L As Integer
Lenthstring = 0
Store = 0
counter = 0
Storestring = ""
ObjectCounte = 0
X = Split(Out)
Dim i As Integer
i = 0
```

```

NX = Split(Out, ",")
Erase xx()
ReDim xx(UBound(NX) - 1)
For k = 0 To UBound(NX)
    'xx(k) = Val(NX(k))

    If k = UBound(NX) Then
        xx(k - 1) = FormatNumber(Val(NX(k)), 3)
        counter = counter + 1
        Exit For
    ElseIf k > 0 And k < UBound(NX) Then
        Lenthstring = Len(NX(k))
        Storestring = Left$(NX(k), (Lenthstring - 2))
        If (IsNumeric(Storestring)) Then
            xx(k - 1) = FormatNumber(Val(Storestring), 3)
        End If
        counter = counter + 1
    End If
End If

Next k

Store = 0
If UBound(xx()) > 0 Then
    For m = 1 To counter / 4
        LineInter.AddItem "line"
        For i = Store To counter
            LineInter.AddItem xx(i)
            ObjectCounte = ObjectCounte + 1
            Store = Store + 1
            If ObjectCounte = 4 Then
                ObjectCounte = 0
            Exit For
            End If
        Next i
    Next m
End If
ElseIf Out <> "" Then
    ReDim xx(UBound(X))
    For z = 0 To UBound(X)
        each_c = InStr(X(z), ",")
        If each_c > 0 Then
            sster = Split(X(z), ",")
            If (UBound(sster) >= 1) Then
                If (IsNumeric(sster(1))) Then
                    xx(i) = Val(sster(1))
                    counter = counter + 1
                    i = i + 1
                    'Debug.Print xx(z)
                End If
            End If
        Else
            If (IsNumeric(X(z))) Then
                xx(i) = Val(X(z))
                i = i + 1
                counter = counter + 1
                'Debug.Print xx(z)
            End If
        End If
    Next z

    If UBound(xx()) > 0 Then
        For j = 1 To counter / 4
            LineInter.AddItem "line"
            For i = Store To counter
                LineInter.AddItem xx(i)
                ObjectCounte = ObjectCounte + 1
                Store = Store + 1
                If ObjectCounte = 4 Then
                    ObjectCounte = 0
                Exit For
            End If
        Next i
    Next j

```



```

End If
End If

Out = ReadDXF(dxFile, "ENTITIES", "CIRCLE", ",10,20,40,")

counter = 0
Store = 0
X = Split(Out)
Erase xx()
ReDim xx(UBound(X))
i = 0
If UBound(X) = 0 Then
NX = Split(Out, ",")
Erase xx()
ReDim xx(UBound(NX) - 1)
For k = 0 To UBound(NX)
'xx(k) = Val(NX(k))

If k = UBound(NX) Then
xx(k - 1) = FormatNumber(Val(NX(k)), 3)
counter = counter + 1
Exit For
Elseif k > 0 And k < UBound(NX) Then
Lenthstring = Len(NX(k))
Storestring = Left$(NX(k), (Lenthstring - 2))
If (IsNumeric(Storestring)) Then
xx(k - 1) = FormatNumber(Val(Storestring), 3)
End If
counter = counter + 1
End If

Next k

Store = 0

If UBound(xx()) > 0 Then
For m = 1 To counter / 3
LineInter.AddItem "circle"
For i = Store To counter
LineInter.AddItem FormatNumber(xx(i), 3)
ObjectCount = ObjectCount + 1
Store = Store + 1
If ObjectCount = 3 Then
ObjectCount = 0
Exit For
End If
Next i
Next m
End If

Elseif Out <> "" Then
Erase xx()
ReDim xx(UBound(X))
'Dim i As Integer
For z = 0 To UBound(X)
each_c = InStr(X(z), ",")
If each_c > 0 Then
sster = Split(X(z), ",")
If (UBound(sster) >= 1) Then
If (IsNumeric(sster(1))) Then
xx(i) = Val(sster(1))
counter = counter + 1
i = i + 1
'Debug.Print xx(z)
End If
End If
Else
If (IsNumeric(X(z))) Then
xx(i) = Val(X(z))
counter = counter + 1
i = i + 1
'Debug.Print xx(z)

```



```

        End If
    End If
Next z
If UBound(xx()) > 0 Then
    For j = 1 To counter / 3
        LineInter.AddItem "circle"
        For i = Store To counter
            LineInter.AddItem xx(i)
            ObjectCounte = ObjectCounte + 1
            Store = Store + 1
            If ObjectCounte = 3 Then
                ObjectCounte = 0
            Exit For
        End If
    Next i
Next j
End If

End If

counter = 0
Store = 0
Out = ReadDXF(dxflfile, "ENTITIES", "ARC", ",,10,20,40,50,51,")
Erase xx()

X = Split(Out)

'Dim i As Integer
i = 0
If UBound(X) = 0 Then
    NX = Split(Out, ",")
    Erase xx()
    ReDim xx(UBound(NX) - 1)
    For k = 0 To UBound(NX)
        'xx(k) = Val(NX(k + 1))
        If k = UBound(NX) Then
            xx(k - 1) = FormatNumber(Val(NX(k)), 3)
            counter = counter + 1
        Exit For
    ElseIf k > 0 And k < UBound(NX) Then
        Lenthstring = Len(NX(k))
        Storestring = Left$(NX(k), (Lenthstring - 2))
        If (IsNumeric(Storestring)) Then
            xx(k - 1) = FormatNumber(Val(Storestring), 3)
        End If
        counter = counter + 1
    End If
Next k
If UBound(xx()) > 0 Then

    For i = 0 To (counter - 1) Step 5
        LineInter.AddItem "arc"
        LineInter.AddItem xx(i)
        LineInter.AddItem xx(i + 1)
        LineInter.AddItem xx(i + 2)
        ArcCenterX = xx(i)
        ArcCenterY = xx(i + 1)
        ArcRadius = xx(i + 2)
        ArcSangle = xx(i + 3)
        ArcEangle = xx(i + 4)
        Call findXYfromAngle(ArcSangle, ArcCenterX, ArcCenterY, ArcRadius, ArcStartX, ArcStartY)
        Call findXYfromAngle(ArcEangle, ArcCenterX, ArcCenterY, ArcRadius, ArcEndX, ArcEndY)
        LineInter.AddItem ArcStartX
        LineInter.AddItem ArcStartY
        LineInter.AddItem ArcEndX
        LineInter.AddItem ArcEndY
        ObjectCounte = ObjectCounte + 1
        Store = Store + 1
    Next i

End If

```

```

Elseif Out <> "" Then
    Erase xx()
    ReDim xx(UBound(X))
    For z = 0 To UBound(X)
        each_c = InStr(X(z), ",")
        If each_c > 0 Then
            sster = Split(X(z), ",")
            If (UBound(sster) >= 1) Then
                If (IsNumeric(sster(1))) Then
                    xx(i) = Val(sster(1))
                    counter = counter + 1
                    i = i + 1
                End If
            End If
        Else
            If (IsNumeric(X(z))) Then
                xx(i) = Val(X(z))
                counter = counter + 1
                i = i + 1
            End If
        End If
    Next z
    If UBound(xx()) > 0 Then

        For i = 0 To (counter - 1) Step 5
            LineInter.AddItem "arc"
            LineInter.AddItem xx(i)
            LineInter.AddItem xx(i + 1)
            LineInter.AddItem xx(i + 2)
            ArcCenterX = xx(i)
            ArcCenterY = xx(i + 1)
            ArcRadius = xx(i + 2)
            ArcSangle = xx(i + 3)
            ArcEangle = xx(i + 4)
            Call findXYfromAngle(ArcSangle, ArcCenterX, ArcCenterY, ArcRadius, ArcStartX, ArcStartY)
            Call findXYfromAngle(ArcEangle, ArcCenterX, ArcCenterY, ArcRadius, ArcEndX, ArcEndY)
            LineInter.AddItem ArcStartX
            LineInter.AddItem ArcStartY
            LineInter.AddItem ArcEndX
            LineInter.AddItem ArcEndY
            ObjectCounte = ObjectCounte + 1
            Store = Store + 1
        Next i
    End If
End If

On Error Resume Next
'Get the AutoCAD Application object if AutoCAD is running
Set acadApp = GetObject("AutoCAD.Application")
If Err Then
    Err.Clear
    ' Start AutoCAD if it is not running.
    Set acadApp = CreateObject("AutoCAD.Application")

    acadApp.Visible = True
    If Err Then
        MsgBox Err.Description
        Exit Function
    End If
End If

Set acadDoc = acadApp.ActiveDocument
Set mspace = acadDoc.ModelSpace

Call Delete_Selectionset
Call ReloadCircle_Array

```

```
Call ReloadArc_Array
Call ReloadLine_Array
Call Draw_Circle_selectionset
    'Call Draw_Arc_selectionset
    Call Draw_Arc_selectionset
    Call Draw_Line_selectionset
    Call Draw_ObjectNumber
ErrorHandler:
' Sending command to AutoCAD command line
acadDoc.sendcommand " _zoom a "
' User pressed Cancel button.
Exit Function

End Function

Private Sub Command4_Click()
Call Closefile
End Sub
Function Closefile()
On Error Resume Next
' Get the AutoCAD Application object if AutoCAD is running.
Set acadApp = GetObject( "AutoCAD.Application")
If Err Then
Err.Clear
End If
Set acadDoc = acadApp.ActiveDocument
Set mspace = acadDoc.ModelSpace

' This example cycles through the documents collection
' and closes all open drawings using the Close method.

' If there are no open documents, then exit
If mspace.count = 0 Then
MsgBox "There are no open documents!"
Exit Function
End If

' Close all open documents
For Each acadDoc In Documents
If MsgBox("Do you wish to close the document: " & DOC.WindowTitle, vbYesNo & vbQuestion) = vbYes Then
If acadDoc.FullName <> "" Then
acadDoc.Close
Else
MsgBox acadDoc.Name & " has not been saved yet, so it will not be closed."
End If
End If
Next

End Function

Public Sub ACAD_Appli()
On Error Resume Next
' Get the AutoCAD Application object if AutoCAD is running.
Set acadApp = GetObject( "AutoCAD.Application")
If Err Then
Err.Clear
' Start AutoCAD if it is not running.
Set acadApp = CreateObject("AutoCAD.Application")
acadApp.Visible = True
If Err Then
MsgBox Err.Description
Exit Sub
End If
End If
Set acadDoc = acadApp.ActiveDocument
End Sub

Private Sub Command5_Click()
' Get the AutoCAD Application object if AutoCAD is running.
Set acadApp = GetObject( "AutoCAD.Application")
If Err Then
Err.Clear
' Start AutoCAD if it is not running.
```

```

Set acadApp = CreateObject("AutoCAD.Application")
acadApp.Visible = True
If Err Then
    MsgBox Err.Description
Exit Sub
End If
End If
Set acadDoc = acadApp.ActiveDocument
Set mspace = acadDoc.ModelSpace

MsgBox "Perform a ZoomScaled using:" & vbCrLf & _
    "Scale Type: acZoomScaledRelative" & vbCrLf & _
    "Scale Factor: 2", , "ZoomScaled"

Dim scaleFactor As Double
Dim ScaleType As Integer

scaleFactor = 4
ScaleType = acZoomScaledRelative
acadDoc.sendcommand " zoom " & scaleFactor & vbCrLf
'& scaleFactor & vbCrLf
End Sub

Public Sub FindLineIntersection( _
    ByVal x11 As Single, ByVal y11 As Single, _
    ByVal x12 As Single, ByVal y12 As Single, _
    ByVal x21 As Single, ByVal y21 As Single, _
    ByVal x22 As Single, ByVal y22 As Single, _
    ByRef inter_x As Single, ByRef inter_y As Single, ByRef t1 As Double, ByRef t2 As Double, ByRef InterSec As Boolean)

Dim dx1 As Single
Dim dy1 As Single
Dim dx2 As Single
Dim dy2 As Single
Dim denominator As Single
InterSec = False
' Get the segments' parameters.
dx1 = x12 - x11
dy1 = y12 - y11
dx2 = x22 - x21
dy2 = y22 - y21

' Solve for t1 and t2.
'On Error Resume Next
denominator = (dy1 * dx2 - dx1 * dy2)
If denominator = 0 Then
    Exit Sub
Else
    t1 = FormatNumber(((x11 - x21) * dy2 + (y21 - y11) * dx2) / _
        denominator, 4)
    t2 = FormatNumber(((x21 - x11) * dy1 + (y11 - y21) * dx1) / (-denominator), 2)

' Find the point of intersection.
If (t1 >= 0 And t1 <= 1) And (t2 >= 0 And t2 <= 1) Then
    inter_x = FormatNumber(x11 + dx1 * t1, 4)
    inter_y = FormatNumber(y11 + dy1 * t1, 4)
    InterSec = True
ElseIf (dy1 * dx2 - dx1 * dy2 = 0) Then
    Exit Sub
Else
    Exit Sub
End If
End If
'If 0 <= t1 <= 1, then the point lies on segment 1.
'If 0 <= t2 <= 1, then the point lies on segment 1.
'If dy1 * dx2 - dx1 * dy2 = 0 then the lines are parallel.
'If the point of intersection is not on both segments, then this is almost certainly not the point where the two segments are closest.

'Nearest distance from one point of the line
'D = FormatNumber(Abs(((X2 - Cx) * (Y1 - Cy)) - ((X1 - Cx) * (Y2 - Cy))) / Sqr((X2 - X1) ^ 2 + (Y2 - Y1) ^ 2))

```

End Sub

```

Private Function CircleInter(X1 As Single, Y1 As Single, X2 As Single, Y2 As Single, CX As Single, CY As Single, _
    R As Single, int_X1 As Single, int_Y1 As Single, int_x2 As Single, int_Y2 As Single, Comp As Single, ByRef
Compare As Single, oneintersec As Boolean, twointersec As Boolean)
    Dim dx As Single, dy As Single
    Dim a As Single, b As Single, C As Single
    Dim Tang As Single
    Dim tempFx As Single, tempSx As Single, tempFy As Single, tempSy As Single
    Dim D As Single
    Compare = 0
    int_X1 = 0
    int_Y1 = 0
    int_x2 = 0
    int_Y2 = 0
    a = FormatNumber((X2 - X1) ^ 2 + (Y2 - Y1) ^ 2, 5)
    b = FormatNumber(2 * ((X2 - X1) * (X1 - CX) + (Y2 - Y1) * (Y1 - CY)), 5)
    C = FormatNumber(CX ^ 2 + CY ^ 2 + X1 ^ 2 + Y1 ^ 2 - 2 * (CX * X1 + CY * Y1) - R ^ 2, 5)
    Comp = b ^ 2 - 4 * a * C
    oneintersec = False
    twointersec = False
    'If Comp < 0 Then
    'Exit Function
    Dim Distance As Single, Distance_1 As Single

    D = FormatNumber(Abs(((X2 - CX) * (Y1 - CY)) - ((X1 - CX) * (Y2 - CY))) / Sqr((X2 - X1) ^ 2 + (Y2 - Y1) ^ 2))

    Compare = FormatNumber(Abs(R - D))

    If (Compare < 0.4) Then

        int_X1 = FormatNumber(X1 + Tang * (X2 - X1), 3)
        int_Y1 = FormatNumber(Y1 + Tang * (Y2 - Y1), 3)
        int_x2 = FormatNumber(X1 + (Tang * (X2 - X1)), 3)
        int_Y2 = FormatNumber(Y1 + (Tang * (Y2 - Y1)), 3)
        Distance = FormatNumber((((int_X1 - CX) ^ 2 + (int_Y1 - CY) ^ 2) ^ (1 / 2))
        Distance_1 = Abs(Distance - R)
        If Distance_1 > 1 Then
            Tang = (-b) / (2 * a)
            int_X1 = FormatNumber(X1 + Tang * (X2 - X1), 3)
            int_Y1 = FormatNumber(Y1 + Tang * (Y2 - Y1), 3)
            int_x2 = int_X1
            int_Y2 = int_Y1
        End If
    ElseIf Comp > 0 Then
        Tang = (-b + Sqr(b ^ 2 - 4 * a * C)) / (2 * a)

        int_X1 = FormatNumber(X1 + Tang * (X2 - X1), 3)
        int_Y1 = FormatNumber(Y1 + Tang * (Y2 - Y1), 3)

        Tang = (-b - Sqr(b ^ 2 - 4 * a * C)) / (2 * a)
        int_x2 = FormatNumber(X1 + (Tang * (X2 - X1)), 3)
        int_Y2 = FormatNumber(Y1 + (Tang * (Y2 - Y1)), 3)

        If X1 > X2 Then
            tempFx = X2
            tempSx = X1
        Else
            tempFx = X1
            tempSx = X2
        End If

        If Y1 > Y2 Then
            tempFy = Y2
            tempSy = Y1
        Else
            tempFy = Y1
            tempSy = Y2
        End If
        If ((int_X1 >= tempFx) And (int_X1 <= tempSx)) And ((int_Y1 >= tempFy) And (int_Y1 <= tempSy)) Then
            oneintersec = True
        End If
        If ((int_x2 >= tempFx) And (int_x2 <= tempSx)) And ((int_Y2 >= tempFy) And (int_Y2 <= tempSy)) Then

```

```

        twointersec = True
    End If
End If

End Function
Private Function FindBoundary(X1 As Single, Y1 As Single, X2 As Single, Y2 As Single, int_X1 As Single, int_Y1 As Single,
    int_x2 As Single, int_Y2 As Single, FirstPoint As Boolean, SecondPoint As Boolean)
FirstPoint = False
SecondPoint = False
    If X1 > X2 Then
        tempFx = X2
        tempSx = X1
    Else
        tempFx = X1
        tempSx = X2
    End If

    If Y1 > Y2 Then
        tempFy = Y2
        tempSy = Y1
    Else
        tempFy = Y1
        tempSy = Y2
    End If
    If ((int_X1 >= tempFx) And (int_X1 <= tempSx)) And ((int_Y1 >= tempFy) And (int_Y1 <= tempSy)) Then
        FirstPoint = True
    Else
        FirstPoint = False
    End If
    If ((int_x2 >= tempFx) And (int_x2 <= tempSx)) And ((int_Y2 >= tempFy) And (int_Y2 <= tempSy)) Then
        SecondPoint = True
    Else
        SecondPoint = False
    End If
End Function
Private Function ArcnotDevideSt(StartArcx As Single, StartArcy As Single, EndArcx As Single, EndArcy As Single, int_X1 As
Single, _
    int_Y1 As Single, oneintersec As Boolean) As Boolean
ArcnotDevideSt = False
    If StartArcx = int_X1 And StartArcy = int_Y1 Then
        ArcnotDevideSt = True
    End If

End Function
Private Function ArcnotDevideEn(StartArcx As Single, StartArcy As Single, EndArcx As Single, EndArcy As Single, int_x2 As
Single, _
    int_Y2 As Single, twointersec As Boolean) As Boolean
ArcnotDevideEn = False
    If EndArcx = int_x2 And EndArcy = int_Y2 Then
        ArcnotDevideEn = True
    End If

End Function
Private Function Match_First_Intersection(x111 As Single, y111 As Single, x122 As Single, y122 As Single, int_X1 As Single,
int_Y1 As Single, _
    Boun_oneintersec As Boolean) As Boolean
    If ((x111 = int_X1) And (y111 = int_Y1)) Or ((x122 = int_X1) And (y122 = int_Y1)) Then
        Match_First_Intersection = True
    End If
    If x111 > x122 Then
        tempFx = x122
        tempSx = x122
    Else
        tempFx = x111
        tempSx = x122
    End If

    If y111 > y122 Then
        tempFy = y122
        tempSy = y111
    Else

```

```

    tempFy = y111
    tempSy = y122
End If
If ((int_X1 >= tempFx) And (int_X1 <= tempSx)) And ((int_Y1 >= tempFy) And (int_Y1 <= tempSy)) Then
    Boun_oneintersec = True
End If

End Function

Private Sub Command6_Click()
Call Edit
End Sub
Function Edit()
Dim Ekeyword_1(4), EparamStr_1 As String, ETokens() As String, EParamCount As Integer, EParamValue As Double
Dim Ecount_5 As Boolean, Eparamvalue_1 As Boolean, EtoKen_1() As String, EFor_cen As Integer, EParamstr_2 As String,
EParamstr_3 As String
Dim EFind_p As Integer
Dim stvalue_2 As String
Ekeyword_1(0) = "fillet"
Ekeyword_1(1) = "chamfer"
Ekeyword_1(2) = "delete"
Dim InterSec As Boolean
InterSec = False
Dim i As Integer
i = 0
Dim CircleIndex As Integer, ArcIndex As Integer
Dim CheckCircle As Integer, CheckArc As Integer, checkObj As Integer, CheckLine As Integer, dependOn As Integer

Dim Ecount_2 As Integer, EcountOFcenter As Integer
Dim Check_fillet As Boolean, check_chamfer As Boolean
Dim Found1 As Boolean, Found2 As Boolean, Permit1 As Boolean, Permit2 As Boolean
Dim FirstPointX As Integer, FirstPointY As Integer, FirstPointX2 As Integer, FirstPointY2 As Integer
Dim C_Permitt1 As Boolean, C_Permitt2 As Boolean
Dim oRder() As Integer
Dim FirstlineIntersectionTrue As Boolean, SecondlineIntersectionTrue As Boolean
Dim X1 As Single, Y1 As Single, X2 As Single, Y2 As Single, CX As Single, CY As Single, dr As Single, D As Single, R As
Single
Dim StartArcx As Single, StartArcy As Single, EndArcx As Single, EndArcy As Single, Boun_oneintersec As Boolean,
FirstPoint As Boolean, SecondPoint As Boolean
Dim firstinterX As Single, firstinterY As Single, Angle1 As Single, Angle2 As Single, Angle3 As Single, Angle4 As Single,
StartArcangle As Single, EndArcangle As Single
Dim interFirstpoint As Boolean, InterSecondpoint As Boolean, TopX As Single, TopY As Single, LineTopAngle As Single
Dim EndX As Single, EndY As Single, LineEndangle As Single, NearestPoint As Boolean, LongestPoint As Boolean
Dim onepoint As Boolean, otherpoint As Boolean
C_Permitt1 = False
C_Permitt2 = True
Permitt1 = False
Permitt2 = True
Check_fillet = False
check_chamfer = False
Circle_Counter = 0
countOFcenter = 0
count_2 = 0
count_5 = 0
Found1 = False
Found2 = False
Dim m As Integer
Dim x111 As Single, y111 As Single, x122 As Single, y122 As Single, x211 As Single, y211 As Single, x222 As Single, y222
As Single
Dim t1 As Double, t2 As Double
Dim TCount_Intersec As Integer
Dim CopyArray() As Double
TCount_Intersec = 0
On Error Resume Next
Set acadApp = GetObject("AutoCAD.Application")
If Err Then
    Err.Clear
    ' Start AutoCAD if it is not running.
    Set acadApp = CreateObject("AutoCAD.Application")
    acadApp.Visible = True
    If Err Then
        MsgBox Err.Description
        Exit Function
    End If
End If

```

```

Set acadDoc = acadApp.ActiveDocument
Set mspace = acadDoc.ModelSpace
eComStr = Split(Text1.Text)
Dim inter_x As Single, inter_y As Single
Dim C As Integer
Dim a As Integer, startAngleInDegree As Single, endAngleInDegree As Single, startAngleInRadian As Single,
endAngleInRadian As Single
Dim FormatedStartangleX As Single, FormatedStartangleY As Single, FormatedEndangleX As Single, FormatedEndangleY
As Single

'Check fillet = True

'Rearranging of line coordinate compairing with center point 0,0
For i = 0 To LineInter.ListCount
    For i + 5 = LineInter.ListCount Then
        Exit For
    End If
    If LineInter.List(i) = "line" Then
        'For lineorder = LBound(Line_Array()) To UBound(Line_Array()) Step 4
        LineX1 = Val(LineInter.List(i + 1))
        LineY1 = Val(LineInter.List(i + 2))
        LineX2 = Val(LineInter.List(i + 3))
        LineY2 = Val(LineInter.List(i + 4))
        d1 = ((LineX1) ^ 2 + (LineY1) ^ 2) ^ 1 / 2
        d2 = ((LineX2) ^ 2 + (LineY2) ^ 2) ^ 1 / 2
        If d1 > d2 Then
            swapvx = LineX1
            LineInter.List(i + 1) = LineX2
            LineInter.List(i + 3) = swapvx

            swapvy = LineY1
            LineInter.List(i + 2) = LineY2
            LineInter.List(i + 4) = swapvy
        End If
        'Next lineorder
    End If
Next i

'Line Line intersection

For i = 0 To (LineInter.ListCount)
    Do While i <= LineInter.ListCount
        'If (i + 5) = (LineInter.ListCount + 1) Then
        ' Exit For
        'End If
        If LineInter.List(i) = "line" Then
            x111 = Val(LineInter.List(i + 1))
            y111 = Val(LineInter.List(i + 2))
            x122 = Val(LineInter.List(i + 3))
            y122 = Val(LineInter.List(i + 4))

            For j = 0 To (LineInter.ListCount - 5)
                If LineInter.List(j) = "line" Then
                    x211 = Val(LineInter.List(j + 1))
                    y211 = Val(LineInter.List(j + 2))
                    x222 = Val(LineInter.List(j + 3))
                    y222 = Val(LineInter.List(j + 4))

                    Call FindLineIntersection(x111, y111, x122, y122, x211, y211, x222, y222, inter_x, inter_y, t1, t2,
InterSec)

                    If InterSec = True Then
                        'Considering first line
                        If SameWithendpointandIntersection(x111, y111, x122, y122, inter_x, inter_y) = False And InterSec =
True Then
                            'Devide the line into two segment
                            LineInter.List(i + 3) = inter_x
                            LineInter.List(i + 4) = inter_y
                        End If
                    End If
                End If
            Next j
        End If
    Loop
Next i

```



```

LineInter.AddItem "line"
LineInter.AddItem (inter_x)
LineInter.AddItem (inter_y)
LineInter.AddItem (x122)
LineInter.AddItem (y122)
'InterSec = False
FirstlineIntersectionTrue = True
End If
'Considering second line
If SameWithendpointandIntersection(x211, y211, x222, y222, inter_x, inter_y) = False And InterSec =
True Then

    LineInter.List(j + 3) = inter_x
    LineInter.List(j + 4) = inter_y

    LineInter.AddItem "line"
    LineInter.AddItem (inter_x)
    LineInter.AddItem (inter_y)
    LineInter.AddItem (x222)
    LineInter.AddItem (y222)
    'InterSec = False
    SecondlineIntersectionTrue = True
End If
If FirstlineIntersectionTrue = True Or SecondlineIntersectionTrue = True Then
    FirstlineIntersectionTrue = False
    SecondlineIntersectionTrue = False
    i = -1
    Exit For
End If
End If
End If

Next j
End If
i = i + 1
Next i
Loop

'For finding out the inter section between Line and Circle
'For i = 0 To (LineInter.ListCount)
Dim coMpx111 As Single, coMpy111 As Single, coMpx122 As Single, coMpy122 As Single, coMpintx1 As Single,
coMpinty1 As Single
Dim Compare_11 As Single, D_11 As Single

i = 0
Do While i <= LineInter.ListCount
    'If (i + 5) = LineInter.ListCount Then
    '    Exit For
    'End If
    If LineInter.List(i) = "line" Then
        x111 = FormatNumber(Val(LineInter.List(i + 1)), 3)
        y111 = FormatNumber(Val(LineInter.List(i + 2)), 3)
        x122 = FormatNumber(Val(LineInter.List(i + 3)), 3)
        y122 = FormatNumber(Val(LineInter.List(i + 4)), 3)

        For k = CircleIndex To LineInter.ListCount

            If LineInter.List(k) = "circle" Then

                CX = FormatNumber(Val(LineInter.List(k + 1)), 3)
                CY = FormatNumber(Val(LineInter.List(k + 2)), 3)
                R = FormatNumber(Val(LineInter.List(k + 3)), 3)

                Call CircleInter(x111, y111, x122, y122, CX, CY, R, int_X1, int_Y1, int_x2, int_Y2, Comp,
Compare, onceintersec, twointersec)
                D_11 = FormatNumber(Abs((((x122 - CX) * (y111 - CY)) - ((x111 - CX) * (y122 - CY)))) /
Sqr((x122 - x111) ^ 2 + (y122 - y111) ^ 2), 4)
                Compare_11 = FormatNumber(Abs(R - D_11), 4)

                coMpx111 = FormatNumber(x111, 0)
                coMpy111 = FormatNumber(y111, 0)
                coMpx122 = FormatNumber(x122, 0)
            End If
        Next k
    End If
    i = i + 1
Next i

```

```

coMpy122 = FormatNumber(y122, 0)
coMpintx1 = FormatNumber(int_X1, 0)
coMpinty1 = FormatNumber(int_Y1, 0)

If (Compare < 0.4) Then
  If (coMpintx1 = coMpx111 And coMpinty1 = coMpy111) Or (coMpintx1 = coMpx122 And
coMpinty1 = coMpy122) Then
    MsgBox "Circle Tangent by one of the end point of the line"
    LineInter.List(k) = "circle" & (Circle_Counter)

    LineInter.AddItem "arc"
    LineInter.AddItem CX
    LineInter.AddItem CY
    LineInter.AddItem R
    LineInter.AddItem (int_X1)
    LineInter.AddItem (int_Y1)
    LineInter.AddItem (int_X1)
    LineInter.AddItem (int_Y1)
    i = -1
    Exit For
  Else
    LineInter.List(i + 3) = int_X1
    LineInter.List(i + 4) = int_Y1

    LineInter.AddItem "line"
    LineInter.AddItem (int_X1)
    LineInter.AddItem (int_Y1)
    LineInter.AddItem (x122)
    LineInter.AddItem (y122)

    LineInter.List(k) = "circle" & (Circle_Counter)
    Do Until ArcIndex > 1
      ArcIndex = LineInter.ListCount
    Loop
    LineInter.AddItem "arc"
    LineInter.AddItem CX
    LineInter.AddItem CY
    LineInter.AddItem R
    LineInter.AddItem (int_X1)
    LineInter.AddItem (int_Y1)
    LineInter.AddItem (int_X1)
    LineInter.AddItem (int_Y1)
    i = -1
    Exit For
  End If

ElseIf Comp > 0 Then 'after intersecting
  If (oneintersec = True) And (twointersec = True) Then
    LineInter.List(i + 3) = int_x2
    LineInter.List(i + 4) = int_Y2

    LineInter.AddItem "line"
    LineInter.AddItem (int_x2)
    LineInter.AddItem (int_Y2)
    LineInter.AddItem (int_X1)
    LineInter.AddItem (int_Y1)

    LineInter.AddItem "line"
    LineInter.AddItem (int_X1)
    LineInter.AddItem (int_Y1)
    LineInter.AddItem (x122)
    LineInter.AddItem (y122)

    Do Until ArcIndex > 1
      ArcIndex = LineInter.ListCount
    Loop
    LineInter.AddItem "arc"
    LineInter.AddItem CX
    LineInter.AddItem CY
    LineInter.AddItem R
    LineInter.AddItem (int_X1)
    LineInter.AddItem (int_Y1)
    LineInter.AddItem (int_x2)
    LineInter.AddItem (int_Y2)

```

```

LineInter.AddItem "arc"
LineInter.AddItem CX
LineInter.AddItem CY
LineInter.AddItem R
LineInter.AddItem (int_x2)
LineInter.AddItem (int_Y2)
LineInter.AddItem (int_X1)
LineInter.AddItem (int_Y1)
LineInter.List(k) = "circle" & (CircLe_Counter)
i = -1
Exit For
ElseIf oneintersec = True Then
LineInter.List(i + 3) = int_X1
LineInter.List(i + 4) = int_Y1

LineInter.AddItem "line"
LineInter.AddItem (int_X1)
LineInter.AddItem (int_Y1)
LineInter.AddItem (x122)
LineInter.AddItem (y122)

LineInter.AddItem "arc"
LineInter.AddItem CX
LineInter.AddItem CY
LineInter.AddItem R
LineInter.AddItem (int_X1)
LineInter.AddItem (int_Y1)
LineInter.AddItem (int_x2)
LineInter.AddItem (int_Y2)

LineInter.AddItem "arc"
LineInter.AddItem CX
LineInter.AddItem CY
LineInter.AddItem R
LineInter.AddItem (int_x2)
LineInter.AddItem (int_Y2)
LineInter.AddItem (int_X1)
LineInter.AddItem (int_Y1)

LineInter.List(k) = "circle" & (CircLe_Counter)
i = -1
Exit For
ElseIf twointersec = True Then
LineInter.List(i + 3) = int_x2
LineInter.List(i + 4) = int_Y2

LineInter.AddItem "line"
LineInter.AddItem (int_x2)
LineInter.AddItem (int_Y2)
LineInter.AddItem (x122)
LineInter.AddItem (y122)

Do Until ArcIndex > 1
ArcIndex = LineInter.ListCount
Loop
LineInter.AddItem "arc"
LineInter.AddItem CX
LineInter.AddItem CY
LineInter.AddItem R
LineInter.AddItem (int_x2)
LineInter.AddItem (int_Y2)
LineInter.AddItem (int_X1)
LineInter.AddItem (int_Y1)

LineInter.AddItem "arc"
LineInter.AddItem CX
LineInter.AddItem CY
LineInter.AddItem R
LineInter.AddItem (int_X1)
LineInter.AddItem (int_Y1)
LineInter.AddItem (int_x2)
LineInter.AddItem (int_Y2)

```

```

        LineInter.List(k) = "circle" & (CircLe_Counter)
        i = -1
        Exit For
    End If
End If
CircLe_Counter = CircLe_Counter + 0

End If

Next k

End If

i = i + 1
Loop

'Rearranging of line coordinate compairing with center point 0,0
For i = 0 To LineInter.ListCount
    If (i + 5) = LineInter.ListCount Then
        Exit For
    End If
    If LineInter.List(i) = "line" Then
        'For lineorder = LBound(Line_Array()) To UBound(Line_Array()) Step 4
        LineX1 = Val(LineInter.List(i + 1))
        LineY1 = Val(LineInter.List(i + 2))
        LineX2 = Val(LineInter.List(i + 3))
        LineY2 = Val(LineInter.List(i + 4))
        d1 = ((LineX1) ^ 2 + (LineY1) ^ 2) ^ 1 / 2
        d2 = ((LineX2) ^ 2 + (LineY2) ^ 2) ^ 1 / 2
        If d1 > d2 Then
            swapvx = LineX1
            LineInter.List(i + 1) = LineX2
            LineInter.List(i + 3) = swapvx

            swapvy = LineY1
            LineInter.List(i + 2) = LineY2
            LineInter.List(i + 4) = swapvy
        End If
        'Next lineorder
    End If
Next i

'for finding out line and Arc intersection
Dim LineArccounter As Integer, ArcLineCounter As Integer, CheckArc1 As Boolean, CheckLine1 As Boolean
Dim DistancefromStartpoint As Single, DistancefromEndpoint As Single, DistanceisLess As Boolean
CheckArc1 = False
CheckLine1 = False
LineArccounter = 0
ArcLineCounter = 0

Do While LineArccounter <= LineInter.ListCount

    If LineInter.List(LineArccounter) = "line" Then
        x111 = FormatNumber(Val(LineInter.List(LineArccounter + 1)), 3)
        y111 = FormatNumber(Val(LineInter.List(LineArccounter + 2)), 3)
        x122 = FormatNumber(Val(LineInter.List(LineArccounter + 3)), 3)
        y122 = FormatNumber(Val(LineInter.List(LineArccounter + 4)), 3)
    End If

```

```

'For k = 0 To LineInter.ListCount
Do While ArcLineCounter <= LineInter.ListCount
  If LineInter.List(ArcLineCounter) = "arc" Then
    CX = FormatNumber(Val(LineInter.List(ArcLineCounter + 1)), 3)
    CY = FormatNumber(Val(LineInter.List(ArcLineCounter + 2)), 3)
    R = FormatNumber(Val(LineInter.List(ArcLineCounter + 3)), 3)
    StartArcx = FormatNumber(Val(LineInter.List(ArcLineCounter + 4)), 3)
    StartArcy = FormatNumber(Val(LineInter.List(ArcLineCounter + 5)), 3)
    EndArcx = FormatNumber(Val(LineInter.List(ArcLineCounter + 6)), 3)
    EndArcy = FormatNumber(Val(LineInter.List(ArcLineCounter + 7)), 3)
    int_X1 = 0
    int_Y1 = 0
    int_x2 = 0
    int_Y2 = 0
    DistanceisLess = False
    Call CircleInter(x111, y111, x122, y122, CX, CY, R, int_X1, int_Y1, int_x2, int_Y2, Comp,
Compare, onceintersec, twointersec)

    CheckArc1 = False
    CheckLine1 = False

    D_11 = FormatNumber(Abs(((x122 - CX) * (y111 - CY)) - ((x111 - CX) * (y122 - CY))) /
Sqr((x122 - x111) ^ 2 + (y122 - y111) ^ 2), 4)
    Compare_11 = FormatNumber(Abs(R - D_11), 4)
    If (Compare_11 < 0.4) Then ' Arc is tangent
      'Find out Intersection layed between startpoint and End point
      firstinterX = StartArcx - CX
      FirstinterY = StartArcy - CY
      StartArcangle = FormatNumber(atan2(firstinterX, FirstinterY), 3)

      firstinterX = EndArcx - CX
      FirstinterY = EndArcy - CY
      EndArcangle = FormatNumber(atan2(firstinterX, FirstinterY), 3)

      firstinterX = int_X1 - CX
      FirstinterY = int_Y1 - CY
      Angle1 = FormatNumber(atan2(firstinterX, FirstinterY), 3)
      Angle2 = Angle1
      DistancefromStartpoint = Sqr((StartArcx - int_X1) ^ 2 + (StartArcy - int_Y1) ^ 2)
      DistancefromEndpoint = Sqr((EndArcx - int_X1) ^ 2 + (EndArcy - int_Y1) ^ 2)
      Call checkAngle(Angle1, Angle2, StartArcangle, EndArcangle, interFirstpoint,
InterSecondpoint)

      If DistancefromStartpoint < 3 Or DistancefromEndpoint < 3 Then
        'Consider intersectionpoint is same with the both Start point and End point No need to devide
        'MsgBox "smaller than 3"
        DistanceisLess = True
      Elseif interFirstpoint = True And DistanceisLess = False Then 'Intersection point between
startpoint and endpoint

        LineInter.List(ArcLineCounter + 6) = int_X1
        LineInter.List(ArcLineCounter + 7) = int_Y1
        Call GeneralARC(CX, CY, R, int_X1, int_Y1, EndArcx, EndArcy)
        CheckArc1 = True

      End If
      'Comparing with the end point of the line with Intesection point
      If CheckForIntersectionMatchingwithEndorStartpointLINE(x111, y111, x122, y122, int_X1,
int_Y1) = False And DistanceisLess = False Then

        Call FindBoundary(x111, y111, x122, y122, int_X1, int_Y1, int_x2, int_Y2, FirstPoint,
SecondPoint)

        If FirstPoint = True Then
          ' Divide the line into two
          LineInter.List(LineArccounter + 3) = int_X1
          LineInter.List(LineArccounter + 4) = int_Y1
          Call GeneralLine(int_X1, int_Y1, x122, y122)
          CheckLine1 = True
        End If
      End If

      If CheckLine1 = True Or CheckArc1 = True Then

```

```

LineArccounter = -1
Exit Do
End If

ElseIf Comp > 0.4 Then

    If oncentersec = True Or twointersec = True Then
        If (FindSameArcLine(StartArcx, StartArcy, EndArcx, EndArcy, x111, y111, x122, y122) =
True) Then
            'no need to do anything

        ElseIf (ArcAndLineSame(StartArcx, StartArcy, EndArcx, EndArcy, x111, y111, x122,
y122) = True) Then
            'no need to same
        Else
            Dim firstIn As Boolean
            Dim secondIn As Boolean
            firstIn = False
            secondIn = False
            'For first intersection point matched with line end points
            If (int_X1 = x111 And int_Y1 = y111) Or (int_X1 = x122 And int_Y1 = y122) Then
                If int_X1 = x111 And int_Y1 = y111 Then
                    Call FindBoundary(x111, y111, x122, y122, int_X1, int_Y1, int_x2, int_Y2,
FirstPoint, SecondPoint)

                If FirstPoint = True And SecondPoint = True Then
                    If (ArcOnTheLine(StartArcx, StartArcy, EndArcx, endstarty, x111, y111,
x122, y122) = True) Then
                        'If (int_x2 <> x122) And (int_y2 <> y122) Then
                        LineInter.List(LineArccounter + 3) = int_x2
                        LineInter.List(LineArccounter + 4) = int_Y2

                        LineInter.AddItem ("line")
                        LineInter.AddItem (int_x2)
                        LineInter.AddItem (int_Y2)
                        LineInter.AddItem (x122)
                        LineInter.AddItem (y122)
                        LineArccounter = -1
                        Exit Do
                    Else
                        'call the method that intersection is valid for arc
                        'if the intersection point is not valid then no didvision
                        'line and arc divided by two

                    End If
                End If
            ElseIf int_X1 = x122 And int_Y1 = y122 Then
                Call FindBoundary(x111, y111, x122, y122, int_X1, int_Y1, int_x2, int_Y2,
FirstPoint, SecondPoint)

                If FirstPoint = True And SecondPoint = True Then
                    If (ArcOnTheLine(StartArcx, StartArcy, EndArcx, EndArcy, x111, y111,
x122, y122) = True) Then
                        'If (int_x2 <> x111) Or (int_y2 <> y111) Then
                        LineInter.List(LineArccounter + 3) = int_x2
                        LineInter.List(LineArccounter + 4) = int_Y2

                        LineInter.AddItem ("line")
                        LineInter.AddItem (int_x2)
                        LineInter.AddItem (int_Y2)
                        LineInter.AddItem (x122)
                        LineInter.AddItem (y122)
                        LineArccounter = -1
                        Exit Do
                    Else
                        'for second intersection point matched with the line end points
                        If (int_x2 = x111 And int_Y2 = y111) Or (int_x2 = x122 And int_Y2 = y122) Then
                            If int_x2 = x111 And int_Y2 = y111 Then
                                Call FindBoundary(x111, y111, x122, y122, int_X1, int_Y1, int_x2, int_Y2,
FirstPoint, SecondPoint)

```

```

x122, y122) = True) Then
    If FirstPoint = True And SecondPoint = True Then
        If (ArcOnTheLine(StartArcx, StartArcy, EndArcx, EndArcy, x111, y111,
            'If (int_x2 <> x122) Or (int_y2 <> y122) Then

                LineInter.List(LineArccounter + 3) = int_X1
                LineInter.List(LineArccounter + 4) = int_Y1

                LineInter.AddItem ("line")
                LineInter.AddItem (int_X1)
                LineInter.AddItem (int_Y1)
                LineInter.AddItem (x122)
                LineInter.AddItem (y122)
                LineArccounter = -1
                Exit Do
            End If
        End If
    ElseIf int_x2 = x122 And int_Y2 = y122 Then
        Call FindBoundary(x111, y111, x122, y122, int_X1, int_Y1, int_x2, int_Y2,
            FirstPoint, SecondPoint)

            If FirstPoint = True And SecondPoint = True Then
                If (int_x2 <> x111) Or (int_Y2 <> y111) Then
                    LineInter.List(LineArccounter + 3) = int_X1
                    LineInter.List(LineArccounter + 4) = int_Y1

                    LineInter.AddItem ("line")
                    LineInter.AddItem (int_X1)
                    LineInter.AddItem (int_Y1)
                    LineInter.AddItem (x111)
                    LineInter.AddItem (y111)
                    LineArccounter = -1
                    Exit Do
                End If
            End If
        End If
        secondIn = True
    End If
    If (firstIn = False And secondIn = False) Then
        'this part is for normal intersection point for arc and line
        'call VaildIntersectionPoint
        'Finding out of 4 angle
        firstinterX = int_X1 - CX
        FirstinterY = int_Y1 - CY
        Angle1 = FormatNumber(atan2(firstinterX, FirstinterY), 3)

        firstinterX = int_x2 - CX
        FirstinterY = int_Y2 - CY

        Angle2 = FormatNumber(atan2(firstinterX, FirstinterY), 3)

        firstinterX = StartArcx - CX
        FirstinterY = StartArcy - CY
        StartArcangle = FormatNumber(atan2(firstinterX, FirstinterY), 3)

        firstinterX = EndArcx - CX
        FirstinterY = EndArcy - CY
        EndArcangle = FormatNumber(atan2(firstinterX, FirstinterY), 3)

        TopX = (x111 - CX)
        TopY = (y111 - CY)
        LineTopAngle = FormatNumber(atan2(TopX, TopY), 3)
        EndX = (x122 - CX)
        EndY = (y122 - CY)
        LineEndangle = FormatNumber(atan2(EndX, EndY), 3)

        Call checkAngle(Angle1, Angle2, StartArcangle, EndArcangle, interFirstpoint,
            InterSecondpoint)

        Call FindBoundaryForArc(x111, y111, x122, y122, int_X1, int_Y1, int_x2, int_Y2,
            interFirstpoint, InterSecondpoint)
        'problem when one point of the line is equal with center point of the arc
        If interFirstpoint = True And InterSecondpoint = True Then

```

```
'Checking both intersection point inside of Box
If FirstPoint = True And SecondPoint = True Then
    Call Distance(xI11, yI11, int_X1, int_Y1, int_x2, int_Y2, NearestPoint,
    LongestPoint, Angle1, Angle2, StartArcangle)
    If NearestPoint = True Then
        LineInter.List(LineArccounter + 3) = int_x2
        LineInter.List(LineArccounter + 4) = int_Y2

        LineInter.AddItem "line"
        LineInter.AddItem (int_x2)
        LineInter.AddItem (int_Y2)
        LineInter.AddItem (int_X1)
        LineInter.AddItem (int_Y1)

        LineInter.AddItem "line"
        LineInter.AddItem (int_X1)
        LineInter.AddItem (int_Y1)
        LineInter.AddItem (xI22)
        LineInter.AddItem (yI22)

        LineInter.List(ArcLineCounter + 6) = int_x2
        LineInter.List(ArcLineCounter + 7) = int_Y2

        LineInter.AddItem "arc"
        LineInter.AddItem (CX)
        LineInter.AddItem (CY)
        LineInter.AddItem (R)
        LineInter.AddItem (int_x2)
        LineInter.AddItem (int_Y2)
        LineInter.AddItem (int_X1)
        LineInter.AddItem (int_Y1)

        LineInter.AddItem "arc"
        LineInter.AddItem (CX)
        LineInter.AddItem (CY)
        LineInter.AddItem (R)
        LineInter.AddItem (int_X1)
        LineInter.AddItem (int_Y1)
        LineInter.AddItem (EndArcx)
        LineInter.AddItem (EndArcy)
        LineArccounter = -1
        Exit Do
    ElseIf LongestPoint = True Then
        LineInter.List(LineArccounter + 3) = int_x2
        LineInter.List(LineArccounter + 4) = int_Y2

        LineInter.AddItem "line"
        LineInter.AddItem (int_x2)
        LineInter.AddItem (int_Y2)
        LineInter.AddItem (int_X1)
        LineInter.AddItem (int_Y1)

        LineInter.AddItem "line"
        LineInter.AddItem (int_X1)
        LineInter.AddItem (int_Y1)
        LineInter.AddItem (xI22)
        LineInter.AddItem (yI22)

        LineInter.List(ArcLineCounter + 6) = int_X1
        LineInter.List(ArcLineCounter + 7) = int_Y1

        LineInter.AddItem "arc"
        LineInter.AddItem (CX)
        LineInter.AddItem (CY)
        LineInter.AddItem (R)
        LineInter.AddItem (int_X1)
        LineInter.AddItem (int_Y1)
        LineInter.AddItem (int_x2)
        LineInter.AddItem (int_Y2)

        LineInter.AddItem "arc"
        LineInter.AddItem (CX)
        LineInter.AddItem (CY)
        LineInter.AddItem (R)
        LineInter.AddItem (int_x2)
```



```

LineInter.AddItem (int_Y2)
LineInter.AddItem (EndArcx)
LineInter.AddItem (EndArcy)
LineArccounter = -1
Exit Do
End If
Elseif FirstPoint = True And SecondPoint = False Then
'Else
    Call GetAngle(x111, y111, x122, y122, CX, CY, int_X1, int_Y1, int_x2, int_Y2,
onepoint, otherpoint)

    If onepoint = True Then
        LineInter.List(ArcLineCounter + 6) = int_X1
        LineInter.List(ArcLineCounter + 7) = int_Y1

        LineInter.AddItem "arc"
        LineInter.AddItem (CX)
        LineInter.AddItem (CY)
        LineInter.AddItem (R)
        LineInter.AddItem (int_X1)
        LineInter.AddItem (int_Y1)
        LineInter.AddItem (EndArcx)
        LineInter.AddItem (EndArcy)

        LineInter.List(LineArccounter + 3) = int_X1
        LineInter.List(LineArccounter + 4) = int_Y1

        LineInter.AddItem "line"
        LineInter.AddItem int_X1
        LineInter.AddItem int_Y1
        LineInter.AddItem (x122)
        LineInter.AddItem (y122)

        LineArccounter = -1
        Exit Do
    End If
    Elseif FirstPoint = False And SecondPoint = True Then
        Call GetAngle(x111, y111, x122, y122, CX, CY, int_X1, int_Y1, int_x2, int_Y2,
onepoint, otherpoint)

        If otherpoint = True Then

            LineInter.List(ArcLineCounter + 6) = int_x2
            LineInter.List(ArcLineCounter + 7) = int_Y2

            LineInter.AddItem "arc"
            LineInter.AddItem (CX)
            LineInter.AddItem (CY)
            LineInter.AddItem (R)
            LineInter.AddItem (int_x2)
            LineInter.AddItem (int_Y2)
            LineInter.AddItem (EndArcx)
            LineInter.AddItem (EndArcy)

            LineInter.List(LineArccounter + 3) = int_x2
            LineInter.List(LineArccounter + 4) = int_Y2

            LineInter.AddItem "line"
            LineInter.AddItem int_x2
            LineInter.AddItem int_Y2
            LineInter.AddItem (x122)
            LineInter.AddItem (y122)

            LineArccounter = -1
            Exit Do
        End If
    End If

    Elseif interFirstpoint = True And InterSecondpoint = False Then
        LineInter.List(ArcLineCounter + 6) = int_X1
        LineInter.List(ArcLineCounter + 7) = int_Y1

        LineInter.AddItem "arc"
        LineInter.AddItem (CX)
        LineInter.AddItem (CY)
        LineInter.AddItem (R)

```

```

LineInter.AddItem (int_X1)
LineInter.AddItem (int_Y1)
LineInter.AddItem (EndArcx)
LineInter.AddItem (EndArcy)

LineInter.List(LineArccounter + 3) = int_X1
LineInter.List(LineArccounter + 4) = int_Y1

LineInter.AddItem "line"
LineInter.AddItem int_X1
LineInter.AddItem int_Y1
LineInter.AddItem (x122)
LineInter.AddItem (y122)

LineArccounter = -1
Exit Do
ElseIf interFirstpoint = False And InterSecondpoint = True Then
LineInter.List(ArcLineCounter + 6) = int_x2
LineInter.List(ArcLineCounter + 7) = int_Y2

LineInter.AddItem "arc"
LineInter.AddItem (CX)
LineInter.AddItem (CY)
LineInter.AddItem (R)
LineInter.AddItem (int_x2)
LineInter.AddItem (int_Y2)
LineInter.AddItem (EndArcx)
LineInter.AddItem (EndArcy)

LineInter.List(LineArccounter + 3) = int_x2
LineInter.List(LineArccounter + 4) = int_Y2

LineInter.AddItem "line"
LineInter.AddItem int_x2
LineInter.AddItem int_Y2
LineInter.AddItem (x122)
LineInter.AddItem (y122)

LineArccounter = -1
Exit Do
End If
End If
End If
End If
End If
End If

ArcLineCounter = ArcLineCounter + 1
Loop
End If

ArcLineCounter = 0

LineArccounter = LineArccounter + 1
Loop
'Circle Circle Intersection
Dim Circlecounter As Integer, CXCircle1 As Single, CYCircle1 As Single, RCircle1 As Single
Dim Circle_CircleCounter As Integer, CxCircle2 As Single, CYCircle2 As Single, RCircle2 As Single
Dim FirstAngleCircle As Single, SecondAngleCircle As Single, FirstAngleCircle2nd As Single, SecondAngleCircle2nd As
Single, TotalDistance As Single
Circlecounter = 0
Circle_CircleCounter = 0
Do While Circlecounter <= LineInter.ListCount

If LineInter.List(Circlecounter) = "circle" Then
CXCircle1 = Val(LineInter.List(Circlecounter + 1))
CYCircle1 = Val(LineInter.List(Circlecounter + 2))
RCircle1 = Val(LineInter.List(Circlecounter + 3))

'For k = 0 To LineInter.ListCount
Do While (Circle_CircleCounter + 2) <= LineInter.ListCount
Circle_CircleCounter = Circle_CircleCounter + 1
If LineInter.List(Circle_CircleCounter) = "circle" Then

```

```

CxCircle2 = Val(LineInter.List(Circle_CircleCounter + 1))
CYCircle2 = Val(LineInter.List(Circle_CircleCounter + 2))
RCircle2 = Val(LineInter.List(Circle_CircleCounter + 3))
TotalDistance = Abs(Sqr((CXCicle1 - CxCircle2) ^ 2 + (CYCircle1 - CYCircle2) ^ 2))
If TotalDistance = 0 Then

ElseIf TotalDistance <= (RCircle1 + RCircle2) Then
Call ArcCircleInter(CXCicle1, CYCircle1, RCircle1, CxCircle2, CYCircle2, RCircle2, int_X1,
int_Y1, int_x2, int_Y2)

'Finding out Angle for the first circle
firstinterX = int_X1 - CXCicle1
FirstinterY = int_Y1 - CYCircle1
FirstAngleCircle = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = int_x2 - CXCicle1
FirstinterY = int_Y2 - CYCircle1
SecondAngleCircle = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

'Finding out Angle for the 2nd circle
firstinterX = int_X1 - CxCircle2
FirstinterY = int_Y1 - CYCircle2
FirstAngleCircle2nd = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = int_x2 - CxCircle2
FirstinterY = int_Y2 - CYCircle2
SecondAngleCircle2nd = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))
'Making Arc by Dividing first circle in to two arc
If FirstAngleCircle < SecondAngleCircle Then

    Call GeneralARC(CXCicle1, CYCircle1, RCircle1, int_X1, int_Y1, int_x2, int_Y2)
    Call GeneralARC(CXCicle1, CYCircle1, RCircle1, int_x2, int_Y2, int_X1, int_Y1)
    LineInter.List(Circlecounter) = "circle1"
ElseIf FirstAngleCircle > SecondAngleCircle Then
    Call GeneralARC(CXCicle1, CYCircle1, RCircle1, int_x2, int_Y2, int_X1, int_Y1)
    Call GeneralARC(CXCicle1, CYCircle1, RCircle1, int_X1, int_Y1, int_x2, int_Y2)
    LineInter.List(Circlecounter) = "circle1"
End If
'Making Arc by Dividing 2nd Circle into two Arc
If FirstAngleCircle2nd < SecondAngleCircle2nd Then

    Call GeneralARC(CxCircle2, CYCircle2, RCircle2, int_X1, int_Y1, int_x2, int_Y2)
    Call GeneralARC(CxCircle2, CYCircle2, RCircle2, int_x2, int_Y2, int_X1, int_Y1)
    LineInter.List(Circle_CircleCounter) = "circle2"
ElseIf FirstAngleCircle2nd > SecondAngleCircle2nd Then
    Call GeneralARC(CxCircle2, CYCircle2, RCircle2, int_x2, int_Y2, int_X1, int_Y1)
    Call GeneralARC(CxCircle2, CYCircle2, RCircle2, int_X1, int_Y1, int_x2, int_Y2)
    LineInter.List(Circle_CircleCounter) = "circle2"
End If
Exit Do
End If
Loop
End If
Circlecounter = Circlecounter + 1
Loop

'ARC Circle intersection
Dim ArcCirclecounter As Integer, CircleArcCounter As Integer, RARC As Single, Rcircle As Single, CXARC As Single,
CYARC As Single
Dim CxCircle As Single
Dim SXarc As Single, SYarc As Single, EXarc As Single, EYarc As Single
Dim Onetrue As Boolean, Othertrue As Boolean, Secondonetrue As Boolean, Secondothertrue As Boolean, OneWay As
Boolean, OtherWay As Boolean
Dim CircleAngle1 As Single, CircleAngle2 As Single
Dim FirstInterX1 As Boolean, FirstIntersecBetweenStartEndangle As Boolean, _
SecondInterX1 As Boolean, SecondIntersecBetweenStartEndAngle As Boolean
Dim Angle2X As Single, Angle2Y As Single, Angle3X As Single, Angle3Y As Single, Inside As Boolean
Dim OneAngle2 As Boolean, OtherAngle3 As Boolean, NormalIntersection As Boolean
Dim CyCircle As Single
ArcCirclecounter = 0
CircleArcCounter = 0
Dim StartAngle As Single, EndAngle As Single, FirstIntersecAngle As Single, SecondIntersecAngle As Single
Dim LAngle1 As Single, LAngle2 As Single, LAngle3 As Single, LAngle4 As Single, XfromAngle As Single, YfromAngle As
Single, XXfromAngle As Single, YYfromAngle As Single

```

```

Do While CircleArcCounter <= LineInter.ListCount

    If LineInter.List(CircleArcCounter) = "circle" Then
        CxCircle = Val(LineInter.List(CircleArcCounter + 1))
        CyCircle = Val(LineInter.List(CircleArcCounter + 2))
        Rcircle = Val(LineInter.List(CircleArcCounter + 3))

        'For k = 0 To LineInter.ListCount

        Do While ArcCirclecounter <= LineInter.ListCount
            If LineInter.List(ArcCirclecounter) = "arc" Then

                CXARC = Val(LineInter.List(ArcCirclecounter + 1))
                CYARC = Val(LineInter.List(ArcCirclecounter + 2))
                RARC = Val(LineInter.List(ArcCirclecounter + 3))
                SXarc = Val(LineInter.List(ArcCirclecounter + 4))
                SYarc = Val(LineInter.List(ArcCirclecounter + 5))
                EXarc = Val(LineInter.List(ArcCirclecounter + 6))
                EYarc = Val(LineInter.List(ArcCirclecounter + 7))
                'Checking is both circle intersec each other or they only touch eachother.
                'If they touch each other at one point then the summation of two radious will be same the distance
of two center.

                TotalDistance = Abs(Sqr((CXARC - CxCircle) ^ 2 + (CYARC - CyCircle) ^ 2))

                If CxCircle = CXARC And CyCircle = CYARC Then

                    ElseIf TotalDistance = (RARC + Rcircle) Then 'they are tangent
                        Call ArcCircleInter(CXARC, CYARC, RARC, CxCircle, CyCircle, Rcircle, int_X1, int_Y1,
int_x2, int_Y2)

                        'Divid the Arc into two and convert the Circle into one ARC
                        LineInter.AddItem "arc"
                        LineInter.AddItem (CxCircle)
                        LineInter.AddItem (CyCircle)
                        ineInter.AddItem (Rcircle)
                        LineInter.AddItem (int_X1)
                        LineInter.AddItem (int_Y1)
                        LineInter.AddItem (int_X1)
                        LineInter.AddItem (int_Y1)

                        LineInter.List(ArcCirclecounter + 6) = int_X1
                        LineInter.List(ArcCirclecounter + 7) = int_Y1

                        LineInter.AddItem "arc"
                        LineInter.AddItem (CXARC)
                        LineInter.AddItem (CYARC)
                        ineInter.AddItem (RARC)
                        LineInter.AddItem (int_X1)
                        LineInter.AddItem (int_Y1)
                        LineInter.AddItem (EXarc)
                        LineInter.AddItem (EYarc)

                        LineInter.List(CircleCircleCounter) = "circleI"
                        CircleArcCounter = -1
                        Exit Do

                    ElseIf TotalDistance < (RARC + Rcircle) Then
                        Call ArcCircleInter(CXARC, CYARC, RARC, CxCircle, CyCircle, Rcircle, int_X1, int_Y1,
int_x2, int_Y2)

                        'Comp, _
                        oneintersec, twointersec)

                        '(onetrue "int_x1=(SXarc & SYarc)",(othertrue("int_x2=(EXarc & EYarc)"))
                        'Secondonetrue "int_x2=(SXarc & SYarc)",secondothertrue,int_x1=(EXarc & EYarc)

                        'for Universel Angle
                        firstinterX = SXarc - CXARC
                        FirstinterY = SYarc - CYARC

```

```

Angle1 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = int_X1 - CXARC
FirstinterY = int_Y1 - CYARC
Angle2 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))
Angle2X = Val(int_X1)
Angle2Y = Val(int_Y1)

firstinterX = int_x2 - CXARC
FirstinterY = int_Y2 - CYARC
Angle3 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))
Angle3X = Val(int_x2)
Angle3Y = Val(int_Y2)

firstinterX = EXarc - CXARC
FirstinterY = EYarc - CYARC
Angle4 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = int_X1 - CxCircle
FirstinterY = int_Y1 - CyCircle
CircleAngle1 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = int_x2 - CxCircle
FirstinterY = int_Y2 - CyCircle
CircleAngle2 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

point
'comparing both endpoint of the Arc with both intersection point. Startpoint of the Arc or End
'of the Arc might be the same with the intersection point.
Call FindSameArcCircle(SXarc, SYarc, EXarc, EYarc, int_X1, int_Y1, int_x2, int_Y2, Onetrue,
Secondonetrue)
'Call LocalAxes(Angle1, Angle2, Angle3, Angle4, FirstInterX1,
FirstIntersecBetweenStartEndangle, _
SecondInterX1, SecondIntersecBetweenStartEndAngle, CircleAngle1, CircleAngle2, _
startangle, Endangle, FirstIntersecAngle, SecondIntersecAngle)

If Onetrue = True And Secondonetrue = True Then
'divide the circle into two Arc keep the arc as it is.
LineInter.AddItem "arc"
LineInter.AddItem (CxCircle)
LineInter.AddItem (CyCircle)
LineInter.AddItem (Rcircle)
LineInter.AddItem (SXarc)
LineInter.AddItem (SYarc)
LineInter.AddItem (EXarc)
LineInter.AddItem (EYarc)

LineInter.AddItem "arc"
LineInter.AddItem (CxCircle)
LineInter.AddItem (CyCircle)
LineInter.AddItem (Rcircle)
LineInter.AddItem (EXarc)
LineInter.AddItem (EYarc)
LineInter.AddItem (SXarc)
LineInter.AddItem (SYarc)
CircleArcCounter = -1
Exit Do
ElseIf Onetrue = True Or Secondonetrue = True Then

If Angle2X = SXarc And Angle2Y = SYarc Then
'check angle3 either inside the circle or outside the circle
'call a function the name is FindArcTipOutsideOrinside
Call FindArcTipOutsideOrinsideCircle(Angle1, Angle3, Angle4, Inside)
'if Inside=True thats means Arc did not intersec the circle
If Inside = True Then 'do not divide the Arc but circle Convert it into 1 Arc

Call DivideCircleStartpointArc(CxCircle, CyCircle, Rcircle, SXarc, SYarc)
LineInter.List(CircleArcCounter) = "circle1"
CircleArcCounter = -1
Exit Do
Else 'Divide the Arc into 2Arc and divide the Circle into 2 Arc

```

```

YfromAngle)
    Call FindXYfromAngle(Angle3, CXARC, CYARC, RARC, XfromAngle,
    LineInter.List(ArcCirclecounter + 6) = XfromAngle
    LineInter.List(ArcCirclecounter + 7) = YfromAngle
    'There will be 2 intersection but One intersection point will be same with startpoint or
endpoint of Arc
    'Remember when startpoint and End point of the Arc is same with one intersection
point
    Call ArcCircleBothintersecOnepointSameStart(Angle3, CXARC, CYARC, RARC,
SXarc, SYarc, EXarc, EYarc, CxCircle, CyCircle, Rcircle)
    LineInter.List(CircleArcCounter) = "circle1"
    CircleArcCounter = -1
    Exit Do
End If
'Now we will consider End point of the Arc
Elseif Angle2X = EXarc And Angle2Y = EYarc Then
'check angle3
    Call FindArcTipOutsideOrinsideCircle(Angle1, Angle3, Angle4, Inside)
    If Inside = True Then
        Call DivideCircleEndpointArc(CxCircle, CyCircle, Rcircle, EXarc, EYarc)
        LineInter.List(CircleArcCounter) = "circle1"
        CircleArcCounter = -1
        Exit Do
    Else
        'Divide the Arc into 2 Arc and divide the Circle into 2Arc
        Call FindXYfromAngle(Angle3, CXARC, CYARC, RARC, XfromAngle, YfromAngle)
        LineInter.List(ArcCirclecounter + 6) = XfromAngle
        LineInter.List(ArcCirclecounter + 7) = YfromAngle
        Call ArcCircleBothintersecOnepointSameEnd(Angle3, CXARC, CYARC, RARC,
SXarc, SYarc, EXarc, EYarc, CxCircle, CyCircle, Rcircle)
    End If

Elseif Angle3X = SXarc And Angle3Y = SYarc Then
'check angle2
'check angle3 either inside the circle or outside the circle
'call a function the name is FindArcTipOutsideOrinside
    Call FindArcTipOutsideOrinsideCircle(Angle1, Angle2, Angle4, Inside)
    'if Inside=True thats means Arc did not intersec the circle
    If Inside = True Then 'do not divide the Arc but circle Convert it into 1 Arc

        Call DivideCircleStartpointArc(CxCircle, CyCircle, Rcircle, SXarc, SYarc)
        LineInter.List(CircleArcCounter) = "circle1"
        CircleArcCounter = -1
        Exit Do
    Else 'Divide the Arc into 2Arc and divide the Circle into 2 Arc
        Call FindXYfromAngle(Angle2, CXARC, CYARC, RARC, XfromAngle,
YfromAngle)

        LineInter.List(ArcCirclecounter + 6) = XfromAngle
        LineInter.List(ArcCirclecounter + 7) = YfromAngle
        'There will be 2 intersection but One intersection point will be same with startpoint or
endpoint of Arc
        'Remember when startpoint and End point of the Arc is same with one intersection
point
        Call ArcCircleBothintersecOnepointSameStart(Angle2, CXARC, CYARC, RARC,
SXarc, SYarc, EXarc, EYarc, CxCircle, CyCircle, Rcircle)
        LineInter.List(CircleArcCounter) = "circle1"
        CircleArcCounter = -1
        Exit Do
    End If

Elseif Angle3X = EXarc And Angle3Y = EYarc Then
'check angle2
    Call FindArcTipOutsideOrinsideCircle(Angle1, Angle2, Angle4, Inside)
    If Inside = True Then
        Call DivideCircleEndpointArc(CxCircle, CyCircle, Rcircle, EXarc, EYarc)
        LineInter.List(CircleArcCounter) = "circle1"
        CircleArcCounter = -1
        Exit Do
    Else
        'Divide the Arc into 2 Arc and divide the Circle into 2Arc
        Call FindXYfromAngle(Angle2, CXARC, CYARC, RARC, XfromAngle,
YfromAngle)

        LineInter.List(ArcCirclecounter + 6) = XfromAngle
        LineInter.List(ArcCirclecounter + 7) = YfromAngle

```

```

        Call ArcCircleBothintersecOnepointSameEnd(Angle2, CXARC, CYARC, RARC,
SXArc, SYArc, EXArc, EYArc, CxCircle, CyCircle, Rcircle)
        CircleArcCounter = -1
        Exit Do
    End If

End If

ElseIf (Angle1 = Angle4) Then
    Call LocalAxes(Angle1, Angle2, Angle3, Angle4, FirstInterX1,
FirstInterSecBetweenStartEndangle, _
        SecondInterX1, SecondInterSecBetweenStartEndAngle, CircleAngle1, CircleAngle2, _
        StartAngle, EndAngle, FirstIntersecAngle, SecondIntersecAngle)
    'Divide the Arc into three Arc
    If FirstInterX1 = True Then
        LineInter.List(ArcCirclecounter + 6) = int_X1
        LineInter.List(ArcCirclecounter + 7) = int_Y1
        'Call GeneralARC(CXARC, CYARC, RARC, SXArc, SYArc, int_x1, int_y1)
        Call GeneralARC(CXARC, CYARC, RARC, int_X1, int_Y1, int_x2, int_Y2)
        Call GeneralARC(CXARC, CYARC, RARC, int_x2, int_Y2, EXArc, EYArc)
    ElseIf FirstInterX1 = False Then
        LineInter.List(ArcCirclecounter + 6) = int_x2
        LineInter.List(ArcCirclecounter + 7) = int_Y2
        Call GeneralARC(CXARC, CYARC, RARC, SXArc, SYArc, int_x2, int_Y2)
        Call GeneralARC(CXARC, CYARC, RARC, int_x2, int_Y2, int_X1, int_Y1)
        Call GeneralARC(CXARC, CYARC, RARC, int_X1, int_Y1, EXArc, EYArc)
    End If
    'For circle
    If CircleAngle1 < CircleAngle2 Then
        Call GeneralARC(CxCircle, CyCircle, Rcircle, int_X1, int_Y1, int_x2, int_Y2)
        Call GeneralARC(CxCircle, CyCircle, Rcircle, int_x2, int_Y2, int_X1, int_Y1)
    ElseIf CircleAngle1 > CircleAngle2 Then
        Call GeneralARC(CxCircle, CyCircle, Rcircle, int_x2, int_Y2, int_X1, int_Y1)
        Call GeneralARC(CxCircle, CyCircle, Rcircle, int_X1, int_Y1, int_x2, int_Y2)
    End If
    LineInter.List(CircleArcCounter) = "circle1"
    CircleArcCounter = -1
    Exit Do
Else
    'Check first angle between Start point or End point
    Call StartpointInsideorRoutside(Angle1, Angle2, Angle3, Angle4, OneAngle2,
OtherAngle3, NormalIntersection)
    If OneAngle2 = True And OtherAngle3 = True Then
        'Arc and Circle has two intersections
        'Call NormalOrAbnormalIntersection(Angle1, Angle2, Angle3, Angle4,
NormalIntersection)
        If NormalIntersection = True Then
            'Call findXYfromAngle(Angle2, CXARC, CYARC, RARC, XfromAngle,
YfromAngle)
            'Divide the Arc into 3 Arc
            LineInter.List(ArcCirclecounter + 6) = int_X1
            LineInter.List(ArcCirclecounter + 7) = int_Y1
            Call GeneralARC(CXARC, CYARC, RARC, int_X1, int_Y1, int_x2, int_Y2)
            Call GeneralARC(CXARC, CYARC, RARC, int_x2, int_Y2, EXArc, EYArc)
            Call GeneralARC(CxCircle, CyCircle, Rcircle, int_X1, int_Y1, int_x2, int_Y2)
            Call GeneralARC(CxCircle, CyCircle, Rcircle, int_x2, int_Y2, int_X1, int_Y1)
            LineInter.List(CircleArcCounter) = "circle1"
            CircleArcCounter = -1
            Exit Do
        ElseIf NormalIntersection = False Then
            'Divide the Arc into 3 Arc
            LineInter.List(ArcCirclecounter + 6) = int_x2
            LineInter.List(ArcCirclecounter + 7) = int_Y2
            Call GeneralARC(CXARC, CYARC, RARC, int_x2, int_Y2, int_X1, int_Y1)
            Call GeneralARC(CXARC, CYARC, RARC, int_X1, int_Y1, EXArc, EYArc)
            Call GeneralARC(CxCircle, CyCircle, Rcircle, int_x2, int_Y2, int_X1, int_Y1)
            Call GeneralARC(CxCircle, CyCircle, Rcircle, int_X1, int_Y1, int_x2, int_Y2)
            LineInter.List(CircleArcCounter) = "circle1"
            CircleArcCounter = -1
            Exit Do
        End If
    ElseIf OneAngle2 = True And OtherAngle3 = False Then
        'The Arc has one intersecion that is Angle2 and Circle has one intersecion at Angle2
        'Call findXYfromAngle(Angle2, CXARC, CYARC, RARC, XfromAngle,
YfromAngle)

```

```

LineInter.List(ArcCirclecounter + 6) = int_X1
LineInter.List(ArcCirclecounter + 7) = int_Y1

EXArc, EYarc)

'Call DivideArcAnglePoint(CXARC, CYARC, RARC, XfromAngle, YfromAngle,

'Divide the circle where Startpoint and End point same
'Call DivideCircleAnglePoint(CxCircle, CyCircle, Rcircle, XfromAngle, YfromAngle)

Call GeneralARC(CXARC, CYARC, RARC, int_X1, int_Y1, EXArc, EYarc)
Call GeneralARC(CxCircle, CyCircle, Rcircle, int_X1, int_Y1, int_X1, int_Y1)
LineInter.List(CircleArcCounter) = "circle1"
CircleArcCounter = -1
Exit Do

ElseIf OneAngle2 = False And OtherAngle3 = True Then
'The arc has one intersestion that is Angle3 and Circle has one ontersestion at Angle3
'Call findXYfromAngle(Angle3, CXARC, CYARC, RARC, XfromAngle,

YfromAngle)

LineInter.List(ArcCirclecounter + 6) = int_x2
LineInter.List(ArcCirclecounter + 7) = int_Y2

EXArc, EYarc)

'Call DivideArcAnglePoint(CXARC, CYARC, RARC, XfromAngle, YfromAngle,

'Call DivideCircleAnglePoint(CxCircle, CyCircle, Rcircle, XfromAngle, YfromAngle)

Call GeneralARC(CXARC, CYARC, RARC, int_x2, int_Y2, EXArc, EYarc)
Call GeneralARC(CxCircle, CyCircle, Rcircle, int_x2, int_Y2, int_x2, int_Y2)
LineInter.List(CircleArcCounter) = "circle1"
CircleArcCounter = -1
Exit Do
End If

End If
End If

End If
ArcCirclecounter = ArcCirclecounter + 1
Loop

End If

CircleArcCounter = CircleArcCounter + 1
Loop
'Arc Arc intersection
Dim Arccounter As Integer, CXArc1 As Single, CYArc1 As Single, RArc1 As Single, SXArc1 As Single, SYArc1 As Single,
EXArc1 As Single, EYArc1 As Single
Dim ArcArcCounter As Integer, CXArc2 As Single, CYArc2 As Single, RArc2 As Single, SXArc2 As Single, SYArc2 As
Single, EXArc2 As Single, EYArc2 As Single
Dim OneArcTrue As Boolean, SecondArcTrue As Boolean, StartpointArc1 As Boolean, endpointArc1 As Boolean,
StartpointArc2 As Boolean, endpointArc2 As Boolean
Dim StartAngleArc2 As Single, EndAngleArc2 As Single, FirstIntersecAngleArc2 As Single, SecondIntersecAngleArc2 As
Single
Dim FirstInterX1Arc2 As Boolean, FirstIntersecBetweenStartEndangleArc2 As Boolean
Dim SecondInterX1Arc2 As Boolean, SecondIntersecBetweenStartEndAngleArc2 As Boolean, CircleAngle1Arc2 As Single,
CircleAngle2Arc2 As Single
Dim Angle1Arc2 As Single, Angle2Arc2 As Single, Angle3Arc2 As Single, Angle4Arc2 As Single, TotalDisArc Arc As
Single
Dim StartPointArc2cenTer1X As Single, StartpointArc2cenTer1Y As Single, StartpointArc2cenTerArc1Angle As Single
Dim EndpointArc2cenTer1X As Single, EndpointArc2cenTer1Y As Single, EndpointArc2cenTerArc1Angle As Single
Dim SumR As Single, OnepointSame As Boolean, OtherpointSame As Boolean, CheckAngle2Arc1 As Boolean,
CheckAngle3Arc1 As Boolean, CheckAngle2Arc2 As Boolean
Dim CheckAngle3Arc2 As Boolean, Intersec1 As Boolean
Dim CheckAngle1Arc1First As Boolean, CheckAngle1Arc2First As Boolean, Satisfied As Boolean
Dim CompareAngle As Single, ForcompareangleX As Single, ForcompareangleY As Single
Dim BotharcSpointEpointsame As Boolean
ArcArcCounter = 0
Arccounter = 0
Dim TwoIntersection As Boolean, OneIntersection As Boolean
Dim Satisfied2 As Boolean, Satisfied3 As Boolean, Satisfied4 As Boolean, AllpointSame As Boolean,
OnepointCheck As Boolean
Dim ExitArc1 As Boolean, ExitArc2 As Boolean
TwoIntersection = False
OneIntersection = False

```



```

Do While ArcCounter <= LineInter.ListCount

    If LineInter.List(ArcCounter) = "arc" Then
        CXArc1 = FormatNumber(Val(LineInter.List(ArcCounter + 1)), 3)
        CYArc1 = FormatNumber(Val(LineInter.List(ArcCounter + 2)), 3)
        RArc1 = FormatNumber(Val(LineInter.List(ArcCounter + 3)), 3)
        SXXArc1 = FormatNumber(Val(LineInter.List(ArcCounter + 4)), 3)
        SYArc1 = FormatNumber(Val(LineInter.List(ArcCounter + 5)), 3)
        EXArc1 = FormatNumber(Val(LineInter.List(ArcCounter + 6)), 3)
        EYArc1 = FormatNumber(Val(LineInter.List(ArcCounter + 7)), 3)

        ArcArcCounter = 0
        'For k = 0 To LineInter.ListCount
        Do While (ArcArcCounter + 8) <= LineInter.ListCount
            ArcArcCounter = ArcArcCounter + 1
            If LineInter.List(ArcArcCounter) = "arc" Then
                CXArc2 = FormatNumber(Val(LineInter.List(ArcArcCounter + 1)), 3)
                CYArc2 = FormatNumber(Val(LineInter.List(ArcArcCounter + 2)), 3)
                RArc2 = FormatNumber(Val(LineInter.List(ArcArcCounter + 3)), 3)
                SXXArc2 = FormatNumber(Val(LineInter.List(ArcArcCounter + 4)), 3)
                SYArc2 = FormatNumber(Val(LineInter.List(ArcArcCounter + 5)), 3)
                EXArc2 = FormatNumber(Val(LineInter.List(ArcArcCounter + 6)), 3)
                EYArc2 = FormatNumber(Val(LineInter.List(ArcArcCounter + 7)), 3)

                CheckAngle2Arc1 = False
                CheckAngle3Arc1 = False
                CheckAngle2Arc2 = False
                CheckAngle3Arc2 = False
                BothArcSpointEpoinstame = False
                TotalDisArcArc = Abs(Sqr((CXArc1 - CXArc2) ^ 2 + (CYArc1 - CYArc2) ^ 2))
                SumR = RArc1 + RArc2
                DifferenceRadiouArc = Abs(TotalDisArcArc - SumR)
                Call ArcBothEndpointChecking(SXXArc1, SYArc1, EXArc1, EYArc1, SXXArc2, SYArc2, EXArc2,
                EYArc2, AllpointSame, OnepointCheck)
                If (CXArc1 = CXArc2 And CYArc1 = CYArc2) Then
                    ElseIf TotalDisArcArc = 0 Then
                        'both Arc has intersec at at one point thats mean they arc in tangent

                    ElseIf DifferenceRadiouArc < 3 Then
                        If (AllpointSame = False Or OnepointCheck = False) Then
                            'MsgBox "no need to check"
                            BothArcSpointEpoinstame = True
                        End If
                        If BothArcSpointEpoinstame = False Then
                            Call ArcCircleInter(CXArc1, CYArc1, RArc1, CXArc2, CYArc2, RArc2, int_X1, int_Y1,
                            int_x2, int_Y2)

                            'Divide arc1 into 2 Arc and divide the arc2 into 2 arc
                            LineInter.List(ArcCounter + 6) = int_X1
                            LineInter.List(ArcCounter + 7) = int_Y1

                            LineInter.AddItem "arc"
                            LineInter.AddItem (CXArc1)
                            LineInter.AddItem (CYArc1)
                            LineInter.AddItem (RArc1)
                            LineInter.AddItem (int_X1)
                            LineInter.AddItem (int_Y1)
                            LineInter.AddItem (EXArc1)
                            LineInter.AddItem (EYArc1)

                            LineInter.List(ArcArcCounter + 6) = int_X1
                            LineInter.List(ArcArcCounter + 7) = int_Y1

                            LineInter.AddItem "arc"
                            LineInter.AddItem (CXArc2)
                            LineInter.AddItem (CYArc2)
                            LineInter.AddItem (RArc2)
                            LineInter.AddItem (int_X1)
                            LineInter.AddItem (int_Y1)
                            LineInter.AddItem (EXArc2)
                            LineInter.AddItem (EYArc2)
                            ArcCounter = -1
                            Exit Do
                        End If
                    End If
                End If
            End If
        End While
    End While

```

```

ElseIf TotalDisArcArc < SumR And TotalDisArcArc > 0 Then
Call ArcCircleInter(CXArc1, CYArc1, RArc1, CXArc2, CYArc2, RArc2, int_X1, int_Y1, int_x2,
int_Y2)

'Findingout ANGLE for the ARC number 1

firstinterX = SXArc1 - CXArc1
FirstinterY = SYArc1 - CYArc1
Angle1 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = int_X1 - CXArc1
FirstinterY = int_Y1 - CYArc1
Angle2 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))
Angle2X = Val(int_X1)
Angle2Y = Val(int_Y1)

firstinterX = int_x2 - CXArc1
FirstinterY = int_Y2 - CYArc1
Angle3 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))
Angle3X = Val(int_x2)
Angle3Y = Val(int_Y2)

firstinterX = EXArc1 - CXArc1
FirstinterY = EYArc1 - CYArc1
Angle4 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

'Call LocalAxes(Angle1, Angle2, Angle3, Angle4, FirstInterX1,
FirstIntersecBetweenStartEndangle, _
SecondInterX1, SecondIntersecBetweenStartEndAngle, CircleAngle1, CircleAngle2, _
startangle, Endangle, FirstIntersecAngle, SecondIntersecAngle)

'Findingout ANGLE for the ARC number 2

firstinterX = SXArc2 - CXArc2
FirstinterY = SYArc2 - CYArc2
Angle1Arc2 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = int_X1 - CXArc2
FirstinterY = int_Y1 - CYArc2
Angle2Arc2 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = int_x2 - CXArc2
FirstinterY = int_Y2 - CYArc2
Angle3Arc2 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = EXArc2 - CXArc2
FirstinterY = EYArc2 - CYArc2
Angle4Arc2 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

'Call LocalAxes(Angle1Arc2, Angle2Arc2, Angle3Arc2, Angle4Arc2, FirstInterX1Arc2,
FirstIntersecBetweenStartEndangleArc2, _
SecondInterX1Arc2, SecondIntersecBetweenStartEndAngleArc2, CircleAngle1Arc2,
CircleAngle2Arc2, _
StartAngleArc2, EndAngleArc2, FirstIntersecAngleArc2, SecondIntersecAngleArc2)

'Consider Int_x1 for the Arc1
'calling For Arc1
Call Arc1SatisfiedByAngle(Angle1, Angle2, Angle3, Angle4, CheckAngle2Arc1,
CheckAngle3Arc1, CheckAngle1Arc1First)
'Calling for Arc2
Call Arc1SatisfiedByAngle(Angle1Arc2, Angle2Arc2, Angle3Arc2, Angle4Arc2, _
CheckAngle2Arc2, CheckAngle3Arc2, CheckAngle1Arc2First)
***For Arc1 Start
'If (int_x1 = SXArc1 And int_y1 = SYArc1) Or (int_x1 = EXArc1 And int_y1 = EYArc1)
Then

'no Need to divide the Arc

If (CheckAngle2Arc1 = True And CheckAngle3Arc1 = True) And (CheckAngle2Arc2 = True
And CheckAngle3Arc2 = True) Then
'For Arc1

```

```

If CheckAngle1Arc1First = True Then 'Normal division for Arc1 like
Start,Angle2,Angle3,Angle4
    Call ArcBothEndpointChecking(SXArc1, SYArc1, EXArc1, EYArc1, SXArc2, SYArc2,
EXArc2, EYArc2, AllpointSame, OnepointCheck)
    If AllpointSame = True And OnepointCheck = True Then 'no need to divide the Arc
    ElseIf AllpointSame = False And OnepointCheck = True Then 'divide the Arc into two
        LineInter.List(ArcCounter + 6) = int_X1
        LineInter.List(ArcCounter + 7) = int_Y1
        Call GeneralARC(CXArc1, CYArc1, RArc1, int_X1, int_Y1, EXArc1, EYArc1)
        ExitArc1 = True
    Else 'Divide the Arc into three
        LineInter.List(ArcCounter + 6) = int_X1
        LineInter.List(ArcCounter + 7) = int_Y1
        Call GeneralARC(CXArc1, CYArc1, RArc1, int_X1, int_Y1, int_x2, int_Y2)
        Call GeneralARC(CXArc1, CYArc1, RArc1, int_x2, int_Y2, EXArc1, EYArc1)
        ExitArc1 = True
    End If
Else 'Not Normal division for Arc1 like Start,Angle3 ,Angle2,Angle4
    Call ArcBothEndpointChecking(SXArc1, SYArc1, EXArc1, EYArc1, SXArc2, SYArc2,
EXArc2, EYArc2, AllpointSame, OnepointCheck)
    If AllpointSame = True And OnepointCheck = True Then 'no need to divide the Arc
    ElseIf AllpointSame = False And OnepointCheck = True Then 'divide the Arc into two
        LineInter.List(ArcCounter + 6) = int_x2
        LineInter.List(ArcCounter + 7) = int_Y2
        Call GeneralARC(CXArc1, CYArc1, RArc1, int_x2, int_Y2, EXArc1, EYArc1)
        ExitArc1 = True
    Else
        LineInter.List(ArcCounter + 6) = int_x2
        LineInter.List(ArcCounter + 7) = int_Y2
        Call GeneralARC(CXArc1, CYArc1, RArc1, int_x2, int_Y2, int_X1, int_Y1)
        Call GeneralARC(CXArc1, CYArc1, RArc1, int_X1, int_Y1, EXArc1, EYArc1)
        ExitArc1 = True
    End If
End If
'For Arc2
If CheckAngle1Arc2First = True Then 'Normal Division for Arc2 like
Start,Angle2Arc2,angle3Arc2,Angle4Arc2
    Call ArcBothEndpointChecking(SXArc1, SYArc1, EXArc1, EYArc1, SXArc2, SYArc2,
EXArc2, EYArc2, AllpointSame, OnepointCheck)
    If AllpointSame = True And OnepointCheck = True Then 'no need to divide the Arc
    ElseIf AllpointSame = False And OnepointCheck = True Then 'divide the Arc into two
        LineInter.List(ArcArcCounter + 6) = int_X1
        LineInter.List(ArcArcCounter + 7) = int_Y1
        Call GeneralARC(CXArc2, CYArc2, RArc2, int_X1, int_Y1, EXArc2, EYArc2)
        ExitArc2 = True
    Else
        LineInter.List(ArcArcCounter + 6) = int_X1
        LineInter.List(ArcArcCounter + 7) = int_Y1
        Call GeneralARC(CXArc2, CYArc2, RArc2, int_X1, int_Y1, int_x2, int_Y2)
        Call GeneralARC(CXArc2, CYArc2, RArc2, int_x2, int_Y2, EXArc2, EYArc2)
        ExitArc2 = True
    End If
Else ' Not Normal division for Arc2 like startArc2,angle3Arc2,Angle2Arc2,Angle4Arc2
    Call ArcBothEndpointChecking(SXArc1, SYArc1, EXArc1, EYArc1, SXArc2, SYArc2,
EXArc2, EYArc2, AllpointSame, OnepointCheck)
    If AllpointSame = True And OnepointCheck = True Then 'no need to divide the Arc
    ElseIf AllpointSame = False And OnepointCheck = True Then 'divide the Arc into two
        LineInter.List(ArcArcCounter + 6) = int_x2
        LineInter.List(ArcArcCounter + 7) = int_Y2
        Call GeneralARC(CXArc2, CYArc2, RArc2, int_x2, int_Y2, EXArc2, EYArc2)
        ExitArc2 = True
    Else
        LineInter.List(ArcArcCounter + 6) = int_x2
        LineInter.List(ArcArcCounter + 7) = int_Y2
        Call GeneralARC(CXArc2, CYArc2, RArc2, int_x2, int_Y2, int_X1, int_Y1)
        Call GeneralARC(CXArc2, CYArc2, RArc2, int_X1, int_Y1, EXArc2, EYArc2)
        ExitArc2 = True
    End If
End If
'Exit Do
'One intersec for both ARC
Elseif CheckAngle2Arc1 = True Or CheckAngle3Arc1 = True Or CheckAngle2Arc2 = True
Or CheckAngle3Arc2 = True Then
    'For Arc1 Single intersection
    If CheckAngle2Arc1 = True Then

```

```

'Call findXYfromAngle(Angle2, CXArc1, CYArc1, RArc1, XfromAngle, YfromAngle)
ForcompareangleX = int_X1 - CXArc2
ForcompareangleY = int_Y1 - CYArc2
CompareAngle = Val(FormatNumber(atan2(ForcompareangleX, ForcompareangleY), 3))

Satisfied)

Call FindIntersectionSatisfiedByAnotherArc(Angle1 Arc2, CompareAngle, Angle4 Arc2,

'Satisfied=true then divide the Arc1 by angle2 into two Arc
If Satisfied = True Then 'Divide the Arc
    LineInter.List(Arccounter + 6) = int_X1
    LineInter.List(Arccounter + 7) = int_Y1
    Call GeneralARC(CXArc1, CYArc1, RArc1, int_X1, int_Y1, EXArc1, EYArc1)
    Satisfied = False
    Satisfied1 = True
End If
End If
If CheckAngle3 Arc1 = True Then
'Call findXYfromAngle(Angle3, CXArc1, CYArc1, RArc1, XfromAngle, YfromAngle)
ForcompareangleX = int_x2 - CXArc2
ForcompareangleY = int_Y2 - CYArc2
CompareAngle = Val(FormatNumber(atan2(ForcompareangleX, ForcompareangleY), 3))
Call FindIntersectionSatisfiedByAnotherArc(Angle1 Arc2, CompareAngle, Angle4 Arc2,

Satisfied)

'Satisfied=true then divide the Arc1 by angle2 into two Arc
If Satisfied = True Then 'Divide the Arc
    LineInter.List(Arccounter + 6) = int_x2
    LineInter.List(Arccounter + 7) = int_Y2
    Call GeneralARC(CXArc1, CYArc1, RArc1, int_x2, int_Y2, EXArc1, EYArc1)
    Satisfied = False
    Satisfied2 = True
End If
End If
'For Arc2 Single intersection
If CheckAngle2 Arc2 = True Then
'Call findXYfromAngle(Angle2, CXArc2, CYArc2, RArc2, XfromAngle, YfromAngle)
ForcompareangleX = int_X1 - CXArc1
ForcompareangleY = int_Y1 - CYArc1
CompareAngle = Val(FormatNumber(atan2(ForcompareangleX, ForcompareangleY), 3))
Call FindIntersectionSatisfiedByAnotherArc(Angle1, CompareAngle, Angle4, Satisfied)
'Satisfied=true then divide the Arc2 by angle2 into two Arc
If Satisfied = True Then 'Divide the Arc
    LineInter.List(ArcArcCounter + 6) = int_X1
    LineInter.List(ArcArcCounter + 7) = int_Y1
    Call GeneralARC(CXArc2, CYArc2, RArc2, int_X1, int_Y1, EXArc2, EYArc2)
    Satisfied = False
    Satisfied3 = True
End If
End If
If CheckAngle3 Arc2 = True Then
'Call findXYfromAngle(Angle3, CXArc2, CYArc2, RArc2, XfromAngle, YfromAngle)
ForcompareangleX = int_x2 - CXArc1
ForcompareangleY = int_Y2 - CYArc1
CompareAngle = Val(FormatNumber(atan2(ForcompareangleX, ForcompareangleY), 3))
Call FindIntersectionSatisfiedByAnotherArc(Angle1, CompareAngle, Angle4, Satisfied)
'Satisfied=true then divide the Arc2 by angle2 into two Arc
If Satisfied = True Then 'Divide the Arc
    LineInter.List(ArcArcCounter + 6) = int_x2
    LineInter.List(ArcArcCounter + 7) = int_Y2
    Call GeneralARC(CXArc2, CYArc2, RArc2, int_x2, int_Y2, EXArc2, EYArc2)
    Satisfied = False
    Satisfied4 = True
End If
End If

End If
'***For Arc1 End
If Satisfied1 = True Or Satisfied2 = True Or Satisfied3 = True Or Satisfied4 = True Or

ExitArc1 = True _

    Or ExitArc2 = True Then
    Satisfied1 = False
    Satisfied2 = False
    Satisfied3 = False
    Satisfied4 = False

```

```
ExitArc1 = False
ExitArc2 = False
Arccounter = -1
Exit Do
End If
```

```
End If
End If
'ArcArcCounter = ArcArcCounter + 1
Loop
```

```
End If
Arccounter = Arccounter + 1
Loop
```

'Count each object like all object of line,circle,arc.... that is call Object_Counter. Create a selection set and give a name of each selection set like

```
'SELECTIONSET_text
```

```
'End If
```

```
Call Delete_Selectionset
Call ReloadLine_Array
Call ReloadCircle_Array
Call ReloadArc_Array
'Call Draw_Line
If Total_Line = 0 And TEST_SELECTIONSET_Line_Access = True Then
    acadDoc.SelectionSets.Item("TEST_SELECTIONSET_line").Delete
    TEST_SELECTIONSET_Line_Access = False
Else
    Call Draw_Line_selectionset
End If
'Call Draw_Circle
If Total_Circle = 0 And TEST_SELECTIONSET_circle_Access = True Then
    acadDoc.SelectionSets.Item("TEST_SELECTIONSET_circle").Delete
    TEST_SELECTIONSET_circle_Access = False
Else
    Call Draw_Circle_selectionset
End If
'Call Draw_Arc
If Total_ARC = 0 And TEST_SELECTIONSET_circle_Access = True Then
    acadDoc.SelectionSets.Item("TEST_SELECTIONSET_arc").Delete
    TEST_SELECTIONSET_Arc_Access = False
Else
    Call Draw_Arc_selectionset
End If
Call Draw_ObjectNumber
```

```
acadDoc.sendcommand " zoom a "
Call Objectlist
End Function
```

```
Private Sub Command7_Click()
Set acadApp = GetObject( "AutoCAD.Application")
If Err Then
    Err.Clear
    ' Start AutoCAD if it is not running.
    Set acadApp = CreateObject("AutoCAD.Application")
    acadApp.Visible = True
    If Err Then
        MsgBox Err.Description
    End If
End If
```

```

Exit Sub
End If
End If
Set acadDoc = acadApp.ActiveDocument
Set mspace = acadDoc.ModelSpace

' This example creates a new layer named "ABC" (colored red.)
' It then creates a circle and assigns it to layer "ABC"

' Create new layer
Dim layerObj As Object
Set layerObj = acadDoc.Layers.Add("ABC")
layerObj.Color = acRed

' Create Circle
Dim Circleobj As Object
Dim center(0 To 2) As Double
Dim Radius As Double
center(0) = 100: center(1) = 100: center(2) = 0
Radius = 40
Set Circleobj = mspace.AddCircle(center, Radius)
ZoomAll
MsgBox "The circle has been created on layer " & Circleobj.Layer, , "Layer Example"

' Set the layer of new circle to "ABC"
Circleobj.Layer = "ABC"
' Refresh view
acadDoc.Regen (True)
MsgBox "The circle is now on layer " & Circleobj.Layer, , "Layer Example"

acadDoc.sendcommand " _zoom a "
End Sub

Private Sub Command8_Click()
Call Toggol
End Sub
Function Toggol()
If toggle_on = False Then
Call Delete_Selectionset
Call ReloadLine_Array
Call ReloadCircle_Array
Call ReloadArc_Array
'Call Draw_Line
If Total_Line = 0 And TEST_SELECTIONSET_Line_Access = True Then
acadDoc.SelectionSets.Item("TEST_SELECTIONSET_line").Delete
TEST_SELECTIONSET_Line_Access = False
Else
Call Draw_Line_selectionset
End If
'Call Draw_Circle
If Total_Circle = 0 And TEST_SELECTIONSET_circle_Access = True Then
acadDoc.SelectionSets.Item("TEST_SELECTIONSET_circle").Delete
TEST_SELECTIONSET_circle_Access = False
Else
Call Draw_Circle_selectionset
End If

'Call Draw_Arc
If Total_ARC = 0 And TEST_SELECTIONSET_circle_Access = True Then
acadDoc.SelectionSets.Item("TEST_SELECTIONSET_arc").Delete
TEST_SELECTIONSET_Arc_Access = False
Else
Call Draw_Arc_selectionset
End If
'Call Draw_ObjectNumber
acadDoc.sendcommand " _zoom a "
'Me.MousePointer = 0
Form1.Show
toggle_on = True
Else
Call Draw_ObjectNumber
toggle_on = False
End If

```

```
acadDoc.sendcommand " _zoom a "
```

```
End Function
```

```
Private Sub Command9_Click()
```

```
On Error Resume Next
```

```
Set acadApp = GetObject(, "AutoCAD.Application")
```

```
If Err Then
```

```
Err.Clear
```

```
' Start AutoCAD if it is not running.
```

```
Set acadApp = CreateObject("AutoCAD.Application")
```

```
acadApp.Visible = True
```

```
If Err Then
```

```
MsgBox Err.Description
```

```
Exit Sub
```

```
End If
```

```
End If
```

```
Set acadDoc = acadApp.ActiveDocument
```

```
Set mspace = acadDoc.ModelSpace
```

```
ThisDrawing.Application.WindowTop = 0
```

```
acadApp.WindowTop = 0
```

```
acadApp.WindowLeft = 0
```

```
acadApp.Width = 8000
```

```
acadApp.Height = 8000
```

```
acadDoc.sendcommand " _zoom a "
```

```
End Sub
```

```
Private Sub Form_Initialize()
```

```
scaleFactor = 1
```

```
End Sub
```

```
Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
```

```
Dim storenumber As Integer, GiveObjectnumberString As String, giveObjectnumber As Integer, k As Integer, Line1full As Boolean, Line2full As Boolean
```

```
'for line line intersection array specify
```

```
Dim inter_x As Single, inter_y As Single, t1 As Double, t2 As Double, InterSec As Boolean
```

```
Dim Circle1full As Boolean, Circle2full As Boolean, Arc1full As Boolean, Arc2full As Boolean
```

```
Dim Angle1 As Single, Angle2 As Single, StartArcangle As Single, EndArcangle As Single, interFirstpoint As Boolean,
```

```
InterSecondpoint As Boolean
```

```
Dim FirstPoint As Boolean, SecondPoint As Boolean
```

```
Dim FirstPointLine As Boolean, SecondPointLine As Boolean
```

```
Dim Scaletype As Integer
```

```
FirstPointLine = True
```

```
SecondPointLine = False
```

```
radiouTrueorFalse = True
```

```
RadiouFillet = 0
```

```
Erase Line1()
```

```
Line1full = False
```

```
Line2full = False
```

```
Circle1full = False
```

```
Circle2full = False
```

```
Arc1full = False
```

```
Arc2full = False
```

```
storenumber = 0
```

```
Position(0) = ""
```

```
Position(1) = ""
```

```
'For fillet
```

```
If KeyCode = 70 Then
```

```
If Shift = 2 Then
```

```
If Total_Object = 0 Then
```

```
MsgBox "There is no object to Edit"
```

```
Exit Sub
```

```
End If
```

```
Call LookforObjectNumber
```

```
For k = 1 To 2
```

```
'Call LookforObjectNumber
```

```
GiveObjectnumberString = InputBox(("Enter the") & k & (" number of the object for fillet"), "Fillet", , 0, 0)
```

```
giveObjectnumber = Val(GiveObjectnumberString)
```

```
If giveObjectnumber > (Total_Object - 1) Or giveObjectnumber < 0 Then
```

```
m = MsgBox("This is not a valid Object Number", vbOKOnly, " ")
```

```
Exit Sub
```

```

Else
    If AlloBjectName(giveObjectnumber) Like "line*" Then
        storenumber = Val(Mid$(AlloBjectName(giveObjectnumber), 5))
        'Search the 4 value of the line like Start X,Start Y, End X, End Y from the list box
        Call LineSearch(storenumber, Line1full, Line2full)
        MsgBox "wait1"
        If Position(0) = "arc" Then
            Position(1) = "line"
        ElseIf Position(0) = "" Then
            Position(0) = "line1"
        ElseIf Position(0) Like "line*" Then
            Position(1) = "line2"
        End If

        ElseIf AlloBjectName(giveObjectnumber) Like "circle*" Then
            storenumber = Val(Mid$(AlloBjectName(giveObjectnumber), 7))
        ElseIf AlloBjectName(giveObjectnumber) Like "arc*" Then
            storenumber = Val(Mid$(AlloBjectName(giveObjectnumber), 4))
            Call Arcsearch(storenumber, Arc1full, Arc2full)
            If Position(0) Like "line*" Then
                Position(1) = "arc"
            Else
                Position(0) = "arc"
            End If
        End If
    End If
Next k
If Line1full = True And Line2full = True Then
    'check for intersection call
    Call FindLineIntersection(Line1(1), Line1(2), Line1(3), Line1(4), Line2(1), Line2(2), Line2(3), Line2(4), _
        inter_x, inter_y, t1, t2, InterSec)
    If InterSec = True Then
        MsgBox "do fillet"
        Call Fillet(inter_x, inter_y)
    Else
        MsgBox "you can't fillet"
        Exit Sub
    End If
ElseIf Line1full = True And Arc1full = True Then
    'Call Linecircleintersec(Line1(1), Line1(2), Line1(3), Line1(4), Line2(1), Line2(2), Line2(3), Line2(4), _
        inter_x, inter_y, t1, t2, InterSec)
    Call CircleInter(Line1(1), Line1(2), Line1(3), Line1(4), Arc1(1), Arc1(2), Arc1(3), int_X1, int_Y1, int_x2, int_Y2,
Comp, Compare, oneintersec, twointersec)
'Validation for oneintersection and twointersection
    firstinterX = Arc1(4) - Arc1(1)
    FirstinterY = Arc1(5) - Arc1(2)
    StartArcangle = FormatNumber(atan2(firstinterX, FirstinterY), 3)

    firstinterX = Arc1(6) - Arc1(1)
    FirstinterY = Arc1(7) - Arc1(2)
    EndArcangle = FormatNumber(atan2(firstinterX, FirstinterY), 3)

    firstinterX = int_X1 - Arc1(1)
    FirstinterY = int_Y1 - Arc1(2)
    Angle1 = FormatNumber(atan2(firstinterX, FirstinterY), 3)
    firstinterX = int_x2 - Arc1(1)
    FirstinterY = int_Y2 - Arc1(2)
    Angle2 = FormatNumber(atan2(firstinterX, FirstinterY), 3)

    'Call checkAngleForfillet(Angle1, Angle2, StartArcangle, EndArcangle, interFirstpoint, InterSecondpoint)

    Call FindSameArcLineForFillet(Arc1(4), Arc1(5), Arc1(6), Arc1(7), Line1(1), Line1(2), Line1(3), Line1(4),
FirstPoint, SecondPoint)

    Call IntersectionpointbetweenStartorEndAngleArc(StartArcangle, Angle1, EndArcangle, interFirstpoint)
    Call IntersectionpointbetweenStartorEndAngleArc(StartArcangle, Angle2, EndArcangle, InterSecondpoint)

    Call FindBoundary(Line1(1), Line1(2), Line1(3), Line1(4), int_X1, int_Y1, _
        int_x2, int_Y2, FirstPointLine, SecondPointLine)
    ' CheckForIntersectionMatchingwithEndorStartpointLINE(x111, y111, x122, y122, int_X1, int_Y1)
    If interFirstpoint = True And FirstPointLine = True Then
        If interFirstpoint = True And FirstPointLine = True Then
            Call ArcLineFillet(int_X1, int_Y1)

```



```

        ElseIf InterSecondpoint = True And SecondPointLine = True Then
            Call ArcLineFillet(int_x2, int_Y2)
        End If
    ElseIf InterSecondpoint = True And SecondPointLine = True Then
        If interFirstpoint = True And FirstPointLine = True Then
            Call ArcLineFillet(int_X1, int_Y1)
        ElseIf InterSecondpoint = True And SecondPointLine = True Then
            Call ArcLineFillet(int_x2, int_Y2)
        End If
    End If

End If
End If
'For chamfer
ElseIf KeyCode = 67 Then
    If Shift = 2 Then

        If Total_Object = 0 Then
            MsgBox "There is no object to Edit"
            Exit Sub
        End If
        Call LookforObjectNumber
        For k = 1 To 2
            GiveObjectnumberString = InputBox(("Enter the") & k & (" number of the object for chamfer"))
            giveObjectnumber = Val(GiveObjectnumberString)
            If giveObjectnumber > (Total_Object - 1) Or giveObjectnumber < 0 Then
                m = MsgBox("This is not a valid Object Number", vbOKOnly, " ")
                Exit Sub
            Else
                If AllobjectName(giveObjectnumber) Like "line*" Then
                    storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 5))
                    'Search the 4 value of the line like Start X,Start Y, End X, End Y form the list box
                    Call LineSearch(storenumber, Line1full, Line2full)
                ElseIf AllobjectName(giveObjectnumber) Like "circle*" Then
                    storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 7))
                ElseIf AllobjectName(giveObjectnumber) Like "arc*" Then
                    storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 4))
                    'call ArcSearch
                End If
            End If
        Next k
        If Line1full = True And Line2full = True Then
            'check for intersection call
            Call FindLineIntersection(Line1(1), Line1(2), Line1(3), Line1(4), Line2(1), Line2(2), Line2(3), Line2(4), _
                inter_x, inter_y, t1, t2, InterSec)
            If InterSec = True Then
                MsgBox " do Chamfer"
                Call Chamfer(inter_x, inter_y)
            Else
                m = MsgBox(" you can't Chamfer", vbOKOnly, "Chamfer")
                Exit Sub
            End If
        End If
    End If
    'Trim
ElseIf KeyCode = 84 Then
    If Shift = 2 Then
        'For k = 1 To 2
        If Total_Object = 0 Then
            MsgBox "There is no object to Trim"
            Exit Sub
        End If
        Call LookforObjectNumber
        giveObjectnumber = InputBox(("Enter the") & 1 & (" number of the object for Trim"))
        If AllobjectName(giveObjectnumber) Like "line*" Then
            storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 5))
            'Search the 4 value of the line like Start X,Start Y, End X, End Y form the list box
            Call LineSearch(storenumber, Line1full, Line2full)
        ElseIf AllobjectName(giveObjectnumber) Like "circle*" Then
            storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 7))
            Call Circlesearch(storenumber, Circle1full, Circle2full)
        ElseIf AllobjectName(giveObjectnumber) Like "arc*" Then
            storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 4))
            Call Arcsearch(storenumber, Arc1full, Arc2full)
        End If
    End If

```

```

Next k
If Line1full = True Then
    'check for intersection call

    LineInter.List(Line1(0)) = "Line1"
    Call DeleteSeleReloadArrayDrawobject
End If
If Circle1full = True Then
    'check for intersection call

    LineInter.List(Circle1(0)) = "circle1"
    Call DeleteSeleReloadArrayDrawobject
End If
If Arc1full = True Then
    'check for intersection call

    LineInter.List(Arc1(0)) = "arc1"
    Call DeleteSeleReloadArrayDrawobject
End If
End If
'Tangent Alt^g
ElseIf KeyCode = 71 Then
    If Shift = 2 Then
        If Total_Object = 0 Then
            MsgBox "There is no object to Edit"
            Exit Sub
        End If
        Call LookforObjectNumber
        For k = 1 To 2
            GiveObjectnumberString = InputBox(("Enter the") & k & (" number of the Circle or Arc"), "For Tangent")
            giveObjectnumber = Val(GiveObjectnumberString)
            If giveObjectnumber > (Total_Object - 1) Or giveObjectnumber < 0 Then
                m = MsgBox("This is not a valid Object Number", vbOKOnly, " ")
                Exit Sub
            Else
                If AllobjectName(giveObjectnumber) Like "circle*" Then
                    storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 7))
                    'Search the 3 value of the circle like center X,center Y and radius form the list box
                    Call Circlesearch(storenumber, Circle1full, Circle2full)
                    If Position(0) = "arc" Then
                        Position(1) = "circle"
                    ElseIf Position(0) = "" Then
                        Position(0) = "circle1"
                    ElseIf Position(0) Like "line*" Then
                        Position(1) = "circle2"
                    End If
                ElseIf AllobjectName(giveObjectnumber) Like "arc*" Then
                    storenumber = Val(Mid$(AllobjectName(giveObjectnumber), 4))
                    Call Arcsearch(storenumber, Arc1full, Arc2full)
                    If Position(0) Like "circle*" Then
                        Position(1) = "arc"
                    Else
                        Position(0) = "arc"
                    End If
                End If
            End If
        Next k
        If Circle1full = True And Circle2full = True Then
            Call TangentForCircleCircle
        ElseIf Circle1full = True And Arc1full = True Then
            Call TangentForCircleArc
        ElseIf Arc1full = True And Arc2full = True Then
            Call TangentForArcArc
        End If
    End If
    'For ^Joint
    ElseIf KeyCode = vbKeyJ Then
        If Shift = 2 Then
            If Total_Object = 0 Then
                MsgBox "There is no object to Edit"
                Exit Sub
            End If
            Call LookforObjectNumber
            For k = 1 To 2
                GiveObjectnumberString = InputBox(("Enter the") & k & (" number of the Circle or Arc"))

```

```

giveObjectnumber = Val(GiveObjectnumberString)
If giveObjectnumber > (Total_Object - 1) Or giveObjectnumber < 0 Then
    m = MsgBox("This is not a valid Object Number", vbOKOnly, " ")
    Exit Sub
Else
    If AlloBjectName(giveObjectnumber) Like "arc*" Then
        storenumber = Val(Mid$(AlloBjectName(giveObjectnumber), 4))
        'Search the 3 value of the circle like center X,center Y and radius form the list box
        Call Arcsearch(storenumber, Arc1full, Arc2full)
        If Position(0) = "line" Then
            Position(1) = "arc"
        ElseIf Position(0) = "" Then
            Position(0) = "arc1"
        ElseIf Position(0) Like "line*" Then
            Position(1) = "arc2"
        ElseIf Position(0) Like "arc*" Then
            Position(1) = "arc2"
        End If
    ElseIf AlloBjectName(giveObjectnumber) Like "line*" Then
        storenumber = Val(Mid$(AlloBjectName(giveObjectnumber), 5))
        Call LineSearch(storenumber, Arc1full, Arc2full)
        If Position(0) Like "arc*" Then
            Position(1) = "line"
        Else
            Position(0) = "line"
        End If
    End If
End If
Next k

If Arc1full = True And Arc2full = True Then
    Call JointArcArc
ElseIf Arc1full = True And Line1full = True Then
    Call JointArcLine
End If
End If
'Zoom In
ElseIf KeyCode = vbKeyUp Then

    If Shift = 2 Then

        Scaletype = acZoomScaledRelative
        acadDoc.sendcommand " _zoom " & scaleFactor & vbCr
        scaleFactor = scaleFactor + 1
    End If
'Zoom Out
ElseIf KeyCode = vbKeyDown Then
    If Shift = 2 Then
        If scaleFactor > 1 Then
            scaleFactor = scaleFactor - 1
        End If
        Scaletype = acZoomScaledRelative
        acadDoc.sendcommand " _zoom " & scaleFactor & vbCr
    End If
End If
End Sub

Function JointArcArc()
Dim AskforStartorEndpointFirstArc As String, AskforStartorEndpointSecondArc As String
AskforStartorEndpointFirstArc = InputBox("Input S as startpoint and E as Endpoint for the First Arc", "Startpoint or Endpoint", , 0, 0)

AskforStartorEndpointSecondArc = InputBox("Input S as startpoint and E as Endpoint for the Second Arc", "Startpoint or Endpoint", , 0, 0)

If (AskforStartorEndpointFirstArc = "S" Or AskforStartorEndpointFirstArc = "s") And (AskforStartorEndpointSecondArc = "S" Or AskforStartorEndpointSecondArc = "s") Then
'Connect Frist Arc Start point with Second Arc's Start point
LineInter.AddItem "line"
LineInter.AddItem (Arc1(4))
LineInter.AddItem (Arc1(5))
LineInter.AddItem (Arc2(4))
LineInter.AddItem (Arc2(5))
ElseIf (AskforStartorEndpointFirstArc = "S" Or AskforStartorEndpointFirstArc = "s") And (AskforStartorEndpointSecondArc = "E" Or AskforStartorEndpointSecondArc = "e") Then

```

```

'Connect Fritst Arc Start point with Second Arc's Endpoint point
    LineInter.AddItem "line"
    LineInter.AddItem (Arc1(4))
    LineInter.AddItem (Arc1(5))
    LineInter.AddItem (Arc2(6))
    LineInter.AddItem (Arc2(7))
ElseIf (AskforStartorEndpointFirstArc = "E" Or AskforStartorEndpointFirstArc = "e") And (AskforStartorEndpointSecondArc =
"S" Or AskforStartorEndpointSecondArc = "s") Then
'Connect Fritst Arc Endpoint point with Second Arc's Start point
    LineInter.AddItem "line"
    LineInter.AddItem (Arc1(6))
    LineInter.AddItem (Arc1(7))
    LineInter.AddItem (Arc2(4))
    LineInter.AddItem (Arc2(5))
ElseIf (AskforStartorEndpointFirstArc = "E" Or AskforStartorEndpointFirstArc = "e") And (AskforStartorEndpointSecondArc =
"E" Or AskforStartorEndpointSecondArc = "e") Then
'Connect Fritst Arc End point with Second Arc's End point
    LineInter.AddItem "line"
    LineInter.AddItem (Arc1(6))
    LineInter.AddItem (Arc1(7))
    LineInter.AddItem (Arc2(6))
    LineInter.AddItem (Arc2(7))
End If
Call DeleteScleReloadArrayDrawobject
End Function
Function JointArcLine()

End Function
Private Function checkAngleForfillet(Angle1 As Single, Angle2 As Single, StartArcangle As Single, EndArcangle As Single,
interFirstpoint As Boolean, InterSecondpoint As Boolean)
'Dim Angle1 As Single, Angle2 As Single, StartArcangle As Single, EndArcangle As Single
Dim DifferentAngle1 As Single
DifferentAngle1 = Abs(Angle1 - StartArcangle)
If DifferentAngle1 < 0.5 Then
    StartArcangle = Angle1
End If
DifferentAngle1 = Abs(Angle1 - EndArcangle)
If DifferentAngle1 < 0.5 Then
    EndArcangle = Angle1
End If
'considering Angle2
DifferentAngle1 = Abs(Angle2 - StartArcangle)
If DifferentAngle1 < 0.5 Then
    StartArcangle = Angle2
End If
DifferentAngle1 = Abs(Angle2 - EndArcangle)
If DifferentAngle1 < 0.5 Then
    EndArcangle = Angle2
End If

If (StartArcangle < EndArcangle) Then
    If (Angle1 >= StartArcangle And Angle1 <= EndArcangle) Then
        interFirstpoint = True
    Else
        interFirstpoint = False
    End If
    If Angle2 >= StartArcangle And Angle2 <= EndArcangle Then
        InterSecondpoint = True
    Else
        InterSecondpoint = False
    End If
Else
    If Angle1 >= StartArcangle And Angle1 <= 360 Or Angle1 >= 0 And Angle1 <= EndArcangle Then
        interFirstpoint = True
    Else
        interFirstpoint = False
    End If
    If Angle2 >= StartArcangle And Angle2 <= 360 Or Angle2 >= 0 And Angle2 <= EndArcangle Then
        InterSecondpoint = True
    Else
        InterSecondpoint = False
    End If
End If
Call Objectlist
End Function

```

```

Private Sub Form_Load()
copy_C
TEST_SELECTIONSET_circle_Access = False
TEST_SELECTIONSET_Arc_Access = False
TEST_SELECTIONSET_Line_Access = False
LineInter.Visible = False
Command1.Visible = False
Command2.Visible = False
Command3.Visible = False
Command4.Visible = False
Command5.Visible = False
Command6.Visible = False
Command8.Visible = False
Command11.Visible = False
End Sub

Public Function FindSame(ByVal x11 As Single, ByVal y11 As Single, _
    ByVal x12 As Single, ByVal y12 As Single, _
    ByVal x21 As Single, ByVal y21 As Single, _
    ByVal x22 As Single, ByVal y22 As Single, _
    ByRef inter_x As Single, ByRef inter_y As Single) As Boolean

    If (x11 = x21 And y11 = y21) Then
        inter_x = x11
        inter_y = y11
        FindSame = True
    ElseIf (x11 = x22 And y11 = y22) Then
        inter_x = x11
        inter_y = y11
        FindSame = True
    ElseIf (x12 = x21 And y12 = y21) Then
        inter_x = x12
        inter_y = y12
        FindSame = True
    ElseIf (x12 = x22 And y12 = y22) Then
        inter_x = x12
        inter_y = y12
        FindSame = True
    End If

End Function

Public Function FindSameArcLine(ByVal ArcX1 As Single, ByVal ArcY1 As Single, _
    ByVal ArcX2 As Single, ByVal ArcY2 As Single, _
    ByRef LineX1 As Single, ByRef LineY1 As Single, _
    ByRef LineX2 As Single, ByRef LineY2 As Single) As Boolean
Dim FirstPoint As Boolean
Dim SecondPoint As Boolean
FindSameArcLine = False
If ((LineX1 = ArcX1 And LineY1 = ArcY1) Or (LineX1 = ArcX2 And LineY1 = ArcY2)) Then
    FirstPoint = True
End If
If ((LineX2 = ArcX1 And LineY2 = ArcY1) Or (LineX2 = ArcX2 And LineY2 = ArcY2)) Then
    SecondPoint = True
End If
If (FirstPoint = True And SecondPoint = True) Then
    FindSameArcLine = True
End If
End Function

Private Function checkAngle(Angle1 As Single, Angle2 As Single, StartArcangle As Single, EndArcangle As Single,
    interFirstpoint As Boolean, InterSecondpoint As Boolean)
'Dim Angle1 As Single, Angle2 As Single, StartArcangle As Single, EndArcangle As Single
Dim DifferentAngle1 As Single
'Angle1 = FormatNumber(Angle1, 0)
'Angle2 = FormatNumber(Angle2, 0)
'StartArcangle = FormatNumber(StartArcangle, 0)
'EndArcangle = FormatNumber(EndArcangle, 0)
'considering Angle1
DifferentAngle1 = Abs(Angle1 - StartArcangle)
If DifferentAngle1 < 0.5 Then
    StartArcangle = Angle1
End If

```

```

DifferentAngle1 = Abs(Angle1 - EndArcangle)
If DifferentAngle1 < 0.5 Then
    EndArcangle = Angle1
End If
'considering Angle2
DifferentAngle1 = Abs(Angle2 - StartArcangle)
If DifferentAngle1 < 0.5 Then
    StartArcangle = Angle2
End If
DifferentAngle1 = Abs(Angle2 - EndArcangle)
If DifferentAngle1 < 0.5 Then
    EndArcangle = Angle2
End If

If (StartArcangle < EndArcangle) Then
    If (Angle1 > StartArcangle And Angle1 < EndArcangle) Then
        interFirstpoint = True
    Else
        interFirstpoint = False
    End If
    If Angle2 > StartArcangle And Angle2 < EndArcangle Then
        InterSecondpoint = True
    Else
        InterSecondpoint = False
    End If
Else
    If Angle1 > StartArcangle And Angle1 <= 360 Or Angle1 >= 0 And Angle1 < EndArcangle Then
        interFirstpoint = True
    Else
        interFirstpoint = False
    End If
    If Angle2 > StartArcangle And Angle2 <= 360 Or Angle2 >= 0 And Angle2 < EndArcangle Then
        InterSecondpoint = True
    Else
        InterSecondpoint = False
    End If
End If

End Function

'Public Function valid1(Angle1 As Single, Angle2 As Single, StartArcAngle As Single, EndArcAngle As Single, InterFirstPoint
As booean, InterSecondPoint As Boolean)

'End Function

Public Function ArcAndLineSame(ByVal ArcX1 As Single, ByVal ArcY1 As Single, _
    ByVal ArcX2 As Single, ByVal ArcY2 As Single, _
    ByRef LineX1 As Single, ByRef LineY1 As Single, _
    ByRef LineX2 As Single, ByRef LineY2 As Single) As Boolean
    ArcAndLineSame = False
    If (ArcX1 = ArcX2 And ArcY2 = ArcY2) Then
        If (ArcX1 = LineX1 And ArcY1 = LineY1) Or (ArcX1 = LineX2 And ArcY1 = LineY2) Then
            ArcAndLineSame = True
        End If
    End If
End Function

Public Function ArcOnTheLine(ByVal ArcX1 As Single, ByVal ArcY1 As Single, _
    ByVal ArcX2 As Single, ByVal ArcY2 As Single, _
    ByRef LineX1 As Single, ByRef LineY1 As Single, _
    ByRef LineX2 As Single, ByRef LineY2 As Single) As Boolean
    Dim firstVal As Single
    Dim secondVal As Single
    ArcOnTheLine = False
    firstVal = ((ArcX1 - LineX1) / (LineX1 - LineX2)) * (LineY1 - LineY2) + LineY1
    secondVal = ((ArcX2 - LineX1) / (LineX1 - LineX2)) * (LineY1 - LineY2) + LineY1
    If (firstVal = ArcY1 And secondVal = ArcY2) Then
        ArcOnTheLine = True
    Else
        ArcOnTheLine = False
    End If
End Function

```

```

End If
End Function
Public Function atan2(X, Y)
PI = 4 * Atn(1)
If X > 0 And Y > 0 Then
atan2 = Atn(Y / X)
ElseIf X > 0 And Y < 0 Then
atan2 = 2 * PI + Atn(Y / X)
ElseIf X < 0 And Y > 0 Then
atan2 = PI - Atn(Abs(Y / X))
ElseIf X < 0 And Y < 0 Then
atan2 = Atn(Abs(Y / X)) + PI
ElseIf X = 0 And Y > 0 Then
atan2 = PI / 2
ElseIf X = 0 And Y < 0 Then
atan2 = 3 / 2 * PI
ElseIf Y = 0 And X > 0 Then
atan2 = 0
ElseIf Y = 0 And X < 0 Then
atan2 = PI
ElseIf X = 0 And Y = 0 Then
atan2 = 0
End If
atan2 = 180 * atan2 / PI
End Function
Public Function Distance(X1 As Single, Y1 As Single, ix1 As Single, iy1 As Single, ix2 As Single, iy2 As Single, NearestPoint
As Boolean, _
, LongestPoint As Boolean, Angle1 As Single, Angle2 As Single, StartArcangle As Single)
Dim mainDistance As Single, Dist1 As Single, Dist2 As Single, Dist3 As Single
Dist1 = Sqr((ix1 - X1) ^ 2 + (iy1 - Y1) ^ 2)
Dist2 = Sqr((ix2 - X1) ^ 2 + (iy2 - Y1) ^ 2)
If Dist1 > Dist2 And ((Angle1 > StartArcangle) And (Angle1 < Angle2)) Then
LongestPoint = True
Else
NearestPoint = True
End If
End Function
' Return a value between PI and -PI.
Public Function GetAngle(ByVal Ax1 As Single, ByVal Ay1 As Single, ByVal Bx2 As Single, ByVal By2 As Single, ByVal
CX As Single, ByVal CY As Single, _
ByVal int_X1 As Single, ByVal int_Y1 As Single, ByVal int_x2 As Single, ByVal int_Y2 As Single, onepoint
As Boolean, _
otherpoint As Boolean) As Single
Dim PI As Single, GetangleLarge As Single, Angle1 As Single, Angle2 As Single
PI = 3.14159265
'Greater triangle
Dim side_a1 As Single, side_b1 As Single, side_c1 As Single

'one of two triangle inside of grater triangle
Dim side_a2 As Single, side_b2 As Single, side_c2 As Single

'Other triangle inside the grater triangle
Dim side_a3 As Single, side_b3 As Single, side_c3 As Single

Dim side_one As Single, side_other As Single, Full_distance As Single, Side_one2 As Single, Side_Other2 As Single

Dim side_a4 As Single, side_b4 As Single, side_c4 As Single

Dim side_a5 As Single, side_b5 As Single, side_c5 As Single, Angle3 As Single, Angle4 As Single

' Get the lengths of the triangle's sides for greater triangle
side_a1 = Sqr((CX - Bx2) ^ 2 + (CY - By2) ^ 2)
side_b1 = Sqr((CX - Ax1) ^ 2 + (CY - Ay1) ^ 2)
side_c1 = Sqr((Ax1 - Bx2) ^ 2 + (Ay1 - By2) ^ 2)

' Get the lengths of the triangle's sides for smaller triangle which is inside of the larger triangle
side_a2 = side_a1 'Sqr((Cx - Bx2) ^ 2 + (Cy - By2) ^ 2)
side_b2 = Sqr((CX - int_X1) ^ 2 + (CY - int_Y1) ^ 2)
side_c2 = Sqr((Bx2 - int_X1) ^ 2 + (By2 - int_Y1) ^ 2)

```

```

' Get the lengths of the triangle's sides for smaller triangle
side_a3 = Sqr((CX - int_X1) ^ 2 + (CY - int_Y1) ^ 2)
side_b3 = Sqr((CX - Ax1) ^ 2 + (CY - Ay1) ^ 2)
side_c3 = Sqr((Ax1 - int_X1) ^ 2 + (Ay1 - int_Y1) ^ 2)

' Get the lengths of the triangle's sides for smaller triangle which is inside of the larger triangle for int_x2 and int_y2
side_a4 = side_a1 * Sqr((Cx - Bx2) ^ 2 + (Cy - By2) ^ 2)
side_b4 = Sqr((CX - int_x2) ^ 2 + (CY - int_Y2) ^ 2)
side_c4 = Sqr((Bx2 - int_x2) ^ 2 + (By2 - int_Y2) ^ 2)

' Get the lengths of the triangle's sides for smaller triangle
side_a5 = Sqr((CX - int_x2) ^ 2 + (CY - int_Y2) ^ 2)
side_b5 = Sqr((CX - Ax1) ^ 2 + (CY - Ay1) ^ 2)
side_c5 = Sqr((Ax1 - int_x2) ^ 2 + (Ay1 - int_Y2) ^ 2)

If side_a1 = 0 Or side_b1 = 0 Or side_c1 = 0 Then
    side_one = Format$(Sqr((Ax1 - int_X1) ^ 2 + (Ay1 - int_Y1) ^ 2), "0.000")
    side_other = Format$(Sqr((int_X1 - Bx2) ^ 2 + (int_Y1 - By2) ^ 2), "0.000")
    Full_distance = Format$(Sqr((Ax1 - Bx2) ^ 2 + (Ay1 - By2) ^ 2), "0.000")
    Side_one2 = Format$(Sqr((Ax1 - int_x2) ^ 2 + (Ay1 - int_Y2) ^ 2), "0.000")
    Side_Other2 = Format$(Sqr((int_x2 - Bx2) ^ 2 + (int_Y2 - By2) ^ 2), "0.000")
    If side_one + side_other = Full_distance Then
        onepoint = True
    ElseIf Side_one2 + Side_Other2 = Full_distance Then
        otherpoint = True
    End If
Else
    ' Calculate angle ABint_x2.
    Angle4 = Format$(Acos((side_c5 ^ 2 - side_a5 ^ 2 - side_b5 ^ 2) / (-2 * side_a5 * side_b5)) / PI * 180, "0.000")
    ' Calculate angle B between sides abc.
    GetangleLarge = Format$(Acos((side_c1 ^ 2 - side_a1 ^ 2 - side_b1 ^ 2) / (-2 * side_a1 * side_b1)) / PI * 180, "0.000")

    ' Calculate angle int_x1BC.
    Angle1 = Format$(Acos((side_c2 ^ 2 - side_a2 ^ 2 - side_b2 ^ 2) / (-2 * side_a2 * side_b2)) / PI * 180, "0.000")
    ' Calculate angle ABint_x1.
    Angle2 = Format$(Acos((side_c3 ^ 2 - side_a3 ^ 2 - side_b3 ^ 2) / (-2 * side_a3 * side_b3)) / PI * 180, "0.000")

    ' Calculate angle int_x2BC.
    Angle3 = Format$(Acos((side_c4 ^ 2 - side_a4 ^ 2 - side_b4 ^ 2) / (-2 * side_a4 * side_b4)) / PI * 180, "0.000")

    If Angle3 + Angle4 = GetangleLarge Then
        otherpoint = True

    End If

    If Angle1 + Angle2 = GetangleLarge Then
        onepoint = True
    End If
End If
End Function

' Return the arccosine of X.
Function Acos(ByVal X As Single) As Single
    Acos = Atn(-X / Sqr(-X * X + 1)) + 2 * Atn(1)
End Function

Function DecideIntersection(ByVal Ax1 As Single, ByVal Ay1 As Single, ByVal Bx2 As Single, ByVal By2 As Single, ByVal
int_X1 As Single, ByVal int_Y1 As Single, ByVal int_x2 As Single, ByVal int_Y2 As Single, onepoint As Boolean, _
    otherpoint As Boolean)

Dim side_one As Single, side_other As Single, Full_distance As Single, Side_one2 As Single, Side_Other2 As Single
side_one = Format$(Sqr((Ax1 - int_X1) ^ 2 + (Ay1 - int_Y1) ^ 2), "0.000")
side_other = Format$(Sqr((int_X1 - Bx2) ^ 2 + (int_Y1 - By2) ^ 2), "0.000")
Full_distance = Format$(Sqr((Ax1 - Bx2) ^ 2 + (Ay1 - By2) ^ 2), "0.000")
Side_one2 = Format$(Sqr((Ax1 - int_x2) ^ 2 + (Ay1 - int_Y2) ^ 2), "0.000")
Side_Other2 = Format$(Sqr((int_x2 - Bx2) ^ 2 + (int_Y2 - By2) ^ 2), "0.000")
If side_one + side_other = Full_distance Then
    onepoint = True
ElseIf Side_one2 + Side_Other2 = Full_distance Then
    otherpoint = True
End If

End Function

```



```
Function ArcCircleInter(ByVal CXARC As Single, ByVal CYARC As Single, ByVal LRARC As Single, ByVal CxCircle_1 As
Single, ByVal CyCircle As Single, _
    ByVal Rcircle As Single, int_X1 As Single, int_Y1 As Single, int_x2 As Single, _
    int_Y2 As Single)
    'ByVal Comp As Single, ByVal oncentersec As Single, ByVal twointersec As Single)
```

```
Dim Distance As Single, Base_A As Single, Height As Single, X2 As Single, Y2 As Single
```

```
Distance = Abs(Sqr((CXARC - CxCircle_1) ^ 2 + (CYARC - CyCircle) ^ 2))
Base_A = (LRARC ^ 2 - Rcircle ^ 2 + Distance ^ 2) / (2 * Distance)
Height = Sqr(LRARC ^ 2 - Base_A ^ 2)
X2 = CXARC + Base_A * (CxCircle_1 - CXARC) / Distance
Y2 = CYARC + Base_A * (CyCircle - CYARC) / Distance
If Distance > (LRARC + Rcircle) Or Distance < Abs(LRARC - Rcircle) Or Distance = 0 Then
    Exit Function
Else
    int_X1 = FormatNumber(Val(X2 + Height * (CyCircle - CYARC) / Distance), 3)
    int_Y1 = FormatNumber(Val(Y2 - Height * (CxCircle_1 - CXARC) / Distance), 3)
    int_x2 = FormatNumber(Val(X2 - Height * (CyCircle - CYARC) / Distance), 3)
    int_Y2 = FormatNumber(Val(Y2 + Height * (CxCircle_1 - CXARC) / Distance), 3)
End If
End Function
```

```
Function FindSameArcCircle(ByVal SXarc As Single, ByVal SYarc As Single, ByVal EXarc As Single, ByVal EYarc As
Single, ByVal int_X1 As Single, _
    ByVal int_Y1 As Single, ByVal int_x2 As Single, ByVal int_Y2 As Single, Onetrue As Boolean, Othertrue As
Boolean)
```

```
'Dim Onetrue As Boolean, Othertrue As Boolean, SecondOnetrue As Boolean, SecondOthertrue As Boolean
If int_X1 = SXarc And int_Y1 = SYarc Then
    Onetrue = True
ElseIf int_X1 = EXarc And int_Y1 = EYarc Then
    Onetrue = True
End If

If int_x2 = SXarc And int_Y2 = SYarc Then
    Othertrue = True
ElseIf int_x2 = EXarc And int_Y2 = EYarc Then
    Othertrue = True
End If
```

```
End Function
Function LocalAxes(ByVal Angle1 As Single, ByVal Angle2 As Single, ByVal Angle3 As Single, ByVal Angle4 As Single, _
    FirstInterX1 As Boolean, FirstIntersecBetweenStartEndangle As Boolean, _
    SecondInterX1 As Boolean, SecondIntersecBetweenStartEndAngle As Boolean, CircleAngle1 As Single,
    CircleAngle2 As Single, _
    StartAngle As Single, EndAngle As Single, FirstIntersecAngle As Single, SecondIntersecAngle As Single)
```

```
'Angle2 mean Int_x1 and Angle3 mean int_x2
'StartAngle=startpoint-startangle
'endangle=360-startangle+endangle
If Angle1 >= Angle4 Then
    EndAngle = Angle4 - Angle1 + 360

    StartAngle = 0
'ElseIf Angle1 < Angle4 Then
    'endangle = Angle4 - Angle1

    'startangle = 0
End If
If Angle1 > Angle2 Then
    FirstIntersecAngle = Angle2 - Angle1 + 360
ElseIf Angle1 < Angle2 Then
    FirstIntersecAngle = Angle2 - Angle1
End If

If Angle1 > Angle3 Then
    SecondIntersecAngle = Angle3 - Angle1 + 360
ElseIf Angle1 < Angle3 Then
    SecondIntersecAngle = Angle3 - Angle1
End If
```

```

If FirstIntersecAngle < SecondIntersecAngle Then
    FirstInterX1 = True
Else
    FirstInterX1 = False
End If

If FirstIntersecAngle > StartAngle And FirstIntersecAngle < EndAngle Then
    FirstIntersecBetweenStartEndangle = True
End If
If SecondIntersecAngle > StartAngle And SecondIntersecAngle < EndAngle Then
    SecondIntersecBetweenStartEndAngle = True
End If
If SecondIntersecAngle < FirstIntersecAngle Then
    SecondInterX1 = True
End If

If StartAngle = FirstIntersecAngle And EndAngle = secondinteseccangle Then
    Normalintersec = True
ElseIf StartAngle = SecondIntersecAngle And EndAngle = FirstIntersecAngle Then
    Abnormalintersec = True
End If
End Function
Function BothArcSame(SXArc1 As Single, SYArc1 As Single, EXArc1 As Single, EYArc1 As Single, _
    SXArc2 As Single, SYArc2 As Single, EXArc2 As Single, EYArc2 As Single, OnepointSame As Boolean,
    OtherpointSame As Boolean)

If (SXArc1 = SXArc2 And SYArc1 = SYArc2) Or (SXArc1 = EXArc2 And SYArc1 = SYArc2) Then
    OnepointSame = True
End If
If (EXArc1 = SXArc2 And EYArc1 = EYArc2) Or (EXArc1 = EXArc2 And EYArc2 = EYArc2) Then
    OtherpointSame = True
End If

End Function

Public Sub Delete_Selectionset()
Dim Delete_line As Integer, Delete_circle As Integer, Delete_arc As Single
'Deleting Text object from Modulespace

    For delete_i = 0 To (Object_Counter - 1)
        SsetObj_text(delete_i).Erase
    Next delete_i
'Deleting line object from
    If Total_Line > 0 Then
        For Delete_line = 0 To (Total_Line - 1)
            ssetobj_Line(Delete_line).Erase
        Next Delete_line
    End If
'Deleting circle object from
    If Total_Circle > 0 Then
        For Delete_circle = 0 To (SellectionSet_TotalCircle - 1)
            ssetObj_Cir(Delete_circle).Erase
        Next Delete_circle
    End If
'Deleting Arc object from
    If Total_ARC > 0 Then
        For Delete_arc = 0 To (Total_ARC - 1)
            ssetObj_ARC(Delete_arc).Erase
        Next Delete_arc
    End If

    If Object_Counter > 0 Then
        acadDoc.SelectionSets.Item("SELECTIONSET_text").Delete
    End If

```

```
End Sub
'Refil all Dynamic Array like (Line_array,Circle_array,Arc_array...) by new Data from Lineinter List box
Public Sub ReloadLine_Array()
'Finding out line information from Lineinter list Box
Dim start As Integer
Linecounter = 0
start = 0
Total_Line = 0
Erase Line_Array

For Line_i = 0 To LineInter.ListCount - 1
    If LineInter.List(Line_i) = "line" Then
        Total_Line = Total_Line + 1
    End If
Next Line_i
If Total_Line > 0 Then
    ReDim Line_Array((Total_Line * 4) - 1)
    line_ii = 0

    Do While line_ii <= LineInter.ListCount
        If LineInter.List(line_ii) = "line" Then
            For i = start To LineInter.ListCount - 1 Step 4
                Line_Array(start) = LineInter.List(line_ii + 1)
                Line_Array(start + 1) = LineInter.List(line_ii + 2)
                Line_Array(start + 2) = LineInter.List(line_ii + 3)
                Line_Array(start + 3) = LineInter.List(line_ii + 4)
                start = start + 4
            Exit For
            Next i
        End If
        line_ii = line_ii + 1
    Loop
End If
'For i = 0 To LineInter.ListCount - 1
'Debug.Print Line_Array(i)
'Next i

End Sub

Public Sub ReloadCircle_Array()
'Finding out Circle information from Lineinter list Box
Dim start As Integer
Total_Circle = 0
Circlecounter = 0
start = 0
Erase Circle_Array
For Circle_i = 0 To LineInter.ListCount - 1
    If LineInter.List(Circle_i) = "circle" Then
        Total_Circle = Total_Circle + 1
    End If
Next Circle_i
If Total_Circle > 0 Then
    ReDim Circle_Array((Total_Circle * 3) - 1)
    circle_ii = 0

    Do While circle_ii <= LineInter.ListCount
        If LineInter.List(circle_ii) = "circle" Then
            For i = start To LineInter.ListCount - 1 Step 3
                Circle_Array(start) = LineInter.List(circle_ii + 1)
                Circle_Array(start + 1) = LineInter.List(circle_ii + 2)
                Circle_Array(start + 2) = LineInter.List(circle_ii + 3)
                start = start + 3
            Exit For
            Next i
        End If
        circle_ii = circle_ii + 1
    Loop
End If
End Sub

Public Sub ReloadArc_Array()
'Finding out Arc information from Lineinter list Box
```

```

Dim start As Integer
Total_ARC = 0
Arccounter = 0
start = 0
Erase Arc_Array
For arc_i = 0 To LineInter.ListCount - 1
    If LineInter.List(arc_i) = "arc" Then
        Total_ARC = Total_ARC + 1
    End If
Next arc_i
If Total_ARC > 0 Then
    ReDim Arc_Array((Total_ARC * 7) - 1)
    arc_ii = 0

Do While arc_ii <= LineInter.ListCount
    If LineInter.List(arc_ii) = "arc" Then
        For i = start To LineInter.ListCount - 1 Step 7
            Arc_Array(start) = LineInter.List(arc_ii + 1)
            Arc_Array(start + 1) = LineInter.List(arc_ii + 2)
            Arc_Array(start + 2) = LineInter.List(arc_ii + 3)
            Arc_Array(start + 3) = LineInter.List(arc_ii + 4)
            Arc_Array(start + 4) = LineInter.List(arc_ii + 5)
            Arc_Array(start + 5) = LineInter.List(arc_ii + 6)
            Arc_Array(start + 6) = LineInter.List(arc_ii + 7)
            start = start + 7
        Exit For
        Next i
    End If
    arc_ii = arc_ii + 1
Loop
End If
End Sub

Public Sub Draw_Line()
    If Total_Line > 0 Then
        ' This add a line in model space
        Line_count = Line_count + 1
        Total_Line = ((UBound(Line_Array()) + 1) / 4)
        ReDim Lineobj_1(Total_Line - 1) As Object
        i = 0
        Dim startPoint(0 To 2) As Double
        Dim endPoint(0 To 2) As Double
        Set ssetobj_Line = acadDoc.SelectionSets.Add("TEST_SELECTIONSET_line")
        ' Define the start and end points for the line
        For i = LBound(Line_Array()) To UBound(Line_Array()) Step 4
            startPoint(0) = Line_Array(i)
            startPoint(1) = Line_Array(i + 1)
            startPoint(2) = 0
            endPoint(0) = Line_Array(i + 2)
            endPoint(1) = Line_Array(i + 3)
            endPoint(2) = 0

            ' Create the line in model space
            If i = 0 Then
                Set Lineobj_1(i) = mspace.addline(startPoint, endPoint)
            Else
                Set Lineobj_1(i / 4) = mspace.addline(startPoint, endPoint)
            End If
        Next i
        ReDim ssobjs_line(Total_Line - 1) As Object
        Dim Line_i As Integer
        For Line_i = 0 To Total_Line - 1
            Set ssobjs_line(Line_i) = Lineobj_1(Line_i)
        Next Line_i
        ssetobj_Line.additems ssobjs_line
        TEST_SELECTIONSET_Line.Access = True
    End If
End Sub

Public Sub Draw_Circle()

    If Total_Circle > 0 Then

        Circle_count = Circle_count + 1
    
```

```

Total_Circle = ((UBound(Circle_Array) + 1) / 3)
ReDim Circleobj(Total_Circle - 1) As Object
'Define the circle
Dim C_centerPoint(0 To 2) As Double
Dim C_radius As Double
'fo selcet circle object as collection
'if edit_circle=
Set ssetObj_Cir = acadDoc.SelectionSets.Add("TEST_SELECTIONSET_circle")
i = 0
Dim count_D As Integer
count_D = 0

For i = LBound(Circle_Array()) To UBound(Circle_Array()) Step 3
    C_centerPoint(0) = Circle_Array(i)
    C_centerPoint(1) = Circle_Array(i + 1)
    C_centerPoint(2) = 0
    C_radius = Circle_Array(i + 2)

    ' Create the Circle object in model space
    If i = 0 Then
        Set Circleobj(i) = mspace.AddCircle(C_centerPoint, C_radius)
    Else
        Set Circleobj(i / 3) = mspace.AddCircle(C_centerPoint, C_radius)
    End If
Next i
ReDim ssobjs(Total_Circle - 1) As Object
Dim select_i As Integer
'PAY CONCENTRATION HERE
For select_i = 0 To Total_Circle - 1
    Set ssobjs(select_i) = Circleobj(select_i)
    If select_i = Total_Circle - 1 Then
        Exit For
    End If
Next select_i
SselectionSet_TotalCircle = Total_Circle
ssetObj_Cir.additems ssobjs
TEST_SELECTIONSET_circle_Access = True
End If
End Sub

Public Sub Draw_Arc()
    If Total_ARC > 0 Then
        'Define the ARC
        Total_ARC = ((UBound(Arc_Array) + 1) / 7)
        ReDim Arcobj(Total_ARC - 1) As Object
        'Define the circle
        Dim A_centerPoint(0 To 2) As Double
        Dim A_radius As Double, startAngleInDegree As Double
        Dim endAngleInDegree As Double
        Dim Angle1 As Double, Angle2 As Double, StartX As Single, StartY As Single, EndX As Single, EndY As Single
        'To selcet ARC object as collection

        Set ssetObj_ARC = acadDoc.SelectionSets.Add("TEST_SELECTIONSET_arc")
        i = 0
        'Dim count_D As Integer
        count_D = 0

        For i = LBound(Arc_Array()) To UBound(Arc_Array()) Step 7
            A_centerPoint(0) = Arc_Array(i)
            A_centerPoint(1) = Arc_Array(i + 1)
            A_centerPoint(2) = 0
            A_radius = Arc_Array(i + 2)
            StartX = Arc_Array(i + 3)
            StartY = Arc_Array(i + 4)
            EndX = Arc_Array(i + 5)
            EndY = Arc_Array(i + 6)
            'Find out start angle from the function name atn2()
            firstinterX = StartX - A_centerPoint(0)
            FirstinterY = StartY - A_centerPoint(1)
            Angle1 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

            firstinterX = EndX - A_centerPoint(0)
            FirstinterY = EndY - A_centerPoint(1)

```

```

Angle2 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))
If Angle2 = 0 Then
    Angle2 = 360
End If
Dim startAngleInRadian As Double
Dim endAngleInRadian As Double
startAngleInRadian = Angle1 * 3.141592 / 180
endAngleInRadian = Angle2 * 3.141592 / 180

' Create the Arc object in model space
If i = 0 Then
    Set Arcobj(i) = mspace.Addarc(A_centerPoint, A_radius, startAngleInRadian, endAngleInRadian)
Else
    Set Arcobj(i / 7) = mspace.Addarc(A_centerPoint, A_radius, startAngleInRadian, endAngleInRadian)
End If
Next i
ReDim ssObjs_ARC(Total_ARC - 1) As Object
Dim arc_i As Integer
'PAY CONCENTRATION HERE
For arc_i = 0 To Total_ARC - 1
    Set ssObjs_ARC(arc_i) = Arcobj(arc_i)
Next arc_i
ssetObj_ARC.additems ssObjs_ARC
TEST_SELECTIONSET_Arc_Access = True

End If
End Sub
'This module only for Draw object number on Moudel space or it will give the number like 0,1,2,3.... beside the objects.
Public Sub Draw_ObjectNumber()
Dim insertionPoint_2(0 To 2) As Double, height_1 As Integer, Height As Integer
Object_Counter = 0
Total_Object = Total_Circle + Total_Line + Total_ARC
If Total_Object = 0 Then
    Exit Sub
Else
    Set SsetObj_text = acadDoc.SelectionSets.Add("SELECTIONSET_text")
    Total_Object = Total_Circle + Total_Line + Total_ARC
    ReDim TextObj(Total_Object - 1) As Object
    If Total_Line > 0 Then

        'For ik = 0 To (LineInter.ListCount - 1) Step 5
        For ik = 0 To UBound(Line_Array()) Step 4
            'If LineInter.List(ik) = "line" Then
            textString = Object_Counter
            'insertionPoint_2(0) = (Val(LineInter.List(ik + 1)) + Val(LineInter.List(ik + 3))) / 2
            'insertionPoint_2(1) = (Val(LineInter.List(ik + 2)) + Val(LineInter.List(ik + 4))) / 2
            insertionPoint_2(0) = (Val(Line_Array(ik)) + Val(Line_Array(ik + 2))) / 2
            insertionPoint_2(1) = (Val(Line_Array(ik + 1)) + Val(Line_Array(ik + 3))) / 2
            insertionPoint_2(2) = 0
            height_1 = 5
            Set TextObj(Object_Counter) = mspace.
                AddText(textString, insertionPoint_2, height_1)

            Object_Counter = Object_Counter + 1

            'End If
        Next ik
    End If
    If Total_Circle > 0 Then
        For i = LBound(Circle_Array()) To UBound(Circle_Array()) Step 3
            textString = Object_Counter
            'CirDy(i) = Object_Counter
            'ObCount.AddItem (Object_Counter)
            'ObCount.AddItem ("circle")
            insertionPoint_2(0) = Circle_Array(i)
            insertionPoint_2(1) = Circle_Array(i + 1)
            insertionPoint_2(2) = 0
            Height = 5
            Set TextObj(Object_Counter) = mspace.
                AddText(textString, insertionPoint_2, Height)
            'TextObj.Update
            Object_Counter = Object_Counter + 1
            'CircleObj_counter = CircleObj_counter + 1
        Next i
    End If

```

```

Dim Center_ArcX As Single, Center_arcY As Single, Center_arcR As Single, PorAngle2 As Double, MidelAngle As Double
If Total_ARC > 0 Then
    For i = LBound(Arc_Array()) To UBound(Arc_Array()) Step 7

        Center_ArcX = Arc_Array(i)
        Center_arcY = Arc_Array(i + 1)
        Center_arcR = Arc_Array(i + 2)
        StartX = Arc_Array(i + 3)
        StartY = Arc_Array(i + 4)
        EndX = Arc_Array(i + 5)
        EndY = Arc_Array(i + 6)

        firstinterX = StartX - Arc_Array(i)
        FirstinterY = StartY - Arc_Array(i + 1)
        'Start Angle
        Angle1 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

        firstinterX = EndX - Arc_Array(i)
        FirstinterY = EndY - Arc_Array(i + 1)
        'End angle
        Angle2 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))
        If Angle1 > Angle2 Then
            Call StartAngleZiro(Angle1, Angle2, Center_ArcX, Center_arcY, Center_arcR, MidAngleX, MidAngleY)
        Else
            PorAngle2 = (Angle2 - Angle1)
            MidelAngle = (PorAngle2 / 2) + Angle1

            midangleinradian = MidelAngle * 3.141592 / 180
            MidAngleX = FormatNumber((Center_ArcX + (Center_arcR) * Cos(midangleinradian)), 3)
            MidAngleY = FormatNumber((Center_arcY + (Center_arcR) * Sin(midangleinradian)), 3)

        End If
        textString_1 = Object_Counter
        'ArcDy(i) = Object_Counter
        'ObCount.AddItem (Object_Counter)
        'ObCount.AddItem ("arc")
        insertionPoint(0) = MidAngleX
        insertionPoint(1) = MidAngleY
        insertionPoint(2) = 0
        Height = 5
        Set TextObj(Object_Counter) = mspace. _
            AddText(textString_1, insertionPoint, Height)
        'TextObj.Update
        Object_Counter = Object_Counter + 1
        'CircleObj_counter = CircleObj_counter + 1
    Next i
End If
ReDim Ssobjs_text((Object_Counter - 1)) As Object
Dim Text_i As Integer
For Text_i = 0 To (Object_Counter - 1)
    Set Ssobjs_text(Text_i) = TextObj(Text_i)
    If Text_i = Object_Counter - 1 Then
        Exit For
    End If
Next Text_i
SsetObj_text.additems Ssobjs_text
Selectionset_text_Access = True
End If
End Sub
Function StartAngleZiro(ByVal Angle1 As Variant, ByVal Angle2 As Variant, ByVal Arc_CenterX As Single, ByVal
Arc_CenterY As Single, _
    ByVal Arc_R As Single, MidAngleX As Variant, MidAngleY As Variant)
Dim StartAngle As Single, EndAngle As Single, FirstMidangle As Single, Midangle As Single
StartAngle = 0
EndAngle = 360 - Angle1 + Angle2
FirstMidangle = EndAngle / 2
Midangle = FirstMidangle + Angle1
If Midangle > 360 Then
    Midangle = Midangle - 360
    midangleinradian = Midangle * 3.141592 / 180
    MidAngleX = FormatNumber((Arc_CenterX + (Arc_R) * Cos(midangleinradian)), 3)
    MidAngleY = FormatNumber((Arc_CenterY + (Arc_R) * Sin(midangleinradian)), 3)
Else
    midangleinradian = Midangle * 3.141592 / 180

```

```

MidAngleX = FormatNumber((Arc_CenterX + (Arc_R) * Cos(midangleinradian)), 3)
MidAngleY = FormatNumber((Arc_CenterY + (Arc_R) * Sin(midangleinradian)), 3)
End If

End Function
Function findXYfromAngle(Uangle As Single, UCxarc As Single, UCyarc As Single, URarc As Single, UX As Single, UY As Single)
Dim midangleinradian As Single
midangleinradian = Uangle * 3.141592 / 180
UX = Val(FormatNumber((UCxarc + (URarc) * Cos(midangleinradian)), 3))
UY = Val(FormatNumber((UCyarc + (URarc) * Sin(midangleinradian)), 3))
End Function
Function localAngle(ByVal UAngle1 As Single, ByVal UAngle2 As Single, ByVal UAngle3 As Single, ByVal UAngle4 As Single, ByVal LoAngle1 As Single, ByVal LoAngle2 As Single, ByVal LoAngle3 As Single, ByVal LAngle4 As Single)
Dim StartAngle As Single, EndAngle As Single
StartAngle = 0
EndAngle = 360 - Angle1 + Angle4
Loangle2 = 360 - Angle1 + Angle2
Loangle3 = 360 - Angle1 + Angle3
End Function
'divide the circle at one point which is starting point of arc
Function DivideCircleStartpointArc(LCxCircle As Single, LCyCircle As Single, LRcircle As Single, LSXarc As Single, LSYarc As Single)
LineInter.AddItem "arc"
LineInter.AddItem (LCxCircle)
LineInter.AddItem (LCyCircle)
LineInter.AddItem (LRcircle)
LineInter.AddItem (LSXarc)
LineInter.AddItem (LSYarc)
LineInter.AddItem (LSXarc)
LineInter.AddItem (LSYarc)
End Function
Function ArcCircleBothintersecOnepointSameStart(LAngle3 As Single, LCXARC As Single, LCYARC As Single, LRARC As Single, LSXarc As Single, LSYarc As Single, LEXarc As Single, LEYarc As Single, LCxCircle As Single, LCyCircle As Single, LRcircle As Single)
Dim XfromAngle As Single, YfromAngle As Single
Call findXYfromAngle(LAngle3, LCXARC, LCYARC, LRARC, XfromAngle, YfromAngle)

LineInter.AddItem "arc"
LineInter.AddItem (LCXARC)
LineInter.AddItem (LCYARC)
LineInter.AddItem (LRARC)
LineInter.AddItem (XfromAngle)
LineInter.AddItem (YfromAngle)
LineInter.AddItem (LEXarc)
LineInter.AddItem (LEYarc)

LineInter.AddItem "arc"
LineInter.AddItem (LCxCircle)
LineInter.AddItem (LCyCircle)
LineInter.AddItem (LRcircle)
LineInter.AddItem (LSXarc)
LineInter.AddItem (LSYarc)
LineInter.AddItem (XfromAngle)
LineInter.AddItem (YfromAngle)

LineInter.AddItem "arc"
LineInter.AddItem (LCxCircle)
LineInter.AddItem (LCyCircle)
LineInter.AddItem (LRcircle)
LineInter.AddItem (XfromAngle)
LineInter.AddItem (YfromAngle)
LineInter.AddItem (LSXarc)
LineInter.AddItem (LSYarc)

End Function
Function ArcCircleBothintersecOnepointSameEnd(LAngle3 As Single, LCXARC, LCYARC As Single, LRARC As Single, LSXarc As Single, LSYarc As Single, _

```



```

LEXarc As Single, LEYarc As Single, LCxCircle As Single, LCyCircle As Single, LRcircle As
Single)
Dim XfromAngle As Single, YfromAngle As Single
Call findXYfromAngle(LAngle3, LCXARC, LCYARC, LRARC, XfromAngle, YfromAngle)

LineInter.AddItem "arc"
LineInter.AddItem (LCXARC)
LineInter.AddItem (LCYARC)
LineInter.AddItem (LRARC)
LineInter.AddItem (XfromAngle)
LineInter.AddItem (YfromAngle)
LineInter.AddItem (LEXarc)
LineInter.AddItem (LEYarc)

LineInter.AddItem "arc"
LineInter.AddItem (LCxCircle)
LineInter.AddItem (LCyCircle)
LineInter.AddItem (LRcircle)
LineInter.AddItem (XfromAngle)
LineInter.AddItem (YfromAngle)
LineInter.AddItem (LEXarc)
LineInter.AddItem (LEYarc)

LineInter.AddItem "arc"
LineInter.AddItem (LCxCircle)
LineInter.AddItem (LCyCircle)
LineInter.AddItem (LRcircle)
LineInter.AddItem (LEXarc)
LineInter.AddItem (LEYarc)
LineInter.AddItem (XfromAngle)
LineInter.AddItem (YfromAngle)

End Function
'divide the circle at one point which is Endpoint point of arc
Function DivideCircleEndpointArc(LCxCircle As Single, LCyCircle As Single, LRcircle As Single, LEXarc As Single, LEYarc
As Single)
    LineInter.AddItem "arc"
    LineInter.AddItem (LCxCircle)
    LineInter.AddItem (LCyCircle)
    LineInter.AddItem (LRcircle)
    LineInter.AddItem (LEXarc)
    LineInter.AddItem (LEYarc)
    LineInter.AddItem (LEXarc)
    LineInter.AddItem (LEYarc)
End Function
Function FindArcTipOutsideOrinsideCircle(ByVal LAngle1 As Single, ByVal TestAngle As Single, ByVal LAngle4 As Single,
LInside As Boolean)
Dim StartAngle As Single, LEndangle As Single, LTestAngle As Single
LInside = False

If LAngle1 > LAngle4 Then
    Lstartangle = 0
    LEndangle = 360 - LAngle1 + LAngle4
    LTestAngle = 360 - LAngle1 + TestAngle
    If LTestAngle < LEndangle Then
        ' if Linside=false thats means Arc has intersec the circle
        LInside = True
    End If
ElseIf TestAngle < LAngle4 Then
    LInside = True
End If
End Function
Function StartpointInsideOrRoutside(ByVal LAngle1 As Single, ByVal LAngle2 As Single, ByVal LAngle3 As Single, ByVal
LAngle4 As Single, LOneAngle2 As Boolean, LOtherAngle3 As Boolean, _
LNormalIntersection As Boolean)
Dim StartAngle As Single, EndAngle As Single, Asume As Single
LOneAngle2 = False
LOtherAngle3 = False
LNormalIntersection = False
If LAngle4 = 0 Then
    Asume = 360
Else
    Asume = LAngle4
End If
If LAngle1 > Asume Then

```

```
StartAngle = 0
EndAngle = 360 - LAngle1 + Asume
LAngle2 = 360 - LAngle1 + LAngle2
If LAngle2 >= 360 Then
    LAngle2 = LAngle2 - 360
End If
LAngle3 = 360 - LAngle1 + LAngle3
If LAngle3 >= 360 Then
    LAngle3 = LAngle3 - 360
End If
If StartAngle < LAngle2 And EndAngle > LAngle2 Then
    LOneAngle2 = True
End If
If StartAngle < LAngle3 And EndAngle > LAngle3 Then
    LOtherAngle3 = True
End If

If StartAngle < LAngle2 And LAngle3 > LAngle2 And EndAngle > LAngle2 Then
    LNormalIntersection = True
ElseIf StartAngle < LAngle2 And LAngle3 < LAngle2 And EndAngle > LAngle3 Then
    LNormalIntersection = False
End If

Else
    If LAngle1 < LAngle2 And Asume > LAngle2 Then
        LOneAngle2 = True
    End If
    If LAngle1 < LAngle3 And Asume > LAngle3 Then
        LOtherAngle3 = True
    End If
If LAngle1 < LAngle2 And LAngle3 > LAngle2 And Asume > LAngle2 Then
    LNormalIntersection = True
ElseIf LAngle1 < LAngle2 And LAngle3 < LAngle2 And Asume > LAngle3 Then
    LNormalIntersection = False
End If
End If

End Function
Function DivideArcAnglePoint(LCXARC As Single, LCYARC As Single, LRARC As Single, LXfromAngle, LYfromAngle,
LEXarc, LEYarc)
'Divide the Arc Startpoint xfromAngle, YfromAngle (for both Angle)
LineInter.AddItem "arc"
LineInter.AddItem (LCXARC)
LineInter.AddItem (LCYARC)
LineInter.AddItem (LRARC)
LineInter.AddItem (LXfromAngle)
LineInter.AddItem (LYfromAngle)
LineInter.AddItem (LEXarc)
LineInter.AddItem (LEYarc)
End Function
Function DivideCircleAnglePoint(LCxCircle As Single, LCyCircle As Single, LRcircle As Single, LXfromAngle As Single,
LYfromAngle As Single)
'divide the circle at XfromAngle, YfromAngle (for both Angle)
LineInter.AddItem "arc"
LineInter.AddItem (LCxCircle)
LineInter.AddItem (LCyCircle)
LineInter.AddItem (LRcircle)
LineInter.AddItem (LXfromAngle)
LineInter.AddItem (LYfromAngle)
LineInter.AddItem (LXfromAngle)
LineInter.AddItem (LYfromAngle)
End Function
Function DivideArcAtbyBothIntersection(LCXARC As Single, LCYARC As Single, LRARC As Single, LXfromAngle As
Single, LYfromAngle As Single, LXXfromAngle As Single, _
    LYYfromAngle As Single)

LineInter.AddItem "arc"
LineInter.AddItem (LCXARC)
LineInter.AddItem (LCYARC)
LineInter.AddItem (LRARC)
LineInter.AddItem (LXfromAngle)
LineInter.AddItem (LYfromAngle)
LineInter.AddItem (LXXfromAngle)
LineInter.AddItem (LYYfromAngle)
End Function
```

```

Function NormalOrAbnormalIntersection(LAngle1 As Single, LAngle2 As Single, LAngle3 As Single, LAngle4 As Single,
LNormalIntersection As Boolean)
LNormalIntersection = False
If LAngle2 > LAngle1 And LAngle2 < LAngle3 And LAngle2 < LAngle4 Then
    LNormalIntersection = True
Else
    LNormalIntersection = False
End If
End Function
Function Arc1SatisfiedByAngle(ByVal LAngle1 As Single, ByVal LAngle2 As Single, ByVal LAngle3 As Single, ByVal
LAngle4 As Single, _
    LCheckAngle1Arc1 As Boolean, LCheckAngle2Arc1 As Boolean, LCheckAngle1Arc1First As Boolean)
Dim StartAngle As Single, EndAngle As Single, Asume As Single
LCheckAngle1Arc1 = False
LCheckAngle2Arc1 = False
LCheckAngle1Arc1First = False
If LAngle4 = 0 Then
    Asume = 360
Else
    Asume = LAngle4
End If
If LAngle1 > Asume Then
    StartAngle = 0
    EndAngle = 360 - LAngle1 + LAngle4
    LAngle2 = 360 - LAngle1 + LAngle2
    If LAngle2 >= 360 Then
        LAngle2 = LAngle2 - 360
    End If
    LAngle3 = 360 - LAngle1 + LAngle3
    If LAngle3 >= 360 Then
        LAngle3 = LAngle3 - 360
    End If
    If LAngle2 >= StartAngle And LAngle2 <= EndAngle Then
        LCheckAngle1Arc1 = True
    End If
    If LAngle3 >= StartAngle And LAngle3 <= EndAngle Then
        LCheckAngle2Arc1 = True
    End If
    If LAngle2 < LAngle3 Then
        LCheckAngle1Arc1First = True
    End If
ElseIf LAngle1 = LAngle4 Then
    StartAngle = 0
    EndAngle = 360 - LAngle1 + LAngle4
    LAngle2 = 360 - LAngle1 + LAngle2
    If LAngle2 >= 360 Then
        LAngle2 = LAngle2 - 360
    End If
    LAngle3 = 360 - LAngle1 + LAngle3
    If LAngle3 >= 360 Then
        LAngle3 = LAngle3 - 360
    End If
    If LAngle2 >= StartAngle And LAngle2 <= EndAngle Then
        LCheckAngle1Arc1 = True
    End If
    If LAngle3 >= StartAngle And LAngle3 <= EndAngle Then
        LCheckAngle2Arc1 = True
    End If
    If LAngle2 < LAngle3 Then
        LCheckAngle1Arc1First = True
    End If
Else
    If LAngle2 >= LAngle1 And LAngle2 <= Asume Then
        LCheckAngle1Arc1 = True
    End If
    If LAngle3 >= LAngle1 And LAngle3 <= Asume Then
        LCheckAngle2Arc1 = True
    End If
    If LAngle2 < LAngle3 Then
        LCheckAngle1Arc1First = True
    End If
End If
End Function

```

```

Function FindIntersectionSatisfiedByAnotherArc(ByVal LAngle1 As Single, ByVal LCompareAngle As Single, ByVal LAngle4
As Single, LSatisfied As Boolean)
Dim StartAngle As Single, EndAngle As Single, Asume As Single
LSatisfied = False
If LAngle4 = 0 Then
    Asume = 360
Else
    Asume = LAngle4
End If

If LAngle1 >= Asume Then
    StartAngle = 0
    EndAngle = 360 - LAngle1 + Asume
    LCompareAngle = 360 - LAngle1 + LCompareAngle
    If LCompareAngle >= 360 Then
        LCompareAngle = LCompareAngle - 360
    End If
    differentcomparasume = Abs(LCompareAngle - EndAngle)
    If LCompareAngle > StartAngle And differentcomparasume > 0.04 And LCompareAngle < EndAngle Then

        LSatisfied = True
    End If
ElseIf LAngle1 = Asume Then
    If LAngle1 = LCompareAngle And LCompareAngle = LAngle4 Then
        LSatisfied = False
    Else
        LSatisfied = True
    End If

Else
    differentcomparasume = Abs(LCompareAngle - Asume)
    If LCompareAngle > LAngle1 And differentcomparasume > 0.04 And (LCompareAngle < Asume) Then

        LSatisfied = True
    End If
End If
End Function
Function FindIntersectionSatisfiedByAnotherArcForTangent(ByVal LAngle1 As Single, ByVal LCompareAngle As Single,
ByVal LAngle4 As Single, LSatisfied As Boolean)
Dim StartAngle As Single, EndAngle As Single, Asume As Single, AsumeCompareAngle As Single
LSatisfied = False
If LAngle4 = 0 Then
    Asume = 360
Else
    Asume = LAngle4
End If
AsumeCompare = LCompareAngle

If LAngle1 >= Asume Then
    StartAngle = 0
    EndAngle = 360 - LAngle1 + Asume
    'If EndAngle = 0 Then
    '    EndAngle = 360
    'End If
    If LCompareAngle = LAngle4 Then
        LCompareAngle = EndAngle
    ElseIf LCompareAngle = LAngle1 Then
        LCompareAngle = StartAngle
    Else
        LCompareAngle = 360 - LAngle1 + LCompareAngle
        If LCompareAngle >= 360 Then
            LCompareAngle = LCompareAngle - 360
        End If
    End If
    If LCompareAngle >= StartAngle And LCompareAngle <= EndAngle Then

        LSatisfied = True
    End If
ElseIf LAngle1 = Asume Then
    If LAngle1 = LCompareAngle And LCompareAngle = LAngle4 Then
        LSatisfied = False
    Else
        LSatisfied = True
    End If

```

```

End If

Else
    If LCompareAngle >= LAngle1 And LCompareAngle <= Asume Then

        LSatisfied = True
    End If
End If
End Function
Function GeneralLine(Lx111 As Single, Ly111 As Single, Lx122 As Single, Ly122 As Single)
LineInter.AddItem "line"
LineInter.AddItem (Lx111)
LineInter.AddItem (Ly111)
LineInter.AddItem (Lx122)
LineInter.AddItem (Ly122)

End Function
Function GeneralARC(LCXARC As Single, LCYARC As Single, LRARC As Single, LStartX As Single, LStartY As Single,
LEndX As Single, LEndY As Single)

LineInter.AddItem "arc"
LineInter.AddItem (LCXARC)
LineInter.AddItem (LCYARC)
LineInter.AddItem (LRARC)
LineInter.AddItem (LStartX)
LineInter.AddItem (LStartY)
LineInter.AddItem (LEndX)
LineInter.AddItem (LEndY)
End Function
Function ArcBothEndpointChecking(ByVal LSXArc1 As Single, ByVal LSYArc1 As Single, ByVal LEXArc1 As Single,
ByVal LEYArc1 As Single,
ByVal LSXArc2 As Single, ByVal LSYArc2 As Single, ByVal LEXArc2 As Single, ByVal LEYArc2 As
Single, _
LAllpointSame As Boolean, LOnepointCheck As Boolean)

LAllpointSame = False
LOnepointCheck = False
If (LSXArc1 = LSXArc2 And LSYArc1 = LSYArc2) And (LEXArc1 = LEXArc2 And LEYArc1 = LEYArc2) Then
    LAllpointSame = True
ElseIf (LSXArc1 = LEXArc2 And LSYArc1 = LEYArc2) And (LEXArc1 = LSXArc2 And LEYArc1 = LSYArc2) Then
    LAllpointSame = True
End If
If (LSXArc1 = LSXArc2 And LSYArc1 = LSYArc2) Then
    LOnepointCheck = True
ElseIf (LEXArc1 = LEXArc2 And LEYArc1 = LEYArc2) Then
    LOnepointCheck = True
ElseIf (LSXArc1 = LEXArc2 And LSYArc1 = LEYArc2) Then
    LOnepointCheck = True
ElseIf (LEXArc1 = LSXArc2 And LEYArc1 = LSYArc2) Then
    LOnepointCheck = True
End If
End Function

Function CheckHowManyCircleleft()
Dim OnlyCircle As Integer, NotOnlyCircle As Integer
OnlyCircle = 0
For i = 0 To LineInter.ListCount - 1
    If LineInter.List(i) Like "circle" Then
        OnlyCircle = OnlyCircle + 1
    End If
Next i
NotOnlyCircle = 0

For i = 0 To LineInter.ListCount - 1
    If List1.List(i) Like "circle*" Then
        NotOnlyCircle = NotOnlyCircle + 1
    End If
Next i
End Function
Function Draw_Circle_selectionset()
If TEST_SELECTIONSET_circle_Access = True Then
    acadDoc.SelectionSets.Item("TEST_SELECTIONSET_circle").Delete
    Call Draw_Circle

```

```

Else
    Call Draw_Circle
End If
End Function
Function Draw_Arc_selectionset()
    If TEST_SELECTIONSET_Arc_Access = True Then
        If Total_ARC > 0 Then
            acadDoc.SelectionSets.Item("TEST_SELECTIONSET_arc").Delete
            Call Draw_Arc
        End If
    Else
        Call Draw_Arc
    End If
End Function
Function Draw_Line_selectionset()
    If TEST_SELECTIONSET_Line_Access = True Then
        acadDoc.SelectionSets.Item("TEST_SELECTIONSET_line").Delete
        Call Draw_Line
    Else
        Call Draw_Line
    End If
End Function
Function LookforObjectNumber()
    Dim counter As Integer, MemriseCounter As Integer, Line_i As Integer, Circle_i As Integer, arc_i As Integer
    counter = 0
    ReDim AllobjectName(Total_Object - 1)
    If Total_Line > 0 Then
        For Line_i = 0 To Total_Line - 1
            AllobjectName(Line_i) = "line" & (counter)
            counter = counter + 1
        Next Line_i
    End If
    MemriseCounter = counter
    If Total_Circle > 0 Then
        For Circle_i = 0 To Total_Circle - 1
            AllobjectName(counter) = "circle" & (counter - MemriseCounter)
            counter = counter + 1
        Next Circle_i
    End If
    MemriseCounter = counter
    If Total_ARC > 0 Then
        For arc_i = 0 To Total_ARC - 1
            AllobjectName(counter) = "arc" & (counter - MemriseCounter)
            counter = counter + 1
        Next arc_i
    End If
End Function

```

```

Function LineSearch(ByVal storenumber As Integer, Line1full As Boolean, Line2full As Boolean)
    Dim Linecounter As Integer
    Linecounter = 0
    If Line1full = False Then
        For i = 0 To LineInter.ListCount
            If LineInter.List(i) = "line" Then
                If storenumber = Linecounter Then
                    Line1(0) = i
                    Line1(1) = FormatNumber(LineInter.List(i + 1), 3)
                    Line1(2) = FormatNumber(LineInter.List(i + 2), 3)
                    Line1(3) = FormatNumber(LineInter.List(i + 3), 3)
                    Line1(4) = FormatNumber(LineInter.List(i + 4), 3)
                End If
                Linecounter = Linecounter + 1
            End If
        Next i
        Line1full = True
    Else
        For i = 0 To LineInter.ListCount
            If LineInter.List(i) = "line" Then
                If storenumber = Linecounter Then
                    Line2(0) = i
                    Line2(1) = LineInter.List(i + 1)
                    Line2(2) = LineInter.List(i + 2)
                    Line2(3) = LineInter.List(i + 3)
                    Line2(4) = LineInter.List(i + 4)
                End If
            End If
        Next i
    End If
End Function

```

```

        End If
        Linecounter = Linecounter + 1
    End If
Next i
Line2full = True
End If
End Function
Function Circlesearch(ByVal storenumber As Integer, Circle1full As Boolean, Circle2full As Boolean)
Dim Circlecounter As Integer
Circlecounter = 0
If Circle1full = False Then
    For i = 0 To LineInter.ListCount
        If LineInter.List(i) = "circle" Then
            If storenumber = Circlecounter Then
                Circle1(0) = i
                Circle1(1) = LineInter.List(i + 1)
                Circle1(2) = LineInter.List(i + 2)
                Circle1(3) = LineInter.List(i + 3)
            End If
            Circlecounter = Circlecounter + 1
        End If
    Next i
    Circle1full = True
Else
    For i = 0 To LineInter.ListCount
        If LineInter.List(i) = "circle" Then
            If storenumber = Circlecounter Then
                Circle2(0) = i
                Circle2(1) = LineInter.List(i + 1)
                Circle2(2) = LineInter.List(i + 2)
                Circle2(3) = LineInter.List(i + 3)
            End If
            Circlecounter = Circlecounter + 1
        End If
    Next i
    Circle2full = True
End If
End Function
Function Arcsearch(ByVal storenumber As Integer, Arc1full As Boolean, Arc2full As Boolean)
Dim Arccounter As Integer
Arccounter = 0
If Arc1full = False Then
    For i = 0 To LineInter.ListCount
        If LineInter.List(i) = "arc" Then
            If storenumber = Arccounter Then
                Arc1(0) = i
                Arc1(1) = LineInter.List(i + 1)
                Arc1(2) = LineInter.List(i + 2)
                Arc1(3) = LineInter.List(i + 3)
                Arc1(4) = LineInter.List(i + 4)
                Arc1(5) = LineInter.List(i + 5)
                Arc1(6) = LineInter.List(i + 6)
                Arc1(7) = LineInter.List(i + 7)
            End If
            Arccounter = Arccounter + 1
        End If
    Next i
    Arc1full = True
Else
    For i = 0 To LineInter.ListCount
        If LineInter.List(i) = "arc" Then
            If storenumber = Arccounter Then
                Arc2(0) = i
                Arc2(1) = LineInter.List(i + 1)
                Arc2(2) = LineInter.List(i + 2)
                Arc2(3) = LineInter.List(i + 3)
                Arc2(4) = LineInter.List(i + 4)
                Arc2(5) = LineInter.List(i + 5)
                Arc2(6) = LineInter.List(i + 6)
                Arc2(7) = LineInter.List(i + 7)
            End If
            Arccounter = Arccounter + 1
        End If
    Next i
    Arc2full = True
End If
End Function

```

```

End If
End Function
Function Chamfer(ByVal inter_x As Single, ByVal inter_y As Single)
Dim F_x As Single, F_y As Single, Finalx1 As Single, Finaly1 As Single, Finalx2 As Single, Finaly2 As Single
Call FinalXY(inter_x, inter_y, Line1(1), Line1(2), Line1(3), Line1(4), F_x, F_y)
Finalx1 = F_x
Finaly1 = F_y
If Line1(1) = inter_x And Line1(2) = inter_y Then
    LineInter.List(Line1(0) + 1) = Finalx1
    LineInter.List(Line1(0) + 2) = Finaly1
Else
    LineInter.List(Line1(0) + 3) = Finalx1
    LineInter.List(Line1(0) + 4) = Finaly1
End If
Call FinalXY(inter_x, inter_y, Line2(1), Line2(2), Line2(3), Line2(4), F_x, F_y)
Finalx2 = F_x
Finaly2 = F_y
If Line2(1) = inter_x And Line2(2) = inter_y Then
    LineInter.List(Line2(0) + 1) = Finalx2
    LineInter.List(Line2(0) + 2) = Finaly2
Else
    LineInter.List(Line2(0) + 3) = Finalx2
    LineInter.List(Line2(0) + 4) = Finaly2
End If
LineInter.AddItem "line"
LineInter.AddItem Finalx1
LineInter.AddItem Finaly1
LineInter.AddItem Finalx2
LineInter.AddItem Finaly2
Call DeleteSeleReloadArrayDrawobject
Call Objectlist
End Function
Function Fillet(ByVal inter_x As Single, ByVal inter_y As Single)
Dim F_x As Single, F_y As Single, Finalx1 As Single, Finaly1 As Single, Finalx2 As Single, Finaly2 As Single
Dim M_Firstline As Single, C_Firstline As Single
Dim M_SEcondline As Single, C_SEcondline As Single, Center_X As Single, Center_Y As Single
Dim Angle1 As Double, Angle2 As Double
'Line1XHoriy this mean line 1 is Horizontal with Y axes,
Dim Line1XParay As Boolean, Line1YParay As Boolean, Line2XParay As Boolean, Line2YParay As Boolean
Dim NewX1 As Single, NewY1 As Single, NewX2 As Single, NewY2 As Single
Call FinalXY(inter_x, inter_y, Line1(1), Line1(2), Line1(3), Line1(4), F_x, F_y)
Finalx1 = F_x
Finaly1 = F_y
If Line1(1) = inter_x And Line1(2) = inter_y Then
    LineInter.List(Line1(0) + 1) = Finalx1
    LineInter.List(Line1(0) + 2) = Finaly1
Else
    LineInter.List(Line1(0) + 3) = Finalx1
    LineInter.List(Line1(0) + 4) = Finaly1
End If

Call FinalXY(inter_x, inter_y, Line2(1), Line2(2), Line2(3), Line2(4), F_x, F_y)
Finalx2 = F_x
Finaly2 = F_y
If Line2(1) = inter_x And Line2(2) = inter_y Then
    LineInter.List(Line2(0) + 1) = Finalx2
    LineInter.List(Line2(0) + 2) = Finaly2
Else
    LineInter.List(Line2(0) + 3) = Finalx2
    LineInter.List(Line2(0) + 4) = Finaly2
End If

If (Line1(1) - Line1(3)) = 0 Then
    'then Multiply the equation by (x1-x2) then put value of (x1-x2)
Else
    M_Firstline = (Line1(2) - Line1(4)) / (Line1(1) - Line1(3))
    'C_firstline = LineY1 - LineX1 * ((LineY1 - LineY2) / (LineX1 - LineX2))
End If

If (Line2(1) - Line2(3)) = 0 Then
    'then Multiply the equation by (x1-x2) then put value of (x1-x2)
Else
    M_SEcondline = (Line2(2) - Line2(4)) / (Line2(1) - Line2(3))

```



```

'C_firstline = LineY1 - LineX1 * ((LineY1 - LineY2) / (LineX1 - LineX2))
'The equation of the line paralal to Firstline
'Y_firstline = M_Secondline * X_firstline + C_firstline
End If
If (Line1(1) - Line1(3)) = 0 Then

    Line1XParay = True
End If
If Line1(2) - Line1(4) = 0 Then

    Line1YParax = True
End If

If Line2(1) - Line2(3) = 0 Then

    Line2XParay = True
End If
If Line2(2) - Line2(4) = 0 Then

    Line2YParax = True
End If
'both lines are Horizontal with the both Axes
If Line1YParax = True And Line2XParay = True Then
    'Draw the fillet considering the Center_x and Center_y as a center
    Center_X = Finalx1
    Center_Y = Finaly2
ElseIf Line1XParay = True And Line2YParax = True Then
    'Draw the fillet considering the Center_x and Center_y as a center
    Center_X = Finalx2
    Center_Y = Finaly1

ElseIf Line1XParay = True And Line2YParax = False Then
    M_Secondline = (Line2(2) - Line2(4)) / (Line2(1) - Line2(3))
    C_Secondline = Finaly1 - M_Secondline * Finalx1
    Center_X = Finalx2
    Center_Y = Center_X * M_Secondline + C_Secondline
ElseIf Line1XParay = False And Line2YParax = True Then
    M_Secondline = (Line1(2) - Line1(4)) / (Line1(1) - Line1(3))
    C_Secondline = Finaly2 - M_Secondline * Finalx2
    Center_Y = Finaly1
    Center_X = (Center_Y - C_Secondline) / M_Secondline

ElseIf Line1YParax = True And Line2XParay = False Then
    M_Secondline = (Line2(2) - Line2(4)) / (Line2(1) - Line2(3))
    C_Secondline = Finaly1 - M_Secondline * Finalx1
    Center_X = (Finaly2 - C_Secondline) / M_Secondline
    Center_Y = Finaly2
ElseIf Line1YParax = False And Line2XParay = True Then
    M_Secondline = (Line1(2) - Line1(4)) / (Line1(1) - Line1(3))
    C_Secondline = Finaly2 - M_Secondline * Finalx2
    Center_X = Finalx1
    Center_Y = Center_X * M_Secondline + C_Secondline
Else
    C_Firstline = Finaly2 - (M_Firstline * Finalx2)
    C_Secondline = Finaly1 - (M_Secondline * Finalx1)
    Center_X = (C_Secondline - C_Firstline) / (M_Firstline - M_Secondline)
    Center_Y = M_Secondline * ((C_Secondline - C_Firstline) / (M_Firstline - M_Secondline)) + C_Secondline
End If

NewX1 = Finalx1 - Center_X
NewY1 = Finaly1 - Center_Y
Angle1 = FormatNumber(atan2(NewX1, NewY1), 3)

NewX2 = Finalx2 - Center_X
NewY2 = Finaly2 - Center_Y
Angle2 = FormatNumber(atan2(NewX2, NewY2), 3)

If Position(0) = "line1" Then
    'If Angle1 > Angle2 Then
    LineInter.AddItem "arc"
    LineInter.AddItem Center_X
    LineInter.AddItem Center_Y
    LineInter.AddItem RadiusFillet
    LineInter.AddItem Finalx1

```

```

LineInter.AddItem Finaly1
LineInter.AddItem Finalx2
LineInter.AddItem Finaly2

Else
    LineInter.AddItem "arc"
    LineInter.AddItem Center_X
    LineInter.AddItem Center_Y
    LineInter.AddItem RadiusFillet
    LineInter.AddItem Finalx2
    LineInter.AddItem Finaly2
    LineInter.AddItem Finalx1
    LineInter.AddItem Finaly1

End If
'End If
Call DeleteSeleReloadArrayDrawobject
Call Objectlist
End Function
Function FinalXY(ByVal inter_x As Single, ByVal inter_y As Single, ByVal LineX1 As Single, ByVal LineY1 As Single,
ByVal LineX2 As Single, _
    ByVal LineY2 As Single, FinalX As Single, FinalY As Single)
Dim Up_x As Boolean, Up_y As Boolean
Dim M_Firstline As Single, C_Firstline As Single, DelltaX1 As Single, DelltaY1 As Single
Dim M_Secondline As Single, C_Secondline As Single
Dim Distance As Single, Ant As Single
DelltaX1 = 0
DelltaY1 = 0
'FinalX As Single, FinalY As Single
If radiouTrueorFalse = True Then
    RadiusFillet = InputBox("Enter the radius of the fillet", " ", 0, 0)
    radiouTrueorFalse = False
End If

Distance = Sqr((LineX1 - LineX2) ^ 2 + (LineY1 - LineY2) ^ 2)
If inter_x = LineX1 Then
    DelltaX1 = LineX2 - inter_x
ElseIf inter_x = LineX2 Then
    DelltaX1 = LineX1 - inter_x
End If
If inter_y = LineY1 Then
    DelltaY1 = LineY2 - inter_y
ElseIf inter_y = LineY2 Then
    DelltaY1 = LineY1 - inter_y
End If
'For i = 0.001 To Distance / 0.001 Step 0.001
For i = 0.001 To Distance Step 0.001
    If Ant >= RadiusFillet Then
        Exit For
    End If
    FinalX = inter_x + DelltaX1 * i
    FinalY = inter_y + DelltaY1 * i
    Ant = Sqr((FinalX - inter_x) ^ 2 + (FinalY - inter_y) ^ 2)
Next i

End Function
Function DeleteSeleReloadArrayDrawobject()
Call Delete_Selectionset
Call ReloadLine_Array
Call ReloadCircle_Array
Call ReloadArc_Array
'Call Draw_Line
If Total_Line = 0 And TEST_SELECTIONSET_Line_Access = True Then
    acadDoc.SelectionSets.Item("TEST_SELECTIONSET_line").Delete
    TEST_SELECTIONSET_Line_Access = False
Else
    Call Draw_Line_selectionset
End If
'Call Draw_Circle
If Total_Circle = 0 And TEST_SELECTIONSET_circle_Access = True Then
    acadDoc.SelectionSets.Item("TEST_SELECTIONSET_circle").Delete
    TEST_SELECTIONSET_circle_Access = False
Else
    Call Draw_Circle_selectionset

```

```

End If

'Call Draw_Arc
If Total_ARC = 0 And TEST_SELECTIONSET_circle_Access = True And TEST_SELECTIONSET_Arc_Access = True Then
    acadDoc.SelectionSets.Item("TEST_SELECTIONSET_arc").Delete
    TEST_SELECTIONSET_Arc_Access = False
Else
    Call Draw_Arc_selectionset
End If
Call Draw_ObjectNumber
acadDoc.sendcommand " _zoom a "
'Me.MousePointer = 0
Form1.Show
End Function
'Rearranging line Ascending to Descending
Function Linerearrange(LineX1 As Single, LineY1 As Single, LineX2 As Single, LineY2 As Single)

d1 = ((LineX1) ^ 2 + (LineY1) ^ 2) ^ 1 / 2
d2 = ((LineX2) ^ 2 + (LineY2) ^ 2) ^ 1 / 2
If d1 > d2 Then
    swapvx = LineX1
    LineX1 = LineX2
    LineX2 = swapvx

    swapvy = LineY1
    LineY1 = LineY2
    LineY2 = swapvy
End If
End Function
Function SameWithendpointandIntersection(ByVal X1 As Single, ByVal Y1 As Single, ByVal X2 As Single, ByVal Y2 As Single, _
    ByVal inter_x As Single, ByVal inter_y As Single) As Boolean

If X1 = inter_x And Y1 = inter_y Then
    SameWithendpointandIntersection = True
ElseIf X2 = inter_x And Y2 = inter_y Then
    SameWithendpointandIntersection = True
End If
End Function
Function ReadDXF( _
    ByVal dxffile As String, ByVal strSection As String, _
    ByVal strObject As String, ByVal strCodeList As String)
Dim tmpCode, lastObj As String
Open dxffile For Input As #1
' Get the first code/value pair
codes = ReadCodes
' Loop through the whole file until the "EOF" line
While codes(1) <> "EOF"
    ' If the group code is '0' and the value is 'SECTION' ..
    If codes(0) = "0" And codes(1) = "SECTION" Then
        ' This must be a new section, so get the next
        ' code/value pair.
        codes = ReadCodes()
        ' If this section is the right one ..
        If codes(1) = strSection Then
            ' Get the next code/value pair and ..
            codes = ReadCodes
            ' Loop through this section until the 'ENDSEC'
            While codes(1) <> "ENDSEC"
                ' While in a section, all '0' codes indicate
                ' an object. If you find a '0' store the
                ' object name for future use.
                If codes(0) = "0" Then lastObj = codes(1)
                ' If this object is one you're interested in
                If lastObj = strObject Then
                    ' Surround the code with commas
                    tmpCode = "," & codes(0) & ","
                    ' If this code is in the list of codes ..
                    If InStr(strCodeList, tmpCode) Then
                        ' Append the return value.
                        ReadDXF = ReadDXF & _
                            codes(0) & "," & codes(1)
                        '& vbCrLf
                    End If
                End If
            End While
            'AllData = ReadDXF
        End If
    End If
End While
Close #1
End Function

```

```

        Debug.Print ReadDXF
    End If
End If
' Read another code/value pair
codes = ReadCodes
Wend
End If
Else
    codes = ReadCodes
End If
Wend
Close #1
End Function
' ReadCodes reads two lines from an open file and returns a two item
' array, a group code and its value. As long as a DXF file is read
' two lines at a time, all should be fine. However, to make your
' code more reliable, you should add some additional error and
' sanity checking.

Function ReadCodes() As Variant
    Dim codeStr, valStr As String
    Line Input #1, codeStr
    Line Input #1, valStr
    ' Trim the leading and trailing space from the code
    ReadCodes = Array(Trim(codeStr), valStr)
End Function

Private Sub Menudraw_Click()
    Call Main
End Sub

Private Sub Menuopen_Click()
    Call OpenFile
End Sub

Private Sub Menusave_Click()
    Call SaveAll
End Sub

Private Sub MenuShowSection_Click()
    Call Edit
End Sub

Private Sub MenuToggle_Click()
    Call Toggol
End Sub
'This function is checking whether intersection lay on the start point or endpoint
Function CheckForIntersectionMatchingwithEndorStartpointARC(StartArcx, StartArcy, EndArcx, EndArcy, int_X1, int_Y1) As
Boolean
Dim FunctionStartArcx As Single, FStartArcy As Single, FEndArcx As Single, FEndArcy As Single, Fint_x1 As Single, Fint_y1
As Single
FunctionStartArcx = Val(FormatNumber(StartArcx, 0))
FStartArcy = Val(FormatNumber(StartArcy, 0))
FEndArcx = Val(FormatNumber(EndArcx, 0))
FEndArcy = Val(FormatNumber(EndArcy, 0))
Fint_x1 = Val(FormatNumber(int_X1, 0))
Fint_y1 = Val(FormatNumber(int_Y1, 0))
If (FunctionStartArcx = Fint_x1 And FStartArcy = Fint_y1) Or (FEndArcx = Fint_x1 And FEndArcy = Fint_y1) Then
    CheckForIntersectionMatchingwithEndorStartpointARC = True
End If
End Function
'This function is checking whether intersection point same with atartpoint or end point
Function CheckForIntersectionMatchingwithEndorStartpointLINE(x111, y111, x122, y122, int_X1, int_Y1) As Boolean
Dim Fx111 As Single, Fy111 As Single, Fx122 As Single, Fy122 As Single, Fint_x1 As Single, Fint_y1 As Single
Dim CompareNumber As Single
CompareNumber = Abs(x111 - int_X1)
If CompareNumber < 0.5 Then
    int_X1 = x111
End If
CompareNumber = Abs(y111 - int_Y1)
If CompareNumber < 0.5 Then
    int_Y1 = y111
End If

```

```

CompareNumber = Abs(x122 - int_X1)
If CompareNumber < 0.5 Then
    int_X1 = x122
End If
CompareNumber = Abs(y122 - int_Y1)
If CompareNumber < 0.5 Then
    int_Y1 = y122
End If

If (x111 = int_X1 And y111 = int_Y1) Or (x122 = int_X1 And y122 = int_Y1) Then
    CheckForIntersectionMatchingWithEndorStartpointLINE = True
End If
End Function

Function ArcLineFillet(ByVal inter_x As Single, ByVal inter_y As Single)
    Dim F_x As Single, F_y As Single, Finalx1 As Single, Finaly1 As Single, Finalx2 As Single, Finaly2 As Single
    Dim M_Firstline As Single, C_Firstline As Single
    Dim M_Secondline As Single, C_Secondline As Single, Center_X As Single, Center_Y As Single
    Dim Angle1 As Double, Angle2 As Double
    Dim Midx As Single, MidY As Single, F_x1 As Single, F_y1 As Single
    Dim NewUpX As Single, NewUpY As Single, NewDownX As Single, NewDownY As Single
    'F_x1 = 1
    'F_y1 = 2
    'Line1XHoriy this mean line 1 is Horizontal with Y axes,
    Dim Line1XParay As Boolean, Line1YParax As Boolean, Line2XParay As Boolean, Line2YParax As Boolean
    Dim NewX1 As Single, NewY1 As Single, NewX2 As Single, NewY2 As Single
    Call FinalXY(inter_x, inter_y, Line1(1), Line1(2), Line1(3), Line1(4), F_x, F_y)
    Finalx1 = F_x
    Finaly1 = F_y
    If Line1(1) = inter_x And Line1(2) = inter_y Then
        LineInter.List(Line1(0) + 1) = Finalx1
        LineInter.List(Line1(0) + 2) = Finaly1
    Else
        LineInter.List(Line1(0) + 3) = Finalx1
        LineInter.List(Line1(0) + 4) = Finaly1
    End If
    If Line1(1) = inter_x And Line1(2) = inter_y Then
        LineInter.List(Line1(0) + 1) = Finalx1
        LineInter.List(Line1(0) + 2) = Finaly1
    Else
        LineInter.List(Line1(0) + 3) = Finalx1
        LineInter.List(Line1(0) + 4) = Finaly1
    End If

    Call ArcFinalXY(inter_x, inter_y, Arc1(1), Arc1(2), Arc1(3), Arc1(4), Arc1(5), Arc1(6), Arc1(7), F_x1, F_y1)
    Midx = (Finalx1 + F_x1) / 2
    MidY = (Finaly1 + F_y1) / 2

    Call FindPerpendicularXY(Midx, MidY, F_x, F_y, F_x1, F_y1, NewUpX, NewUpY, NewDownX, NewDownY)

    firstinterX = Arc1(4) - Arc1(1)
    FirstinterY = Arc1(5) - Arc1(2)
    StartAngle1 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

    firstinterX = Arc1(6) - Arc1(1)
    FirstinterY = Arc1(7) - Arc1(2)
    EndAngle1 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

    firstinterX = inter_x - Arc1(1)
    FirstinterY = inter_y - Arc1(2)
    IntersectionAngle = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

    If StartAngle1 = IntersectionAngle Then
        LineInter.List(Arc1(0) + 4) = F_x1
        LineInter.List(Arc1(0) + 5) = F_y1
    ElseIf EndAngle1 = IntersectionAngle Then
        LineInter.List(Arc1(0) + 6) = F_x1
        LineInter.List(Arc1(0) + 7) = F_y1
    End If
    If Position(0) = "arc" Then
        LineInter.AddItem "arc"
        LineInter.AddItem NewUpX
    End If
End Function

```

```

LineInter.AddItem NewUpY
LineInter.AddItem RadiusFillet
LineInter.AddItem F_x1
LineInter.AddItem F_y1
LineInter.AddItem F_x
LineInter.AddItem F_y
Elseif Position(0) Like "line*" Then
LineInter.AddItem "arc"
LineInter.AddItem NewDownX
LineInter.AddItem NewDownY
LineInter.AddItem RadiusFillet
LineInter.AddItem F_x
LineInter.AddItem F_y
LineInter.AddItem F_x1
LineInter.AddItem F_y1
End If
'End If
Call DeleteSeleReloadArrayDrawobject
Call Objectlist
End Function
Function ArcFinalXY(ByVal inter_x As Single, ByVal inter_y As Single, ByVal CenterX As Single, ByVal CenterY As Single,
ByVal Radius As Single, ByVal StartX As Single, ByVal StartY As Single, _
ByVal EndX As Single, ByVal EndY As Single, CurrentX As Single, CurrentY As Single)

Dim StartAngle1 As Single, EndAngle1 As Single, firstinterX As Single, FirstinterY As Single, IntersectionAngle As Single,
Distance As Single
Dim Ant As Single
Ant = 0
'Findingout ANGLE for the ARC' start point and end point.

firstinterX = StartX - CenterX
FirstinterY = StartY - CenterY
StartAngle1 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = EndX - CenterX
FirstinterY = EndY - CenterY
EndAngle1 = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

firstinterX = inter_x - CenterX
FirstinterY = inter_y - CenterY
IntersectionAngle = Val(FormatNumber(atan2(firstinterX, FirstinterY), 3))

If IntersectionAngle = StartAngle1 Then
'reduce the Arc from Startpoint
Distance = StartAngle1 * 3.141592 / 180
For i = 0.001 To RadiusFillet Step 0.001
If Ant >= RadiusFillet Then
Exit For
End If
CurrentX = CenterX + Radius * (Cos(Distance + i))
CurrentY = CenterY + Radius * (Sin(Distance + i))
Ant = Sqr((StartX - CurrentX) ^ 2 + (CurrentY - StartY) ^ 2)
Next i

Elseif IntersectionAngle = EndAngle1 Then
'reduce the Arc from End point
Distance = EndAngle1 * 3.141592 / 180
For i = 0.001 To RadiusFillet Step 0.001
If Ant >= RadiusFillet Then
Exit For
End If
CurrentX = CenterX + Radius * (Cos(Distance - i))
CurrentY = CenterY + Radius * (Sin(Distance - i))
Ant = Sqr((EndX - CurrentX) ^ 2 + (CurrentY - EndY) ^ 2)
Next i
End If

MsgBox "wait"

End Function
Function signCos(ByVal angle As Single, ByVal se As Boolean) As Integer
If (se = True) Then

```

```

    If (angle < 180) Then signCos = -1
    Else
        signCos = 1
    End If
End If
End Function
Function FindPerpendicularXY(ByVal Midx As Single, ByVal MidY As Single, ByVal F_x As Single, ByVal F_y As Single, _
    ByVal F_x1 As Single, ByVal F_y1 As Single, ByRef NewUpX As Single, ByRef NewUpY As Single, _
    ByRef NewDownX As Single, ByRef NewDownY As Single)
    Dim DelX As Single, DelY As Single, Ant As Single
    Dim t As Single, Distance As Single
    Ant = 0
    DelX = F_x - F_x1
    DelY = F_y - F_y1
    Ant = 0

    For i = 0.001 To RadiusFillet Step 0.00001
        If Ant >= RadiusFillet Then
            MsgBox NewUpX & NewUpY
            Exit For
        End If
        NewUpX = Midx + i * (-DelY)
        NewUpY = MidY + i * (DelX)
        Ant = Sqr((F_x - NewUpX) ^ 2 + (NewUpY - F_y) ^ 2)
    Next i
    Ant = 0
    For i = 0.001 To RadiusFillet Step 0.001
        If Ant >= RadiusFillet Then
            MsgBox NewDownX & NewDownY
            Exit For
        End If
        NewDownX = Midx + i * (DelY)
        NewDownY = MidY + i * (-DelX)
        Ant = Sqr((F_x - NewDownX) ^ 2 + (NewDownY - F_y) ^ 2)
    Next i
    MsgBox "wait"
End Function
Private Function FindBoundaryForArc(X1 As Single, Y1 As Single, X2 As Single, Y2 As Single, int_X1 As Single, int_Y1 As
Single, _
    int_x2 As Single, int_Y2 As Single, FirstPoint As Boolean, SecondPoint As Boolean, ByRef
interFirstpoint As Boolean, ByRef InterSecondpoint As Boolean)
    If X1 > X2 Then
        tempFx = X2
        tempSx = X1
    Else
        tempFx = X1
        tempSx = X2
    End If

    If Y1 > Y2 Then
        tempFy = Y2
        tempSy = Y1
    Else
        tempFy = Y1
        tempSy = Y2
    End If
    If interFirstpoint = True Then
        If ((int_X1 >= tempFx) And (int_X1 <= tempSx)) And ((int_Y1 >= tempFy) And (int_Y1 <= tempSy)) Then
            FirstPoint = True
        Else
            FirstPoint = False
            interFirstpoint = False
        End If
    Else
        FirstPoint = False
    End If
    If InterSecondpoint = True Then
        If ((int_x2 >= tempFx) And (int_x2 <= tempSx)) And ((int_Y2 >= tempFy) And (int_Y2 <= tempSy)) Then
            SecondPoint = True
        Else
            SecondPoint = False
            InterSecondpoint = False
        End If
    Else

```

```

    SecondPoint = False
End If

'End Function
End Function
Function CheckOnlyforArc(ByVal Arc1X As Single, ByVal Arc1Y As Single, ByVal CX As Single, ByVal CY As Single,
ByVal R As Single, _
    ByVal Sx As Single, ByVal Sy As Single, ByVal Ex As Single, ByVal Ey As Single, Arc1Satisfied As
Boolean)

Dim StartX As Single, StartY As Single, EndX As Single, EndY As Single, Startanglearc1 As Single, Endanglearc1 As Single, X
As Single, Y As Single, StartAngleArc2 As Single
Dim EndAngleArc2 As Single, Returnvalue As Boolean
Arc1Satisfied = False
NewCircle1Satisfied = False
Arc2Satisfied = False
NewCircle2Satisfied = False
    StartX = Sx - CX
    StartY = Sy - CY
    Startanglearc1 = Val(FormatNumber(atan2(StartX, StartY), 3))

    EndX = Ex - CX
    EndY = Ey - CY
    Endanglearc1 = Val(FormatNumber(atan2(EndX, EndY), 3))

    X = Arc1X - CX
    Y = Arc1Y - CY
    line1angle = Val(FormatNumber(atan2(X, Y), 3))

    'Call checkintersectionPointbetweenStartangleAndEndAngle(Startanglearc1, Endanglearc1, line1angle, Returnvalue)
    Call FindIntersectionSatisfiedByAnotherArcForTangent(Startanglearc1, line1angle, Endanglearc1, Returnvalue)
    Arc1Satisfied = Returnvalue
End Function
Function TangentForCircleArc()
Dim MidxNewCircle As Single, MidyNewCircle As Single, RNewCircle As Single, Firstintersection_x1 As Single,
Firstintersection_y1 As Single
Dim Firstintersection_x2, Firstintersection_y2
Dim Secondintersection_x1 As Single, Secondintersection_y1 As Single
Dim Secondintersection_x2 As Single, Secondintersection_y2 As Single
Dim int_X1 As Single, int_Y1 As Single, int_x2 As Single, int_Y2 As Single
Dim Innint_X1 As Single, Innint_Y1 As Single, Innint_x2 As Single, Innint_Y2 As Single
Dim Delta_X As Single, Delta_Y As Single, DeltaXin1 As Single, DeltaYin1 As Single, DeltaXin2 As Single, DeltaYin2 As
Single
Dim Dot_1 As Single, Dot_2 As Single
Dim Arc1Satisfied As Boolean, NewCircle1Satisfied As Boolean, Arc2Satisfied As Boolean, NewCircle2Satisfied As Boolean
Dim x1_line As Single, y1_line As Single, x2_line As Single, y2_line, newCircle_x1 As Single, newCircle_y1 As Single,
newCircle_x2 As Single, newCircle_y2 As Single
Dim inter_x As Single, inter_y As Single, t1 As Double, t2 As Double
    'Findingout bigger Circle
If Arc1(3) = Circle1(3) Then
    'Both circle and Arc Radius are same
    Dim Circle1X1 As Single, Circle1Y1 As Single, Circle1X2 As Single, Circle1Y2 As Single
Dim Arc1X1 As Single, Arc1Y1 As Single, Arc1X2 As Single, Arc1Y2 As Single
RADIUSFillet = Circle1(3)
    Dim NewUpX As Single, NewUpY As Single, NewDownX As Single, NewDownY As Single
    Call FindPerpendicularXY(Circle1(1), Circle1(2), Circle1(1), Circle1(2), _
        Arc1(1), Arc1(2), NewUpX, NewUpY, NewDownX, NewDownY)

    Circle1X1 = NewUpX
    Circle1Y1 = NewUpY
    Circle1X2 = NewDownX
    Circle1Y2 = NewDownY

    Call FindPerpendicularXY(Arc1(1), Arc1(2), Arc1(1), Arc1(2), _
        Circle1(1), Circle1(2), NewUpX, NewUpY, NewDownX, NewDownY)
    Arc1X1 = NewUpX
    Arc1Y1 = NewUpY
    Arc1X2 = NewDownX
    Arc1Y2 = NewDownY
    Call FindLineIntersection(Circle1X1, Circle1Y1, Arc1X1, Arc1Y1, Arc1X2, Arc1Y2, Circle1X2, Circle1Y2, inter_x, inter_y, t1,
t2, InterSec)
If InterSec = True Then

```



```

Call CheckOnlyforArc(Arc1X2, Arc1y2, Arc1(1), Arc1(2), Arc1(3), Arc1(4), Arc1(5), Arc1(6), Arc1(7), Arc1Satisfied)
If Arc1Satisfied = True Then
    LineInter.AddItem "line"
    LineInter.AddItem Circle1X1
    LineInter.AddItem Circle1y1
    LineInter.AddItem Arc1X2
    LineInter.AddItem Arc1y2
End If
Call CheckOnlyforArc(Arc1X1, Arc1y1, Arc1(1), Arc1(2), Arc1(3), Arc1(4), Arc1(5), Arc1(6), Arc1(7), Arc1Satisfied)
If Arc1Satisfied = True Then
    LineInter.AddItem "line"
    LineInter.AddItem Arc1X1
    LineInter.AddItem Arc1y1
    LineInter.AddItem Circle1X2
    LineInter.AddItem Circle1y2
End If
Else
    Call CheckOnlyforArc(Arc1X1, Arc1y1, Arc1(1), Arc1(2), Arc1(3), Arc1(4), Arc1(5), Arc1(6), Arc1(7), Arc1Satisfied)
    If Arc1Satisfied = True Then
        LineInter.AddItem "line"
        LineInter.AddItem Circle1X1
        LineInter.AddItem Circle1y1
        LineInter.AddItem Arc1X1
        LineInter.AddItem Arc1y1
    End If
    Call CheckOnlyforArc(Arc1X2, Arc1y2, Arc1(1), Arc1(2), Arc1(3), Arc1(4), Arc1(5), Arc1(6), Arc1(7), Arc1Satisfied)
    If Arc1Satisfied = True Then

        LineInter.AddItem "line"
        LineInter.AddItem Circle1X2
        LineInter.AddItem Circle1y2
        LineInter.AddItem Arc1X2
        LineInter.AddItem Arc1y2
    End If
End If
ElseIf Arc1(3) > Circle1(3) Then
    'Circle1 is bigger than Circle2
    Call ImaginaryCircleforCircleArc(MidxNewCircle, MidyNewCircle, RNewCircle)
    Call ArcCircleInter(Arc1(1), Arc1(2), (Arc1(3) - Circle1(3)), MidxNewCircle, MidyNewCircle, RNewCircle, int_X1, int_Y1,
int_x2, int_Y2)
    'Finding out line and circle intersection point through the smaller inner circle
    Call LineCircleInterSectionforTangent(Arc1(1), Arc1(2), int_X1, int_Y1, Arc1(1), Arc1(2), Arc1(3), Innint_X1, Innint_Y1,
Innint_x2, Innint_Y2)
    Firstintersection_x1 = Innint_X1
    Firstintersection_y1 = Innint_Y1
    Firstintersection_x2 = Innint_x2
    Firstintersection_y2 = Innint_Y2
    Call LineCircleInterSectionforTangent(Arc1(1), Arc1(2), int_x2, int_Y2, Arc1(1), Arc1(2), Arc1(3), Innint_X1, Innint_Y1,
Innint_x2, Innint_Y2)
    Secondintersection_x1 = Innint_X1
    Secondintersection_y1 = Innint_Y1
    Secondintersection_x2 = Innint_x2
    Secondintersection_y2 = Innint_Y2
ElseIf Arc1(3) < Circle1(3) Then
    Call ImaginaryCircleforCircleArc(MidxNewCircle, MidyNewCircle, RNewCircle)
    Call ArcCircleInter(Circle1(1), Circle1(2), Circle1(3) - Arc1(3), MidxNewCircle, MidyNewCircle, RNewCircle, int_X1,
int_Y1, int_x2, int_Y2)
    'Finding out line and circle intersection point through the smaller inner circle
    Call LineCircleInterSectionforTangent(Circle1(1), Circle1(2), int_X1, int_Y1, Circle1(1), Circle1(2), Circle1(3), Innint_X1,
Innint_Y1, Innint_x2, Innint_Y2)
    Firstintersection_x1 = Innint_X1
    Firstintersection_y1 = Innint_Y1
    Firstintersection_x2 = Innint_x2
    Firstintersection_y2 = Innint_Y2
    Call LineCircleInterSectionforTangent(Circle1(1), Circle1(2), int_x2, int_Y2, Circle1(1), Circle1(2), Circle1(3), Innint_X1,
Innint_Y1, Innint_x2, Innint_Y2)
    Secondintersection_x1 = Innint_X1
    Secondintersection_y1 = Innint_Y1
    Secondintersection_x2 = Innint_x2
    Secondintersection_y2 = Innint_Y2
End If
'Dim x1_line As Single, y1_line As Single, x2_line As Single, y2_line As Single
If Arc1(3) > Circle1(3) Then
    Delta_X = int_X1 - Arc1(1)
    Delta_Y = int_Y1 - Arc1(2)

```

```

DeltaXin1 = Firstintersection_x1 - Arc1(1)
DeltaYin1 = Firstintersection_y1 - Arc1(2)
DeltaXin2 = Firstintersection_x2 - Arc1(1)
DeltaYin2 = Firstintersection_y2 - Arc1(2)
Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
Dot_2 = Delta_X * DeltaXin2 + Delta_Y * DeltaYin2
If Dot_1 >= 0 And Dot_2 < 0 Then
    'target point
    x1_line = Firstintersection_x1
    y1_line = Firstintersection_y1
Elseif Dot_1 < 0 And Dot_2 >= 0 Then
    'target point
    x1_line = Firstintersection_x2
    y1_line = Firstintersection_y2
End If
Delta_X = int_x2 - Arc1(1)
Delta_Y = int_Y2 - Arc1(2)
DeltaXin1 = Secondintersection_x1 - Arc1(1)
DeltaYin1 = Secondintersection_y1 - Arc1(2)
DeltaXin2 = Secondintersection_x2 - Arc1(1)
DeltaYin2 = Secondintersection_y2 - Arc1(2)
Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
Dot_2 = Delta_X * DeltaXin2 + Delta_Y * DeltaYin2
If Dot_1 >= 0 And Dot_2 < 0 Then
    'target point
    x2_line = Secondintersection_x1
    y2_line = Secondintersection_y1
Elseif Dot_1 < 0 And Dot_2 >= 0 Then
    'target point
    x2_line = Secondintersection_x2
    y2_line = Secondintersection_y2
End If
m1 = (y1_line - Arc1(2)) / (x1_line - Arc1(1)) 'center to intersection of large circle slope for first point
m2 = (y2_line - Arc1(2)) / (x2_line - Arc1(1)) 'center to intersection of large circle slope for second point
tan_1_m1 = -1 / m1 'slope for tangent at first point
tan_2_m2 = -1 / m2 'slope for tangent at second point
c_m1 = m1 'slope for center to intersection point in small circle for first line
c_m2 = m2 'slope for center to intersection point in small circle for second line
c_tan_1 = y1_line - tan_1_m1 * x1_line 'c of first tangent (y = mx+c)
c_tan_2 = y2_line - tan_2_m2 * x2_line 'c of second tangent (y = mx+c)
c_c1 = Circle1(2) - c_m1 * Circle1(1) 'c of small circle first (y = mx+c)
c_c2 = Circle1(2) - c_m2 * Circle1(1) 'c of small circle second (y = mx+c)
newCircle_x1 = -(c_tan_1 - c_c1) / (tan_1_m1 - c_m1) 'point on small circle x (first point)
newCircle_y1 = c_m1 * newCircle_x1 + c_c1 'point on small circle y (first point)

newCircle_x2 = -(c_tan_2 - c_c2) / (tan_2_m2 - c_m2) 'second point x
newCircle_y2 = c_m2 * newCircle_x2 + c_c2 'second point y

'Checking X1_line, y1_line is this two point between start point of the Arc1 and NewCircleX1, NewCircleY1 is this two point
between start point of Arc2
Call CheckingArc1Arc2(x1_line, y1_line, x2_line, y2_line, Arc1Satisfied, NewCircle1Satisfied, newCircle_x1, newCircle_y1,
newCircle_x2, newCircle_y2, Arc2Satisfied, NewCircle2Satisfied)
'draw line from (x_1_line, y_1_line) to (newCircle_x1, newCircle_y1)
'draw line from (x_2_line, y_2_line) to (newCircle_x2, newCircle_y2)
If Arc1Satisfied = True Then
    LineInter.AddItem "line"
    LineInter.AddItem x1_line
    LineInter.AddItem y1_line
    LineInter.AddItem newCircle_x1
    LineInter.AddItem newCircle_y1
End If
If Arc2Satisfied = True Then
    LineInter.AddItem "line"
    LineInter.AddItem x2_line
    LineInter.AddItem y2_line
    LineInter.AddItem newCircle_x2
    LineInter.AddItem newCircle_y2
End If
End If

If Arc1(3) < Circle1(3) Then
    Delta_X = int_X1 - Circle1(1)
    Delta_Y = int_Y1 - Circle1(2)
    DeltaXin1 = Firstintersection_x1 - Circle1(1)
    DeltaYin1 = Firstintersection_y1 - Circle1(2)

```

```

DeltaXin2 = Firstintersection_x2 - Circle1(1)
DeltaYin2 = Firstintersection_y2 - Circle1(2)
Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
Dot_2 = Delta_X * Deltaxin2 + Delta_Y * DeltaYin2

If Dot_1 >= 0 And Dot_2 < 0 Then
    'target point
    x1_line = Firstintersection_x1 'Firstintersection_x1
    y1_line = Firstintersection_y1 'Firstintersection_y1
Elseif Dot_1 < 0 And Dot_2 >= 0 Then
    'target point
    x1_line = Firstintersection_x2
    y1_line = Firstintersection_y2

End If
Delta_X = int_x2 - Circle1(1)
Delta_Y = int_Y2 - (2)
DeltaXin1 = Secondintersection_x1 - Circle1(1)
DeltaYin1 = Secondintersection_y1 - Circle1(2)
DeltaXin2 = Secondintersection_x2 - Circle1(1)
DeltaYin2 = Secondintersection_y2 - Circle1(2)
Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
Dot_2 = Delta_X * Deltaxin2 + Delta_Y * DeltaYin2

If Dot_1 >= 0 And Dot_2 < 0 Then
    'target point
    'secondintersection_x1
    'secondintersection_y1
    x2_line = Secondintersection_x1
    y2_line = Secondintersection_y1
Elseif Dot_1 < 0 And Dot_2 >= 0 Then
    'target point
    'secondintersection_x2
    'secondintersection_y2
    x2_line = Secondintersection_x2
    y2_line = Secondintersection_y2
End If
m1 = (y1_line - Circle1(2)) / (x1_line - Circle1(1)) 'center to intersection of large circle slope for first point
m2 = (y2_line - Circle1(2)) / (x2_line - Circle1(1)) 'center to intersection of large circle slope for second point
tan_1_m1 = -1 / m1 'slope for tangent at first point
tan_2_m2 = -1 / m2 'slope for tangent at second point
c_m1 = m1 'slope for center to intersection point in small circle for first line
c_m2 = m2 'slope for center to intersection point in small circle for second line
c_tan_1 = y1_line - tan_1_m1 * x1_line 'c of first tangent (y = mx+c)
c_tan_2 = y2_line - tan_2_m2 * x2_line 'c of second tangent (y = mx+c)
c_c1 = Arc1(2) - c_m1 * Arc1(1) 'c of small circle first (y = mx+c)
c_c2 = Arc1(2) - c_m2 * Arc1(1) 'c of small circle second (y = mx+c)
newCircle_x1 = -(c_tan_1 - c_c1) / (tan_1_m1 - c_m1) 'point on small circle x (first point)
newCircle_y1 = c_m1 * newCircle_x1 + c_c1 'point on small circle y (first point)

newCircle_x2 = -(c_tan_2 - c_c2) / (tan_2_m2 - c_m2) 'second point x
newCircle_y2 = c_m2 * newCircle_x2 + c_c2 'second point y

Call CheckingArc1Arc2(x1_line, y1_line, x2_line, y2_line, Arc1Satisfied, NewCircle1Satisfied, newCircle_x1, newCircle_y1,
newCircle_x2, newCircle_y2, Arc2Satisfied, NewCircle2Satisfied)
'draw line from (x_1_line, y_1_line) to (newCircle_x1, newCircle_y1)
'draw line from (x_2_line, y_2_line) to (newCircle_x2, newCircle_y2)

If Arc1Satisfied = True Then
    LineInter.AddItem "line"
    LineInter.AddItem x1_line
    LineInter.AddItem y1_line
    LineInter.AddItem newCircle_x1
    LineInter.AddItem newCircle_y1
End If
If Arc2Satisfied = True Then
    LineInter.AddItem "line"
    LineInter.AddItem x2_line
    LineInter.AddItem y2_line
    LineInter.AddItem newCircle_x2
    LineInter.AddItem newCircle_y2
End If
End If

```

```

Call DeleteSeleReloadArrayDrawobject
Call Objectlist
End Function
Function CheckingOnlyForArc1Arc2(ByVal x1_line As Single, ByVal y1_line As Single, ByVal x2_line As Single, ByVal
y2_line As Single, Arc1Satisfied As Boolean, NewCircle1Satisfied As Boolean, _
ByVal newCircle_x1 As Single, ByVal newCircle_y1 As Single, ByVal newCircle_x2 As Single, ByVal
newCircle_y2 As Single, Arc2Satisfied As Boolean, NewCircle2Satisfied As Boolean)

Dim StartX As Single, StartY As Single, EndX As Single, EndY As Single, Startanglearc1 As Single, Endanglearc1 As Single,
X As Single, Y As Single, StartAngleArc2 As Single
Dim EndAngleArc2 As Single, Returnvalue As Boolean
Arc1Satisfied = False
NewCircle1Satisfied = False
Arc2Satisfied = False
NewCircle2Satisfied = False
StartX = Arc1(4) - Arc1(1)
StartY = Arc1(5) - Arc1(2)
Startanglearc1 = Val(FormatNumber(atan2(StartX, StartY), 3))

EndX = Arc1(6) - Arc1(1)
EndY = Arc1(7) - Arc1(2)
Endanglearc1 = Val(FormatNumber(atan2(EndX, EndY), 3))

X = x1_line - Arc1(1)
Y = y1_line - Arc1(2)
line1angle = Val(FormatNumber(atan2(X, Y), 3))

'Call checkintersectionPointbetweenStartangleAndEndAngle(Startanglearc1, Endanglearc1, line1angle, Returnvalue)
Call FindIntersectionSatisfiedByAnotherArcForTangent(Startanglearc1, line1angle, Endanglearc1, Returnvalue)
Arc1Satisfied = Returnvalue
'x = x2_line - arc1(1)
'y = ....
'angle calculate
'call findintersection = true
X = x2_line - Arc1(1)
Y = y2_line - Arc1(2)
newcircleAngle2 = Val(FormatNumber(atan2(X, Y), 3))
'Call checkintersectionPointbetweenStartangleAndEndAngle(StartAngleArc2, EndAngleArc2, newcircleAngle2, Returnvalue)
Call FindIntersectionSatisfiedByAnotherArcForTangent(Startanglearc1, newcircleAngle2, Endanglearc1, Returnvalue)
NewCircle1Satisfied = Returnvalue

StartX = Arc2(4) - Arc2(1)
StartY = Arc2(5) - Arc2(2)
StartAngleArc2 = Val(FormatNumber(atan2(StartX, StartY), 3))

EndX = Arc2(6) - Arc2(1)
EndY = Arc2(7) - Arc2(2)
EndAngleArc2 = Val(FormatNumber(atan2(EndX, EndY), 3))

X = newCircle_x1 - Arc2(1)
Y = newCircle_y1 - Arc2(2)
line1angle = Val(FormatNumber(atan2(X, Y), 3))
Call FindIntersectionSatisfiedByAnotherArcForTangent(StartAngleArc2, line1angle, EndAngleArc2, Returnvalue)
Arc2Satisfied = Returnvalue

X = newCircle_x2 - Arc2(1)
Y = newCircle_y2 - Arc2(2)
line1angle = Val(FormatNumber(atan2(X, Y), 3))
Call FindIntersectionSatisfiedByAnotherArcForTangent(StartAngleArc2, line1angle, EndAngleArc2, Returnvalue)
NewCircle2Satisfied = Returnvalue
End Function
Function TangentForArcArc()
Dim MidxNewCircle As Single, MidyNewCircle As Single, RNewCircle As Single, Firstintersection_x1 As Single,
Firstintersection_y1 As Single
Dim Firstintersection_x2, Firstintersection_y2
Dim Secondintersection_x1 As Single, Secondintersection_y1 As Single
Dim Secondintersection_x2 As Single, Secondintersection_y2 As Single
Dim int_X1 As Single, int_Y1 As Single, int_x2 As Single, int_Y2 As Single
Dim Innint_X1 As Single, Innint_Y1 As Single, Innint_x2 As Single, Innint_Y2 As Single
Dim Delta_X As Single, Delta_Y As Single, DeltaXin1 As Single, DeltaYin1 As Single, DeltaXin2 As Single, DeltaYin2 As
Single
Dim Dot_1 As Single, Dot_2 As Single
Dim Arc1Satisfied As Boolean, NewCircle1Satisfied As Boolean, Arc2Satisfied As Boolean, NewCircle2Satisfied As Boolean
Dim x1_line As Single, y1_line As Single, x2_line As Single, y2_line, newCircle_x1 As Single, newCircle_y1 As Single,
newCircle_x2 As Single, newCircle_y2 As Single

```

```

Dim inter_x As Single, inter_y As Single, t1 As Double, t2 As Double
'Findingout bigger Circle
If Arc1(3) = Arc2(3) Then
    'Both circles are same
    Dim Circle1X1 As Single, Circle1y1 As Single, Circle1X2 As Single, Circle1y2 As Single
    Dim Circle2X1 As Single, Circle2y1 As Single, Circle2X2 As Single, Circle2y2 As Single
    RadiusFillet = Arc1(3)
    Dim NewUpX As Single, NewUpY As Single, NewDownX As Single, NewDownY As Single
    Call FindPerpendicularXY(Arc1(1), Arc1(2), Arc1(1), Arc1(2), _
        Arc2(1), Arc2(2), NewUpX, NewUpY, NewDownX, NewDownY)

    Circle1X1 = NewUpX
    Circle1y1 = NewUpY
    Circle1X2 = NewDownX
    Circle1y2 = NewDownY

    Call FindPerpendicularXY(Arc2(1), Arc2(2), Arc2(1), Arc2(2), _
        Arc1(1), Arc1(2), NewUpX, NewUpY, NewDownX, NewDownY)
    Circle2X1 = NewUpX
    Circle2y1 = NewUpY
    Circle2X2 = NewDownX
    Circle2y2 = NewDownY
    Call FindLineIntersection(Circle1X1, Circle1y1, Circle2X1, Circle2y1, Circle1X2, Circle1y2, Circle2X2, Circle2y2, inter_x,
        inter_y, t1, t2, InterSec)
    If InterSec = True Then
        Call CheckingOnlyForArc1Arc2(Circle1X1, Circle1y1, Circle1X2, Circle1y2, Arc1Satisfied, NewCircle1Satisfied, _
            Circle2X1, Circle2y1, Circle2X2, Circle2y2, Arc2Satisfied, NewCircle2Satisfied)

        If Arc1Satisfied = True And NewCircle1Satisfied = True And Arc2Satisfied = True And NewCircle2Satisfied = True Then
            LineInter.AddItem "line"
            LineInter.AddItem Circle1X1
            LineInter.AddItem Circle1y1
            LineInter.AddItem Circle2X2
            LineInter.AddItem Circle2y2

            LineInter.AddItem "line"
            LineInter.AddItem Circle1X2
            LineInter.AddItem Circle1y2
            LineInter.AddItem Circle2X1
            LineInter.AddItem Circle2y1
        ElseIf Arc1Satisfied = True Then
            If Arc2Satisfied = True And NewCircle2Satisfied = False Then
                LineInter.AddItem "line"
                LineInter.AddItem Circle1X1
                LineInter.AddItem Circle1y1
                LineInter.AddItem Circle2X1
                LineInter.AddItem Circle2y1

                ElseIf Arc2Satisfied = False And NewCircle2Satisfied = True Then
                    LineInter.AddItem "line"
                    LineInter.AddItem Circle1X1
                    LineInter.AddItem Circle1y1
                    LineInter.AddItem Circle2X2
                    LineInter.AddItem Circle2y2
            End If
        ElseIf NewCircle1Satisfied = True Then
            If Arc2Satisfied = True And NewCircle2Satisfied = False Then
                LineInter.AddItem "line"
                LineInter.AddItem Circle1X2
                LineInter.AddItem Circle1y2
                LineInter.AddItem Circle2X1
                LineInter.AddItem Circle2y1

                ElseIf Arc2Satisfied = False And NewCircle2Satisfied = True Then
                    LineInter.AddItem "line"
                    LineInter.AddItem Circle1X2
                    LineInter.AddItem Circle1y2
                    LineInter.AddItem Circle2X2
                    LineInter.AddItem Circle2y2
            End If
        End If
    End If
End If
End If

```

```

ElseIf Arc1(3) > Arc2(3) Then
    'Circle1 is bigger than Circle2
    Call ImaginaryCircleForArcArc(MidxNewCircle, MidyNewCircle, RNewCircle)
    Call ArcCircleInter(Arc1(1), Arc1(2), (Arc1(3) - Arc2(3)), MidxNewCircle, MidyNewCircle, RNewCircle, int_X1, int_Y1,
    int_x2, int_Y2)
    'Finding out line and circle intersection point through the smaller inner circle
    Call LineCircleInterSectionforTangent(Arc1(1), Arc1(2), int_X1, int_Y1, Arc1(1), Arc1(2), Arc1(3), Innint_X1, Innint_Y1,
    Innint_x2, Innint_Y2)
    Firstintersection_x1 = Innint_X1
    Firstintersection_y1 = Innint_Y1
    Firstintersection_x2 = Innint_x2
    Firstintersection_y2 = Innint_Y2
    Call LineCircleInterSectionforTangent(Arc1(1), Arc1(2), int_x2, int_Y2, Arc1(1), Arc1(2), Arc1(3), Innint_X1, Innint_Y1,
    Innint_x2, Innint_Y2)
    Secondintersection_x1 = Innint_X1
    Secondintersection_y1 = Innint_Y1
    Secondintersection_x2 = Innint_x2
    Secondintersection_y2 = Innint_Y2
ElseIf Arc1(3) < Arc2(3) Then
    Call ImaginaryCircleForArcArc(MidxNewCircle, MidyNewCircle, RNewCircle)
    Call ArcCircleInter(Arc2(1), Arc2(2), Arc2(3) - Arc1(3), MidxNewCircle, MidyNewCircle, RNewCircle, int_X1, int_Y1,
    int_x2, int_Y2)
    'Finding out line and circle intersection point through the smaller inner circle
    Call LineCircleInterSectionforTangent(Arc2(1), Arc2(2), int_X1, int_Y1, Arc2(1), Arc2(2), Arc2(3), Innint_X1, Innint_Y1,
    Innint_x2, Innint_Y2)
    Firstintersection_x1 = Innint_X1
    Firstintersection_y1 = Innint_Y1
    Firstintersection_x2 = Innint_x2
    Firstintersection_y2 = Innint_Y2
    Call LineCircleInterSectionforTangent(Arc2(1), Arc2(2), int_x2, int_Y2, Arc2(1), Arc2(2), Arc2(3), Innint_X1, Innint_Y1,
    Innint_x2, Innint_Y2)
    Secondintersection_x1 = Innint_X1
    Secondintersection_y1 = Innint_Y1
    Secondintersection_x2 = Innint_x2
    Secondintersection_y2 = Innint_Y2
End If
'Dim x1_line As Single, y1_line As Single, x2_line As Single, y2_line As Single
If Arc1(3) > Arc2(3) Then
    Delta_X = int_X1 - Arc1(1)
    Delta_Y = int_Y1 - Arc1(2)
    DeltaXin1 = Firstintersection_x1 - Arc1(1)
    DeltaYin1 = Firstintersection_y1 - Arc1(2)
    DeltaXin2 = Firstintersection_x2 - Arc1(1)
    DeltaYin2 = Firstintersection_y2 - Arc1(2)
    Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
    Dot_2 = Delta_X * DeltaXin2 + Delta_Y * DeltaYin2
    If Dot_1 >= 0 And Dot_2 < 0 Then
        'target point
        x1_line = Firstintersection_x1
        y1_line = Firstintersection_y1
    ElseIf Dot_1 < 0 And Dot_2 >= 0 Then
        'target point
        x1_line = Firstintersection_x2
        y1_line = Firstintersection_y2
    End If
    Delta_X = int_x2 - Arc1(1)
    Delta_Y = int_Y2 - Arc1(2)
    DeltaXin1 = Secondintersection_x1 - Arc1(1)
    DeltaYin1 = Secondintersection_y1 - Arc1(2)
    DeltaXin2 = Secondintersection_x2 - Arc1(1)
    DeltaYin2 = Secondintersection_y2 - Arc1(2)
    Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
    Dot_2 = Delta_X * DeltaXin2 + Delta_Y * DeltaYin2
    If Dot_1 >= 0 And Dot_2 < 0 Then
        'target point
        x2_line = Secondintersection_x1
        y2_line = Secondintersection_y1
    ElseIf Dot_1 < 0 And Dot_2 >= 0 Then
        'target point
        x2_line = Secondintersection_x2

```

```

    y2_line = Secondintersection_y2
End If
m1 = (y1_line - Arc1(2)) / (x1_line - Arc1(1)) 'center to intersection of large circle slope for first point
m2 = (y2_line - Arc1(2)) / (x2_line - Arc1(1)) 'center to intersection of large circle slope for second point
tan_1_m1 = -1 / m1 'slope for tangent at first point
tan_2_m2 = -1 / m2 'slope for tangent at second point
c_m1 = m1 'slope for center to intersection point in small circle for first line
c_m2 = m2 'slope for center to intersection point in small circle for second line
c_tan_1 = y1_line - tan_1_m1 * x1_line 'c of first tangent (y = mx+c)
c_tan_2 = y2_line - tan_2_m2 * x2_line 'c of second tangent (y = mx+c)
c_c1 = Arc2(2) - c_m1 * Arc2(1) 'c of small circle first (y = mx+c)
c_c2 = Arc2(2) - c_m2 * Arc2(1) 'c of small circle second (y = mx+c)
newCircle_x1 = -(c_tan_1 - c_c1) / (tan_1_m1 - c_m1) 'point on small circle x (first point)
newCircle_y1 = c_m1 * newCircle_x1 + c_c1 'point on small circle y (first point)

newCircle_x2 = -(c_tan_2 - c_c2) / (tan_2_m2 - c_m2) 'second point x
newCircle_y2 = c_m2 * newCircle_x2 + c_c2 'second point y

'Checking X1_line, y1_line is this two point between start point of the Arc1 and NewCircle1, NewCircle1 is this two point
between start point of Arc2
Call CheckingArc1Arc2(x1_line, y1_line, x2_line, y2_line, Arc1Satisfied, NewCircle1Satisfied, newCircle_x1, newCircle_y1,
newCircle_x2, newCircle_y2, Arc2Satisfied, NewCircle2Satisfied)
'draw line from (x_1_line, y_1_line) to (newCircle_x1, newCircle_y1)
'draw line from (x_2_line, y_2_line) to (newCircle_x2, newCircle_y2)
If Arc1Satisfied = True And NewCircle1Satisfied = True Then
    LineInter.AddItem "line"
    LineInter.AddItem x1_line
    LineInter.AddItem y1_line
    LineInter.AddItem newCircle_x1
    LineInter.AddItem newCircle_y1
End If
If Arc2Satisfied = True And NewCircle2Satisfied = True Then
    LineInter.AddItem "line"
    LineInter.AddItem x2_line
    LineInter.AddItem y2_line
    LineInter.AddItem newCircle_x2
    LineInter.AddItem newCircle_y2
End If
End If

If Arc1(3) < Arc2(3) Then
    Delta_X = int_X1 - Arc2(1)
    Delta_Y = int_Y1 - Arc2(2)
    DeltaXin1 = Firstintersection_x1 - Arc2(1)
    DeltaYin1 = Firstintersection_y1 - Arc2(2)
    DeltaXin2 = Firstintersection_x2 - Arc2(1)
    DeltaYin2 = Firstintersection_y2 - Arc2(2)
    Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
    Dot_2 = Delta_X * DeltaXin2 + Delta_Y * DeltaYin2

    If Dot_1 >= 0 And Dot_2 < 0 Then
        'target point
        x1_line = Firstintersection_x1 'Firstintersection_x1
        y1_line = Firstintersection_y1 'Firstintersection_y1
    ElseIf Dot_1 < 0 And Dot_2 >= 0 Then
        'target point
        x1_line = Firstintersection_x2
        y1_line = Firstintersection_y2
    End If

    Delta_X = int_x2 - Arc2(1)
    Delta_Y = int_Y2 - Arc2(2)
    DeltaXin1 = Secondintersection_x1 - Arc2(1)
    DeltaYin1 = Secondintersection_y1 - Arc2(2)
    DeltaXin2 = Secondintersection_x2 - Arc2(1)
    DeltaYin2 = Secondintersection_y2 - Arc2(2)
    Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
    Dot_2 = Delta_X * DeltaXin2 + Delta_Y * DeltaYin2

    If Dot_1 >= 0 And Dot_2 < 0 Then
        'target point
        'secondintersection_x1
        'secondintersection_y1
        x2_line = Secondintersection_x1
        y2_line = Secondintersection_y1
    End If

```

```

Elseif Dot_1 < 0 And Dot_2 >= 0 Then
    'target point
    'secondintersection_x2
    'secondintersection_y2
    x2_line = Secondintersection_x2
    y2_line = Secondintersection_y2
End If
m1 = (y1_line - Arc2(2)) / (x1_line - Arc2(1)) 'center to intersection of large circle slope for first point
m2 = (y2_line - Arc2(2)) / (x2_line - Arc2(1)) 'center to intersection of large circle slope for second point
tan_1_m1 = -1 / m1 'slope for tangent at first point
tan_2_m2 = -1 / m2 'slope for tangent at second point
c_m1 = m1 'slope for center to intersection point in small circle for first line
c_m2 = m2 'slope for center to intersection point in small circle for second line
c_tan_1 = y1_line - tan_1_m1 * x1_line 'c of first tangent (y = mx+c)
c_tan_2 = y2_line - tan_2_m2 * x2_line 'c of second tangent (y = mx+c)
c_c1 = Arc1(2) - c_m1 * Arc1(1) 'c of small circle first (y = mx+c)
c_c2 = Arc1(2) - c_m2 * Arc1(1) 'c of small circle second (y = mx+c)
newCircle_x1 = -(c_tan_1 - c_c1) / (tan_1_m1 - c_m1) 'point on small circle x (first point)
newCircle_y1 = c_m1 * newCircle_x1 + c_c1 'point on small circle y (first point)

newCircle_x2 = -(c_tan_2 - c_c2) / (tan_2_m2 - c_m2) 'second point x
newCircle_y2 = c_m2 * newCircle_x2 + c_c2 'second point y

Call CheckingArc1Arc2(x1_line, y1_line, x2_line, y2_line, Arc1Satisfied, NewCircle1Satisfied, newCircle_x1, newCircle_y1,
newCircle_x2, newCircle_y2, Arc2Satisfied, NewCircle2Satisfied)
'draw line from (x_1_line, y_1_line) to (newCircle_x1, newCircle_y1)
'draw line from (x_2_line, y_2_line) to (newCircle_x2, newCircle_y2)

If Arc1Satisfied = True And NewCircle1Satisfied = True Then
    LineInter.AddItem "line"
    LineInter.AddItem x1_line
    LineInter.AddItem y1_line
    LineInter.AddItem newCircle_x1
    LineInter.AddItem newCircle_y1
End If
If Arc2Satisfied = True And NewCircle2Satisfied = True Then
    LineInter.AddItem "line"
    LineInter.AddItem x2_line
    LineInter.AddItem y2_line
    LineInter.AddItem newCircle_x2
    LineInter.AddItem newCircle_y2
End If
End If

Call DeleteSeleReloadArrayDrawobject
Call Objectlist
End Function
Function checkintersectionPointbetweenStartangleAndEndAngle(ByVal Startanglearc1 As Single, ByVal Endanglearc1 As
Single, ByVal line1angle As Single, Returnvalue As Boolean)
Returnvalue = False
If Endanglearc1 < Startanglearc1 Then
    StartAngle = 0
    Endanglearc1 = 360 - Endanglearc1 + Startanglearc1
    line1angle = 360 - Startanglearc1 + line1angle
    If line1angle >= 360 Then
        line1angle = line1angle - 360
    End If
    If line1angle >= StartAngle And line1angle <= Endanglearc1 Then
        Returnvalue = True
    End If
Elseif line1angle >= Startanglearc1 And line1angle <= Endanglearc1 Then
    Returnvalue = True
End If
End Function

Function CheckingArc1Arc2(ByVal x1_line As Single, ByVal y1_line As Single, ByVal x2_line As Single, ByVal y2_line As
Single, Arc1Satisfied As Boolean, NewCircle1Satisfied As Boolean, _
ByVal newCircle_x1 As Single, ByVal newCircle_y1 As Single, ByVal newCircle_x2 As Single, ByVal
newCircle_y2 As Single, Arc2Satisfied As Boolean, NewCircle2Satisfied As Boolean)

Dim StartX As Single, StartY As Single, EndX As Single, EndY As Single, Startanglearc1 As Single, Endanglearc1 As Single,
X As Single, Y As Single, StartAngleArc2 As Single
Dim EndAngleArc2 As Single, Returnvalue As Boolean
Arc1Satisfied = False

```



```

NewCircle1Satisfied = False
Arc2Satisfied = False
NewCircle2Satisfied = False
  StartX = Arc1(4) - Arc1(1)
  StartY = Arc1(5) - Arc1(2)
  StartAngleArc1 = Val(FormatNumber(atan2(StartX, StartY), 3))

  EndX = Arc1(6) - Arc1(1)
  EndY = Arc1(7) - Arc1(2)
  EndAngleArc1 = Val(FormatNumber(atan2(EndX, EndY), 3))

  X = x1_line - Arc1(1)
  Y = y1_line - Arc1(2)
  line1angle = Val(FormatNumber(atan2(X, Y), 3))

  'Call checkIntersectionPointbetweenStartangleAndEndAngle(StartangleArc1, EndangleArc1, line1angle, Returnvalue)
  Call FindIntersectionSatisfiedByAnotherArcForTangent(StartangleArc1, line1angle, EndangleArc1, Returnvalue)
  Arc1Satisfied = Returnvalue
  'x = x2_line - arc1(1)
  'y = ...
  'angle calculate
  'call findintersection = true

  StartX = Arc2(4) - Arc2(1)
  StartY = Arc2(5) - Arc2(2)
  StartAngleArc2 = Val(FormatNumber(atan2(StartX, StartY), 3))

  EndX = Arc2(6) - Arc2(1)
  EndY = Arc2(7) - Arc2(2)
  EndAngleArc2 = Val(FormatNumber(atan2(EndX, EndY), 3))

  X = x1_line - Arc1(1)
  Y = y1_line - Arc1(2)
  newcircleAngle1 = Val(FormatNumber(atan2(X, Y), 3))
  'Call checkIntersectionPointbetweenStartangleAndEndAngle(StartAngleArc2, EndAngleArc2, newcircleAngle1, Returnvalue)
  Call FindIntersectionSatisfiedByAnotherArcForTangent(StartAngleArc2, newcircleAngle1, EndAngleArc2, Returnvalue)
  NewCircle1Satisfied = Returnvalue

  StartX = Arc1(4) - Arc1(1)
  StartY = Arc1(5) - Arc1(2)
  StartAngleArc1 = Val(FormatNumber(atan2(StartX, StartY), 3))

  EndX = Arc1(6) - Arc1(1)
  EndY = Arc1(7) - Arc1(2)
  EndangleArc1 = Val(FormatNumber(atan2(EndX, EndY), 3))
  X = x2_line - Arc1(1)
  Y = y2_line - Arc1(2)
  line2angle = Val(FormatNumber(atan2(X, Y), 3))
  'Call checkIntersectionPointbetweenStartangleAndEndAngle(StartangleArc1, EndangleArc1, line2angle, Returnvalue)
  Call FindIntersectionSatisfiedByAnotherArcForTangent(StartangleArc1, line2angle, EndangleArc1, Returnvalue)
  Arc2Satisfied = Returnvalue

  StartX = Arc2(4) - Arc2(1)
  StartY = Arc2(5) - Arc2(2)
  StartAngleArc2 = Val(FormatNumber(atan2(StartX, StartY), 3))

  EndX = Arc2(6) - Arc2(1)
  EndY = Arc2(7) - Arc2(2)
  EndAngleArc2 = Val(FormatNumber(atan2(EndX, EndY), 3))

  X = x2_line - Arc1(1)
  Y = y2_line - Arc1(2)
  newcircleAngle2 = Val(FormatNumber(atan2(X, Y), 3))
  'Call checkIntersectionPointbetweenStartangleAndEndAngle(StartAngleArc2, EndAngleArc2, newcircleAngle2, Returnvalue)
  Call FindIntersectionSatisfiedByAnotherArcForTangent(StartAngleArc2, newcircleAngle2, EndAngleArc2, Returnvalue)
  NewCircle2Satisfied = Returnvalue

End Function
Function TangentForCircleCircle()
Dim MidxNewCircle As Single, MidyNewCircle As Single, RNewCircle As Single, Firstintersection_x1 As Single,
Firstintersection_y1 As Single
Dim Firstintersection_x2, Firstintersection_y2

```

```
Dim Secondintersection_x1 As Single, Secondintersection_y1 As Single
Dim Secondintersection_x2 As Single, Secondintersection_y2 As Single
Dim int_X1 As Single, int_Y1 As Single, int_x2 As Single, int_Y2 As Single
Dim Innint_X1 As Single, Innint_Y1 As Single, Innint_x2 As Single, Innint_Y2 As Single
Dim Delta_X As Single, Delta_Y As Single, DeltaXin1 As Single, DeltaYin1 As Single, DeltaXin2 As Single, DeltaYin2 As Single
Dim Dot_1 As Single, Dot_2 As Single

Dim inter_x As Single, inter_y As Single, InterSec As Boolean, t1 As Double, t2 As Double

'Findingout bigger Circle
If Circle1(3) = Circle2(3) Then
    'Both circles are same
    Dim Circle1X1 As Single, Circle1y1 As Single, Circle1X2 As Single, Circle1y2 As Single
    Dim Circle2X1 As Single, Circle2y1 As Single, Circle2X2 As Single, Circle2y2 As Single
    RadiusFillet = Circle1(3)
    Dim NewUpX As Single, NewUpY As Single, NewDownX As Single, NewDownY As Single
    Call FindPerpendicularXY(Circle1(1), Circle1(2), Circle1(1), Circle1(2), _
        Circle2(1), Circle2(2), NewUpX, NewUpY, NewDownX, NewDownY)

    Circle1X1 = NewUpX
    Circle1y1 = NewUpY
    Circle1X2 = NewDownX
    Circle1y2 = NewDownY

    Call FindPerpendicularXY(Circle2(1), Circle2(2), Circle2(1), Circle2(2), _
        Circle1(1), Circle1(2), NewUpX, NewUpY, NewDownX, NewDownY)
    Circle2X1 = NewUpX
    Circle2y1 = NewUpY
    Circle2X2 = NewDownX
    Circle2y2 = NewDownY
    Call FindLineIntersection(Circle1X1, Circle1y1, Circle2X1, Circle2y1, Circle1X2, Circle1y2, Circle2X2, Circle2y2, inter_x,
        inter_y, t1, t2, InterSec)
    If InterSec = True Then
        LineInter.AddItem "line"
        LineInter.AddItem Circle1X1
        LineInter.AddItem Circle1y1
        LineInter.AddItem Circle2X2
        LineInter.AddItem Circle2y2

        LineInter.AddItem "line"
        LineInter.AddItem Circle1X2
        LineInter.AddItem Circle1y2
        LineInter.AddItem Circle2X1
        LineInter.AddItem Circle2y1
    Else
        LineInter.AddItem "line"
        LineInter.AddItem Circle1X1
        LineInter.AddItem Circle1y1
        LineInter.AddItem Circle2X2
        LineInter.AddItem Circle2y2

        LineInter.AddItem "line"
        LineInter.AddItem Circle1X2
        LineInter.AddItem Circle1y2
        LineInter.AddItem Circle2X1
        LineInter.AddItem Circle2y1
    End If
    Elseif Circle1(3) > Circle2(3) Then
        'Circle1 is bigger than Circle2
        Call ImaginaryCircle(MidxNewCircle, MidyNewCircle, RNewCircle)
        Call ArcCircleInter(Circle1(1), Circle1(2), (Circle1(3) - Circle2(3)), MidxNewCircle, MidyNewCircle, RNewCircle, int_X1,
            int_Y1, int_x2, int_Y2)
        'Finding out line and circle intersection point through the smaller inner circle
        Call LineCircleInterSectionforTangent(Circle1(1), Circle1(2), int_X1, int_Y1, Circle1(1), Circle1(2), Circle1(3), Innint_X1,
            Innint_Y1, Innint_x2, Innint_Y2)
        Firstintersection_x1 = Innint_X1
        Firstintersection_y1 = Innint_Y1
        Firstintersection_x2 = Innint_x2
        Firstintersection_y2 = Innint_Y2
```

```

    Call LineCircleInterSectionforTangent(Circle1(1), Circle1(2), int_x2, int_Y2, Circle1(1), Circle1(2), Circle1(3), Innint_X1,
Innint_Y1, Innint_x2, Innint_Y2)
    Secondintersection_x1 = Innint_X1
    Secondintersection_y1 = Innint_Y1
    Secondintersection_x2 = Innint_x2
    Secondintersection_y2 = Innint_Y2
ElseIf Circle1(3) < Circle2(3) Then
    Call ImaginaryCircle(MidxNewCircle, MidyNewCircle, RNewCircle)
    Call ArcCircleInter(Circle2(1), Circle2(2), Circle2(3) - Circle1(3), MidxNewCircle, MidyNewCircle, RNewCircle, int_X1,
int_Y1, int_x2, int_Y2)
    'Finding out line and circle intersection point through the smaller inner circle
    Call LineCircleInterSectionforTangent(Circle2(1), Circle2(2), int_X1, int_Y1, Circle2(1), Circle2(2), Circle2(3), Innint_X1,
Innint_Y1, Innint_x2, Innint_Y2)
    Firstintersection_x1 = Innint_X1
    Firstintersection_y1 = Innint_Y1
    Firstintersection_x2 = Innint_x2
    Firstintersection_y2 = Innint_Y2
    Call LineCircleInterSectionforTangent(Circle2(1), Circle2(2), int_x2, int_Y2, Circle2(1), Circle2(2), Circle2(3), Innint_X1,
Innint_Y1, Innint_x2, Innint_Y2)
    Secondintersection_x1 = Innint_X1
    Secondintersection_y1 = Innint_Y1
    Secondintersection_x2 = Innint_x2
    Secondintersection_y2 = Innint_Y2
End If
Dim x1_line As Single, y1_line As Single, x2_line As Single, y2_line As Single
If Circle1(3) > Circle2(3) Then
    Delta_X = int_X1 - Circle1(1)
    Delta_Y = int_Y1 - Circle1(2)
    DeltaXin1 = Firstintersection_x1 - Circle1(1)
    DeltaYin1 = Firstintersection_y1 - Circle1(2)
    DeltaXin2 = Firstintersection_x2 - Circle1(1)
    DeltaYin2 = Firstintersection_y2 - Circle1(2)
    Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
    Dot_2 = Delta_X * DeltaXin2 + Delta_Y * DeltaYin2
    If Dot_1 >= 0 And Dot_2 < 0 Then
        'target point
        x1_line = Firstintersection_x1
        y1_line = Firstintersection_y1
    ElseIf Dot_1 < 0 And Dot_2 >= 0 Then
        'target point
        x1_line = Firstintersection_x2
        y1_line = Firstintersection_y2
    End If
    Delta_X = int_x2 - Circle1(1)
    Delta_Y = int_Y2 - Circle1(2)
    DeltaXin1 = Secondintersection_x1 - Circle1(1)
    DeltaYin1 = Secondintersection_y1 - Circle1(2)
    DeltaXin2 = Secondintersection_x2 - Circle1(1)
    DeltaYin2 = Secondintersection_y2 - Circle1(2)
    Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
    Dot_2 = Delta_X * DeltaXin2 + Delta_Y * DeltaYin2
    If Dot_1 >= 0 And Dot_2 < 0 Then
        'target point
        x2_line = Secondintersection_x1
        y2_line = Secondintersection_y1
    ElseIf Dot_1 < 0 And Dot_2 >= 0 Then
        'target point
        x2_line = Secondintersection_x2
        y2_line = Secondintersection_y2
    End If
    m1 = (y1_line - Circle1(2)) / (x1_line - Circle1(1)) ' center to intersection of large circle slope for first point
    m2 = (y2_line - Circle1(2)) / (x2_line - Circle1(1)) ' center to intersection of large circle slope for second point
    tan_1_m1 = -1 / m1 ' slope for tangent at first point
    tan_2_m2 = -1 / m2 ' slope for tangent at second point
    c_m1 = m1 ' slope for center to intersection point in small circle for first line
    c_m2 = m2 ' slope for center to intersection point in small circle for second line
    c_tan_1 = y1_line - tan_1_m1 * x1_line ' c of first tangent (y = mx+c)
    c_tan_2 = y2_line - tan_2_m2 * x2_line ' c of second tangent (y = mx+c)
    c_c1 = Circle2(2) - c_m1 * Circle2(1) ' c of small circle first (y = mx+c)
    c_c2 = Circle2(2) - c_m2 * Circle2(1) ' c of small circle second (y = mx+c)
    newCircle_x1 = -(c_tan_1 - c_c1) / (tan_1_m1 - c_m1) ' point on small circle x (first point)
    newCircle_y1 = c_m1 * newCircle_x1 + c_c1 ' point on small circle y (first point)

    newCircle_x2 = -(c_tan_2 - c_c2) / (tan_2_m2 - c_m2) ' second point x
    newCircle_y2 = c_m2 * newCircle_x2 + c_c2 ' second point y

```

```
' draw line from (x_1_line, y_1_line) to (newCircle_x1, newCircle_y1)
' draw line from (x_2_line, y_2_line) to (newCircle_x2, newCircle_y2)
LineInter.AddItem "line"
LineInter.AddItem x1_line
LineInter.AddItem y1_line
LineInter.AddItem newCircle_x1
LineInter.AddItem newCircle_y1

LineInter.AddItem "line"
LineInter.AddItem x2_line
LineInter.AddItem y2_line
LineInter.AddItem newCircle_x2
LineInter.AddItem newCircle_y2
End If

If Circle1(3) < Circle2(3) Then
    Delta_X = int_X1 - Circle2(1)
    Delta_Y = int_Y1 - Circle2(2)
    DeltaXin1 = Firstintersection_x1 - Circle2(1)
    DeltaYin1 = Firstintersection_y1 - Circle2(2)
    DeltaXin2 = Firstintersection_x2 - Circle2(1)
    DeltaYin2 = Firstintersection_y2 - Circle2(2)
    Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
    Dot_2 = Delta_X * DeltaXin2 + Delta_Y * DeltaYin2

    If Dot_1 >= 0 And Dot_2 < 0 Then
        'target point
        x1_line = Firstintersection_x1 'Firstintersection_x1
        y1_line = Firstintersection_y1 'Firstintersection_y1
    ElseIf Dot_1 < 0 And Dot_2 >= 0 Then
        'target point
        x1_line = Firstintersection_x2
        y1_line = Firstintersection_y2

    End If
    Delta_X = int_x2 - Circle2(1)
    Delta_Y = int_Y2 - Circle2(2)
    DeltaXin1 = Secondintersection_x1 - Circle2(1)
    DeltaYin1 = Secondintersection_y1 - Circle2(2)
    DeltaXin2 = Secondintersection_x2 - Circle2(1)
    DeltaYin2 = Secondintersection_y2 - Circle2(2)
    Dot_1 = Delta_X * DeltaXin1 + Delta_Y * DeltaYin1
    Dot_2 = Delta_X * DeltaXin2 + Delta_Y * DeltaYin2

    If Dot_1 >= 0 And Dot_2 < 0 Then
        'target point
        'secondintersection_x1
        'secondintersection_y1
        x2_line = Secondintersection_x1
        y2_line = Secondintersection_y1
    ElseIf Dot_1 < 0 And Dot_2 >= 0 Then
        'target point
        'secondintersection_x2
        'secondintersection_y2
        x2_line = Secondintersection_x2
        y2_line = Secondintersection_y2
    End If
    m1 = (y1_line - Circle2(2)) / (x1_line - Circle2(1)) ' center to intersection of large circle slope for first point
    m2 = (y2_line - Circle2(2)) / (x2_line - Circle2(1)) 'center to intersection of large circle slope for second point
    tan_1_m1 = -1 / m1 ' slope for tangent at first point
    tan_2_m2 = -1 / m2 ' slope for tangent at second point
    c_m1 = m1 ' slope for center to intersection point in small circle for first line
    c_m2 = m2 ' slope for center to intersection point in small circle for second line
    c_tan_1 = y1_line - tan_1_m1 * x1_line ' c of first tangent (y = mx+c)
    c_tan_2 = y2_line - tan_2_m2 * x2_line ' c of second tangent (y = mx+c)
    c_c1 = Circle1(2) - c_m1 * Circle1(1) ' c of small circle first (y = mx+c)
    c_c2 = Circle1(2) - c_m2 * Circle1(1) ' c of small circle second (y = mx+c)
    newCircle_x1 = -(c_tan_1 - c_c1) / (tan_1_m1 - c_m1) ' point on small circle x (first point)
    newCircle_y1 = c_m1 * newCircle_x1 + c_c1 ' point on small circle y (first point)

    newCircle_x2 = -(c_tan_2 - c_c2) / (tan_2_m2 - c_m2) ' second point x
    newCircle_y2 = c_m2 * newCircle_x2 + c_c2 ' second point y
    ' draw line from (x_1_line, y_1_line) to (newCircle_x1, newCircle_y1)
    ' draw line from (x_2_line, y_2_line) to (newCircle_x2, newCircle_y2)
    LineInter.AddItem "line"
```

```

LineInter.AddItem x1_line
LineInter.AddItem y1_line
LineInter.AddItem newCircle_x1
LineInter.AddItem newCircle_y1

LineInter.AddItem "line"
LineInter.AddItem x2_line
LineInter.AddItem y2_line
LineInter.AddItem newCircle_x2
LineInter.AddItem newCircle_y2
End If

Call DeleteSeleReloadArrayDrawobject
Call Objectlist
End Function
Function ImaginaryCircleforCircleArc(ByRef MidxNewCircle As Single, ByRef MidyNewCircle As Single, ByRef RNewCircle
As Single)

MidxNewCircle = (Circle1(1) + Arc1(1)) / 2
MidyNewCircle = (Circle1(2) + Arc1(2)) / 2
RNewCircle = Sqr((MidxNewCircle - Circle1(1)) ^ 2 + (MidyNewCircle - Circle1(2)) ^ 2)
End Function
Function ImaginaryCircle(ByRef MidxNewCircle As Single, ByRef MidyNewCircle As Single, ByRef RNewCircle As Single)

MidxNewCircle = (Circle1(1) + Circle2(1)) / 2
MidyNewCircle = (Circle1(2) + Circle2(2)) / 2
RNewCircle = Sqr((MidxNewCircle - Circle2(1)) ^ 2 + (MidyNewCircle - Circle2(2)) ^ 2)
End Function
Function ImaginaryCircleForArcArc(ByRef MidxNewCircle As Single, ByRef MidyNewCircle As Single, ByRef RNewCircle
As Single)

MidxNewCircle = (Arc1(1) + Arc2(1)) / 2
MidyNewCircle = (Arc1(2) + Arc2(2)) / 2
RNewCircle = Sqr((MidxNewCircle - Arc2(1)) ^ 2 + (MidyNewCircle - Arc2(2)) ^ 2)
End Function
Function LineCircleInterSectionforTangent(X1 As Single, Y1 As Single, X2 As Single, Y2 As Single, CX As Single, CY As
Single, _
    R As Single, int_X1 As Single, int_Y1 As Single, int_x2 As Single, int_Y2 As Single)

Dim dx As Single, dy As Single, Compare As Single
Dim a As Single, b As Single, C As Single
Dim Tang As Single
Dim tempFx As Single, tempSx As Single, tempFy As Single, tempSy As Single
Dim D As Single

a = FormatNumber((X2 - X1) ^ 2 + (Y2 - Y1) ^ 2, 5)
b = FormatNumber(2 * ((X2 - X1) * (X1 - CX) + (Y2 - Y1) * (Y1 - CY)), 5)
C = FormatNumber(CX ^ 2 + CY ^ 2 + X1 ^ 2 + Y1 ^ 2 - 2 * (CX * X1 + CY * Y1) - R ^ 2, 5)
Comp = b ^ 2 - 4 * a * C
oneintersec = False
twointersec = False
If Comp < 0 Then
    Exit Function
Dim Distance As Single, Distance_1 As Single

D = FormatNumber(Abs(((X2 - CX) * (Y1 - CY)) - ((X1 - CX) * (Y2 - CY))) / Sqr((X2 - X1) ^ 2 + (Y2 - Y1) ^ 2))

Compare = FormatNumber(Abs(R - D), 4)

If (Compare < 0.4) Then
    Tang = -b / (2 * a)
    Tang = (-b + Sqr(b ^ 2 - 4 * a * C)) / (2 * a)
    int_X1 = FormatNumber(X1 + Tang * (X2 - X1), 3)
    int_Y1 = FormatNumber(Y1 + Tang * (Y2 - Y1), 3)

    Tang = (-b - Sqr(b ^ 2 - 4 * a * C)) / (2 * a)
    int_x2 = X1 + (Tang * (X2 - X1))
    int_Y2 = Y1 + (Tang * (Y2 - Y1))
    Distance = FormatNumber(((int_X1 - CX) ^ 2 + (int_Y1 - CY) ^ 2) ^ (1 / 2))
    Distance_1 = Abs(Distance - R)
    If Distance_1 > 1 Then
        Tang = (-b + Sqr(b ^ 2 - 4 * a * C)) / (2 * a)

```

```

    int_X1 = FormatNumber(X1 + Tang * (X2 - X1), 3)
    int_Y1 = FormatNumber(Y1 + Tang * (Y2 - Y1), 3)
End If
ElseIf Comp > 0 Then
    Tang = (-b + Sqr(b ^ 2 - 4 * a * C)) / (2 * a)

    int_X1 = X1 + Tang * (X2 - X1)
    int_Y1 = Y1 + Tang * (Y2 - Y1)

    Tang = (-b - Sqr(b ^ 2 - 4 * a * C)) / (2 * a)
    int_x2 = X1 + (Tang * (X2 - X1))
    int_Y2 = Y1 + (Tang * (Y2 - Y1))
End If
End Function

Public Function FindSameArcLineForFillet(ByVal ArcX1 As Single, ByVal ArcY1 As Single, _
    ByVal ArcX2 As Single, ByVal ArcY2 As Single, _
    ByRef LineX1 As Single, ByRef LineY1 As Single, _
    ByRef LineX2 As Single, ByRef LineY2 As Single, FirstPoint As Boolean, SecondPoint As Boolean)

    'Dim SecondPoint As Boolean
    If ((LineX1 = ArcX1 And LineY1 = ArcY1) Or (LineX1 = ArcX2 And LineY1 = ArcY2)) Then
        FirstPoint = True
    End If
    If ((LineX2 = ArcX1 And LineY2 = ArcY1) Or (LineX2 = ArcX2 And LineY2 = ArcY2)) Then
        SecondPoint = True
    End If

End Function

Function IntersectionpointbetweenStartorEndAngleArc(ByVal LAngle1 As Single, ByVal CompareAngle As Single, ByVal
    LAngle4 As Single, Returnvalue As Boolean)
    Dim Asume As Single, EndAngle As Single
    Returnvalue = False
    If LAngle4 = 0 Then
        Asume = 360
    Else
        Asume = LAngle4
    End If
    If LAngle1 > LAngle4 Then
        StartAngle = 0
        EndAngle = 360 - LAngle1 + LAngle4
        CompareAngle = 360 - LAngle1 + CompareAngle
        If CompareAngle >= 360 Then
            CompareAngle = CompareAngle - 360
        End If
        LAngle3 = 360 - LAngle1 + LAngle3
        'If LAngle3 >= 360 Then
        '    LAngle3 = LAngle3 - 360
        'End If
        If CompareAngle >= StartAngle And CompareAngle <= EndAngle Then
            Returnvalue = True
        End If
    Else
        If CompareAngle >= LAngle1 And CompareAngle <= Asume Then
            Returnvalue = True
        End If
    End If
End Function

Function Objectlist()

'End Function
Dim objectnumber As Integer
objectnumber = 0
List1.Clear
For i = 0 To LineInter.ListCount
    If LineInter.List(i) = "line" Then
        List1.AddItem "line" & " " & objectnumber
        List1.AddItem LineInter.List(i + 1)
        List1.AddItem LineInter.List(i + 2)
        List1.AddItem LineInter.List(i + 3)
        List1.AddItem LineInter.List(i + 4)
        objectnumber = objectnumber + 1
    End If
Next i
For i = 0 To LineInter.ListCount

```

```
If LineInter.List(i) = "circle" Then
    List1.AddItem "circle" & " " & objectnumber
    List1.AddItem LineInter.List(i + 1)
    List1.AddItem LineInter.List(i + 2)
    List1.AddItem LineInter.List(i + 3)
    objectnumber = objectnumber + 1
End If
Next i
For i = 0 To LineInter.ListCount
    If LineInter.List(i) = "arc" Then
        List1.AddItem "arc" & " " & objectnumber
        List1.AddItem LineInter.List(i + 1)
        List1.AddItem LineInter.List(i + 2)
        List1.AddItem LineInter.List(i + 3)
        List1.AddItem LineInter.List(i + 4)
        List1.AddItem LineInter.List(i + 5)
        List1.AddItem LineInter.List(i + 6)
        List1.AddItem LineInter.List(i + 7)
        objectnumber = objectnumber + 1
    End If
Next i
End Function
```