Journal of Information Engineering and Applications
ISSN 2224-5782 (print) ISSN 2225-0506 (online)
Vol.13, No.2, 2023

Visual Basic Drawing Codes from 2D AutoCAD Drawings and Machine Parts Applications

Telat TÜRKYILMAZ
Vocational School, Department of Mechanics and Metal Technologies, Bilecik Şeyh Edebali University, 1100, Bilecik, Turkey

E-mail address: telat.turkyilmaz@bilecik.edu.tr

Abstract
It is of great importance to support vocational programs with graphics. Before the necessary graphic codes are written, drawings are usually prepared in the CAD environment. Then, graphic coordinates are taken from the drawing environment and software graphic codes are prepared. This situation requires a lot of effort and time.

In this study, a software has been developed in AutoLISP programming language, which converts drawings in Autocad environment into Visual Basic drawing codes in a very short time. The use of mechanical engineering drawing codes produced with the developed program is shown in visual basic environment with animation support.

The conversion of bolts, radiused parts, gear and propeller drawings in Autocad environment into Visual Basic codes and program examples are shown in the software developed in this study.

It is expected that the developed program will save time and effort in writing the graphic codes to be used in new program designs.

Keywords: AutoLISP, Visual Basic, Autocad, Simulation, Software Development

DOI: 10.7176/JIEA/13-2-04
Publication date: March 31st 2023

Introduction

Software development requires great effort, expertise and software knowledge. Thanks to the use of these developed software, previously spent effort and costs are recovered in a short time. In addition, their future use provides benefits and advantages for the purpose of the software. AutoLISP programming language runs in Autocad environment. In addition to its own functions, it can use Autocad commands and capabilities. Support can be obtained for the graphics and calculations of the software to be developed from this program.

When the academic studies are examined, it is seen that the AutoLISP programming and Visual Basic language is widely used.

Türkyılmaz (1997), in his master's thesis, calculated and drew bending beams in Autocad environment using the AutoLISP programming language.

Singh & Sekhon (1999), In their studies, with the help of the expert system they developed using the AutoLISP programming language, they selected press machines from the database in accordance with the parameters defined in the processing of sheet metal sheets.

Stojkovic & Stankic (2006) developed a software with AutoLISP and Visual Basic programming languages for lightning protection of power transmission lines.

Khalili et al. (2007), obtain 3d model data using 3d scanner for reverse engineering. In their studies, 3D models were obtained from scanner data in AutoCAD environment by using AutoLISP and Visual Basic programming languages.

Ayyildiz et al. (2010), developed a software for parametric gear drawing and modeling using AutoLISP and Visual Basic. The software they developed saves time and effort by using it instead of manual design.

Kashid & Kumar (2014), mentioned that the selection and design for Components of Compound Die is difficult and complex. They have shown that it can be done with less time and effort with the software they developed using AutoLISP, VB and AutoCAD in their studies.

Naranje & Kumar (2014), made automatic design software for deep drawing die. AutoLISP, Visual Basic 6.0 programming languages and AutoCAD environment were used in their work. They preferred artificial intelligence and knowledge-based modeling in the software they developed. In addition, all operations in the software are provided automatically and in a short time.

It is aimed to prepare Visual Basic graphic codes from Autocad drawings through AutoLISP programming language in our study.

Supporting the developed programs with graphics increases the quality of the program. Graphics can be created in the program environment with software codes. Unfortunately, the desired result is not always achieved. It may be a good solution to draw the graphics in a design program such as Autocad before writing the graphic codes. However, time-consuming work is required by the software developer to convert point information into software codes.

With the AutoLISP software "VB_BL.LSP" developed in this study, the defined drawing objects in the Autocad environment are converted into Visual Basic program codes in a very short time.

AutoLISP programs with "LSP" extension work in Autocad environment. It is a powerful programming language that uses Autocad commands in addition to its own graphics functions. The software developed with this programming language can generate Visual Basic software codes in a new file by taking the drawing information from Autocad environment.

Thus, the target software codes can be obtained automatically from the drawings in a short time.

AutoLISP Software Introduction

AutoLISP programming language is open source code run in Autocad environment (Molotnikov & Molotnikova 2023; Ambrosius 2015). Codes written in this language can be encrypted and protected without hindering their operation. In this section, applications about the structure of the AutoLISP software are given (Ambrosius 2015).

Getting Drawing Information from Autocad Environment

E1 Autocad command has been developed to read and understand drawing information. This command is given in Figure 1**.** By running this command, information about the selected drawing object is transferred to the variable E1 then the saved file is opened in notepad. With the Autocad's "appload" command, the AutoLISP program in the e1.lsp file is loaded. Another method for loading is to paste the developed codes into the Command line.


```
(defun c:E1()
(setq DOSYA1 (strcat (getvar "dwgprefix")
(vl-filename-base (getvar "dwgname")) ".txt"))

(setq F1(open (STRCAT DOSYA1) "w"))

(setq da (ssget))
(SETQ SIRA 0)
(REPEAT (SSLENGTH DA)
(setq E1(ENTGET(SSNAME da SIRA)))
```

```
(foreach o E1
(print o)
(print o F1)
);foreach
(PRINC "\n")(PRINC "\n" F1)(SETQ SIRA(+ SIRA 1))
);repeat
(PRINC (STRCAT "\n" DOSYA1))
(TEXTSCR)
(startapp (strcat "explorer " DOSYA1 ))
(princ)
)
```

Figure 1. The E1 command prepares drawing information.

Getting and Using Drawing Information

The E1 command developed in this study transfers the selected any object information from Autocad environment to the E1 variable for later use. Table 1 presents some examples of usage of drawing information.

Table 1. Using AutoLISP Commands

	Command	**Result**	**Explanation**
1	E1 <enter> “line selection” <enter>	Selected line information E1 variable is created	Drawing object selection
2	(cdr(assoc 10 e1))	(80.0 100.0 0.0)	Getting the 10 parameter information
3	(car(cdr(assoc 10 e1)))	80.0	Getting the x1 coordinate of the first point
4	(cadr(cdr(assoc 10 e1)))	100.0	Getting the y1 coordinate of the first point
5	(cdr(assoc 11 e1))	(500.0 400.0 0.0)	Getting the 11 parameter information
6	(car(cdr(assoc 11 e1)))	500.0	Getting the x2 coordinate of the second point
7	(cadr(cdr(assoc 11 e1)))	400.0	Getting the y2 coordinate of the second point

Table 1 rows descriptions:

Line 1: E1 command is written on Autocad command line. Then the line object shown in Figure 3 is selected. Thus, the information of the selected object is transferred to the variable E1. The content of E1 is shown in Figure 1.

Line 2: Type "(cdr(assoc 10 e1))" in Autocad command line and press enter. With this code sequence, row information starting with 10 is taken from E1 variable.

Line 3: The car function gets the first value from the selected row in variable E1

Line 4: The cadr function gets the second value from the selected row in variable E1.

Line 5: With the code in the 5th line, the information at the position starting with 11 is taken from database shown in Figure 4.

Line 6: The car function gets the first value from the selected row in variable E1

Line 7: The cadr function gets the second value from the selected row in variable E1.

The strcat AutoLISP function is used to concatenate text variables.

Preparing VB code with the AUTOLISP programming language	**Result**
(strcat "CIZ.DrawLine(PEN1," (rtos (car(cdr(assoc 10 E1))) 2 0) ", " (rtos (cadr(cdr(assoc 10 E1))) 2 0) ", " (rtos (car(cdr(assoc 11 E1))) 2 0) ", " (rtos (cadr(cdr(assoc 11 E1))) 2 0))	CIZ.DrawLine(PEN1,80, 100, 500, 400)

Figure 2. Obtaining VB code from line drawing using AutoLISP program

Basic drawings and coding processes in Autocad environment.
In this section, information is given about the drawings and generated codes in the developed software.

Receiving LINE Drawing Information From Autocad by AutoLISP

The line object depicted in Figure 3 is available in the Autocad environment.

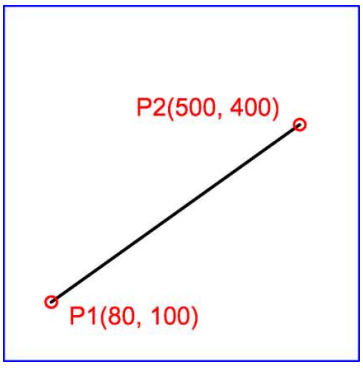

Figure 3. LINE in Autocad environment

The drawing information in Figure 4 is acquired from the LINE object using the newly developed "E1" com-mand.

```
Information from The Line Object
(-1 . <Entity name: 1eb43f49e50>)
(0 . "LINE")
(330 . <Entity name: 1eb43f3a9f0>)
(5 . "1E65")
(100 . "AcDbEntity")
(67 . 0)
(410 . "Model")
(8 . "0")
(100 . "AcDbLine")
(10 80.0 100.0 0.0)
(11 500.0 400.0 0.0)
(210 0.0 0.0 1.0)
```

Figure 4. Information from the LINE object

The Visual Basic codes in are acquired from the LINE object using the newly developed "VB_BL" command as shown in Figure 5.

```
Function LINE1(
Optional PB As PictureBox = Nothing,
Optional ANGLE As Integer = 0,
Optional DX As Integer = 0,
Optional DY As Integer = 0) As Bitmap
  If PB Is Nothing Then PB = PictureBox1


  Dim P1 As New List(Of Point)
  Dim RESIM As New Bitmap(PB.Width, PB.Height)
  Dim CIZ As Graphics = Graphics.FromImage(RESIM)
  CIZ.Clear(Color.White)
  CIZ.TranslateTransform(0 + DX, PB.Height - DY)
  CIZ.RotateTransform(ANGLE)
  CIZ.ScaleTransform(1, -1)
  Dim PEN0 As Pen = New Pen(Color.Black, 2)
  Dim PEN1 As Pen = New Pen(Color.Black, 3)
```



```
Dim PEN2 As Pen = New Pen(Color.Blue, 2)
PEN2.LineJoin = Drawing2D.LineJoin.Bevel
Dim PEN3 As Pen = New Pen(Color.Red, 2.5)
PEN3.DashPattern = {6, 2}
Dim PEN4 As Pen = New Pen(Color.Magenta, 2)
PEN4.DashPattern = {20, 3, 3, 3, 3}
PEN1.Color = Color.Black
CIZ.DrawLine(PEN1, 80, 100, 500, 400)
PB.Image = RESIM
Return RESIM
End Function
```

Figure 5. Visual Basic program produced from LINE

Receiving LWPOLYLINE Drawing Information from Autocad by AutoLISP

The LWPOLYLINE object depicted in Figure 6 is available in the Autocad environment.

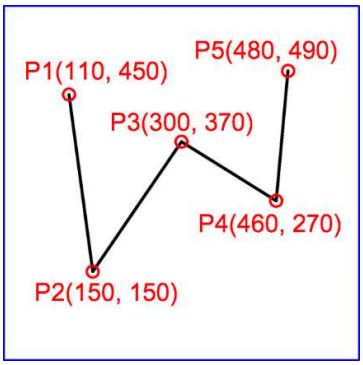

Figure 6. LWPOLYLINE in Autocad environment

The drawing information in Figure 7 is acquired from the LWPOLYLINE object using the newly developed "E1" command.

(-1 . <Entity name: 1ba4ee6e3b0>)	(40 . 0.0)
(0 . "LWPOLYLINE")	(41 . 0.0)
(330 . <Entity name: 1ba565239f0>)	(42 . 0.0)
(5 . "6D3")	(91 . 0)
(100 . "AcDbEntity")	(10 300.0 370.0)
(67 . 0)	(40 . 0.0)
(410 . "Model")	(41 . 0.0)
(8 . "0")	(42 . 0.0)
(100 . "AcDbPolyline")	(91 . 0)
(90 . 5)	(10 460.0 270.0)
(70 . 0)	(40 . 0.0)
(43 . 0.0)	(41 . 0.0)
(38 . 0.0)	(42 . 0.0)
(39 . 0.0)	(91 . 0)
(10 110.0 450.0)	(10 480.0 490.0)
(40 . 0.0)	(40 . 0.0)
(41 . 0.0)	(41 . 0.0)
(42 . 0.0)	(42 . 0.0)
(91 . 0)	(91 . 0)
(10 150.0 150.0)	(210 0.0 0.0 1.0)

Figure 7. Information from the LWPOLYLINE object

The Visual Basic codes in Figure 8 are acquired from the LWPOLYLINE object using the newly developed "VB_BL" command.

```
Function LWPOLYLINE1( _
Optional PB As PictureBox = Nothing, _
Optional ANGLE As Integer = 0, _
Optional DX As Integer = 0, _
Optional DY As Integer = 0) As Bitmap
If PB Is Nothing Then PB = PictureBox1

Dim P1 As New List(Of Point)
Dim RESIM As New Bitmap(PB.Width, PB.Height)
Dim CIZ As Graphics = Graphics.FromImage(RESIM)
CIZ.Clear(Color.White)
CIZ.TranslateTransform(0 + DX, PB.Height - DY)
CIZ.RotateTransform(ANGLE)
CIZ.ScaleTransform(1, -1)
Dim PEN0 As Pen = New Pen(Color.Black, 2)
Dim PEN1 As Pen = New Pen(Color.Black, 3)
Dim PEN2 As Pen = New Pen(Color.Blue, 2)
PEN2.LineJoin = Drawing2D.LineJoin.Bevel
Dim PEN3 As Pen = New Pen(Color.Red, 2.5)
PEN3.DashPattern = {6, 2}
Dim PEN4 As Pen = New Pen(Color.Magenta, 2)
PEN4.DashPattern = {20, 3, 3, 3, 3}

P1 = New List(Of Point)
P1.Add(New Point(110, 450))
P1.Add(New Point(150, 150))
P1.Add(New Point(300, 370))
P1.Add(New Point(460, 270))
P1.Add(New Point(480, 490))
CIZ.DrawLines(PEN1, P1.ToArray)
PB.Image = RESIM
Return RESIM
End Function
```

Figure 8. LWPOLYLINE object visual basic code

Receiving CIRCLE Drawing Information From Autocad by AutoLISP

The CIRCLE object depicted in Figure 9 is available in the Autocad environment.

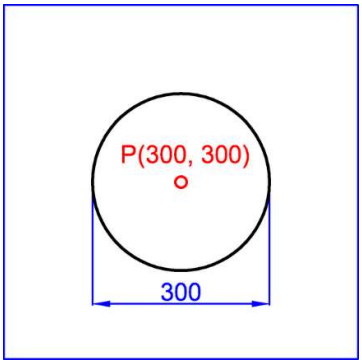

Figure 9. CIRCLE in Autocad environment

The drawing information in Figure 10 is acquired from the CIRCLE object using the newly developed "E1" command.

Information From The Circle Object
(-1 . <Entity name: 254326e6b80>)
(0 . "CIRCLE")
(330 . <Entity name: 254326e99f0>)
(5 . "C8")
(100 . "AcDbEntity")
(67 . 0)
(410 . "Model")
(8 . "0")
(100 . "AcDbCircle")
(10 300.0 300.0 0.0)
(40 . 150.0)
(210 0.0 0.0 1.0)

Figure 10. Information from the CIRCLE object

The Visual Basic codes in Figure 11 are acquired from the `CIRCLE` object using the newly developed “VB_BL” command.


```
Function CIRCLE1( _
Optional PB As PictureBox = Nothing, _
Optional ANGLE As Integer = 0, _
Optional DX As Integer = 0, _
Optional DY As Integer = 0) As Bitmap
If PB Is Nothing Then PB = PictureBox1

Dim P1 As New List(Of Point)
Dim RESIM As New Bitmap(PB.Width, PB.Height)
Dim CIZ As Graphics = Graphics.FromImage(RESIM)
CIZ.Clear(Color.White)
CIZ.TranslateTransform(0 + DX, PB.Height - DY)
CIZ.RotateTransform(ANGLE)
CIZ.ScaleTransform(1, -1)
Dim PEN0 As Pen = New Pen(Color.Black, 2)
Dim PEN1 As Pen = New Pen(Color.Black, 3)
Dim PEN2 As Pen = New Pen(Color.Blue, 2)
PEN2.LineJoin = Drawing2D.LineJoin.Bevel
Dim PEN3 As Pen = New Pen(Color.Red, 2.5)
PEN3.DashPattern = {6, 2}
Dim PEN4 As Pen = New Pen(Color.Magenta, 2)
PEN4.DashPattern = {20, 3, 3, 3, 3}
PEN1.Color = Color.Black
CIZ.DrawEllipse(PEN1, 150, 150, 300, 300)
PB.Image = RESIM
Return RESIM
End Function
```

Figure 11. Visual Basic program produced from CIRCLE

Receiving ARC Drawing Information From Autocad by AutoLISP

The ARC object depicted in Figure 12 is available in the Autocad environment.

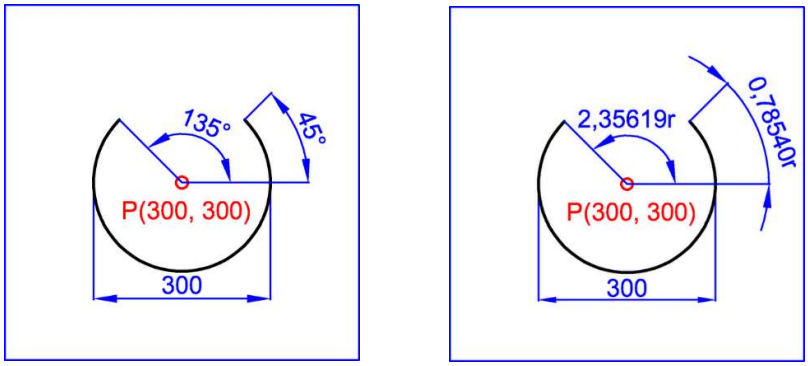

Figure 12. ARC in Autocad environment

The drawing information in Figure 13 is acquired from the ARC object using the newly developed "E1" command.

```
(-1 . <Entity name: 254326e6b80>)
(0 . "ARC")
(330 . <Entity name: 254326e99f0>)
(5 . "C8")
(100 . "AcDbEntity")
(67 . 0)
(410 . "Model")
(8 . "0")
(100 . "AcDbCircle")
(10 300.0 300.0 0.0)
(40 . 150.0)
(210 0.0 0.0 1.0)
(100 . "AcDbArc")
(50 . 2.35619)
(51 . 0.785398)
```

Figure 13. Information from the ARC object

The Visual Basic codes in Figure 14 are acquired from the ARC object using the newly developed “VB_BL” command.


```
Function ARC1( _
Optional PB As PictureBox = Nothing, _
Optional ANGLE As Integer = 0, _
Optional DX As Integer = 0, _
Optional DY As Integer = 0) As Bitmap
If PB Is Nothing Then PB = PictureBox1

Dim P1 As New List(Of Point)
Dim RESIM As New Bitmap(PB.Width, PB.Height)
Dim CIZ As Graphics =
Graphics.FromImage(RESIM)
CIZ.Clear(Color.White)
CIZ.TranslateTransform(0 + DX, PB.Height -
DY)
CIZ.RotateTransform(ANGLE)
CIZ.ScaleTransform(1, -1)
Dim PEN0 As Pen = New Pen(Color.Black, 2)
Dim PEN1 As Pen = New Pen(Color.Black, 3)
Dim PEN2 As Pen = New Pen(Color.Blue, 2)
PEN2.LineJoin = Drawing2D.LineJoin.Bevel
Dim PEN3 As Pen = New Pen(Color.Red, 2.5)
PEN3.DashPattern = {6, 2}
Dim PEN4 As Pen = New Pen(Color.Magenta, 2)
PEN4.DashPattern = {20, 3, 3, 3, 3}
CIZ.DrawArc(PEN1, 150, 150, 300, 300, 135,
270)
PB.Image = RESIM
Return RESIM
End Function
```

Figure 14. Visual Basic program produced from ARC

Receiving ROTATEDDIMENSION Drawing Information from Autocad by AutoLISP

The ROTATEDDIMENSION object depicted in Figure 15 is available in the Autocad environment.

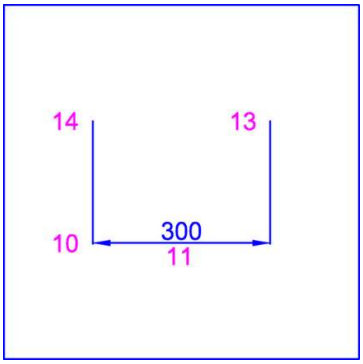

Figure 15. ROTATEDDIMENSION in Autocad environment

The drawing information in Figure 16 is acquired from the ROTATEDDIMENSION object using the newly developed "E1" command.

(-1 . <Entity name: 2172bab1270>)	(42 . 300.0)
(0 . "DIMENSION")	(73 . 0)
(330 . <Entity name: 2172576c1f0>)	(74 . 0)
(5 . "3EF")	(75 . 0)
(100 . "AcDbEntity")	(52 . 0.0)
(67 . 0)	(53 . 0.0)
(410 . "Model")	(54 . 0.0)
(8 . "iNCE")	(51 . 0.0)
(100 . "AcDbDimension")	(210 0.0 0.0 1.0)
(280 . 0)	(3 . "VB")
(2 . "*D2")	(100 . "AcDbAlignedDimension")
(10 150.0 196.84 0.0)	(13 450.0 403.16 0.0)
(11 300.0 217.167 0.0)	(14 150.0 403.16 0.0)
(12 0.0 0.0 0.0)	(15 0.0 0.0 0.0)
(70 . 32)	(16 0.0 0.0 0.0)
(1 . "")	(40 . 0.0)
(71 . 5)	(50 . 0.0)
(72 . 1)	(100 . "AcDbRotatedDimension")
(41 . 1.0)	

Figure 16. Information from the ROTATEDDIMENSION object

The Visual Basic codes in Figure 17 are acquired from the ROTATEDDIMENSION object using the newly developed “VB_BL” command.


```
Function RotatedDimension1( _
Optional PB As PictureBox = Nothing, _
Optional ANGLE As Integer = 0, _
Optional DX As Integer = 0, _
Optional DY As Integer = 0) As Bitmap
If PB Is Nothing Then PB = PictureBox1


Dim P1 As New List(Of Point)
Dim RESIM As New Bitmap(PB.Width,
PB.Height)
Dim CIZ As Graphics =
Graphics.FromImage(RESIM)
CIZ.Clear(Color.White)
CIZ.TranslateTransform(0 + DX,
PB.Height - DY)
CIZ.RotateTransform(ANGLE)
CIZ.ScaleTransform(1, -1)
Dim PEN0 As Pen = New Pen(Color.Black,
2)
Dim PEN1 As Pen = New Pen(Color.Black,
3)
Dim PEN2 As Pen = New Pen(Color.Blue,
2)
PEN2.LineJoin =
Drawing2D.LineJoin.Bevel
Dim PEN3 As Pen = New Pen(Color.Red,
2.5)
PEN3.DashPattern = {6, 2}
Dim PEN4 As Pen = New
Pen(Color.Magenta, 2)
PEN4.DashPattern = {20, 3, 3, 3, 3}
```

```
TEXT_ROTATED_BL(CIZ, PB, "300", 0, 300,
217, "MC", "ARIAL", 30, Brushes.Black,
, ANGLE, DX, DY)
P1 = New List(Of Point)
P1.Add(New Point(150, 403))
P1.Add(New Point(150, 196))
P1.Add(New Point(150, 197))
P1.Add(New Point(450, 197))
P1.Add(New Point(450, 196))
P1.Add(New Point(450, 403))
CIZ.DrawLines(New Pen(Brushes.Blue, 2),
P1.ToArray)
P1 = New List(Of Point)
P1.Add(New Point(150, 197))
P1.Add(New Point(180, 204))
P1.Add(New Point(180, 189))
CIZ.FillPolygon(Brushes.Blue,
P1.ToArray)
P1 = New List(Of Point)
P1.Add(New Point(450, 197))
P1.Add(New Point(420, 204))
P1.Add(New Point(420, 189))
CIZ.FillPolygon(Brushes.Blue,
P1.ToArray)
PB.Image = RESIM
Return RESIM
End Function
```

Figure 17. Visual Basic program produced from ROTATEDDIMENSION

Receiving ALIGNEDDIMENSION Drawing Information from Autocad by AutoLISP

The ALIGNEDDIMENSION object depicted in Figure 18 is available in the Autocad environment.

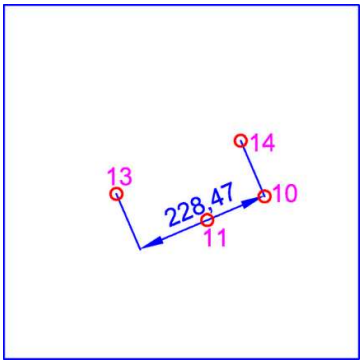

Figure 18. ALIGNEDDIMENSION in Autocad environment

The drawing information in Figure 19 is acquired from the ALIGNEDDIMENSION object using the newly developed "E1" command.

(-1 . <Entity name: 1ba4f46e920>)	(41 . 1.0)
(0 . "DIMENSION")	(42 . 228.473)
(330 . <Entity name: 1ba565239f0>)	(73 . 0)
(5 . "812")	(74 . 0)
(100 . "AcDbEntity")	(75 . 0)
(67 . 0)	(52 . 0.0)
(410 . "Model")	(53 . 0.0)
(8 . "KESİK")	(54 . 0.0)
(100 . "AcDbDimension")	(51 . 0.0)
(280 . 0)	(210 0.0 0.0 1.0)
(2 . "*D5")	(3 . "VBR")
(10 439.904 276.891 0.0)	(100 . "AcDbAlignedDimension")
(11 343.496 235.574 0.0)	(13 190.0 280.0 0.0)
(12 0.0 0.0 0.0)	(14 400.0 370.0 0.0)
(70 . 161)	(15 0.0 0.0 0.0)
(1 . "")	(16 0.0 0.0 0.0)
(71 . 5)	(40 . 0.0)
(72 . 1)	(50 . 0.0)

Figure 19. Information from the ALIGNEDDIMENSION object

The Visual Basic codes in Figure 20 are acquired from the ALIGNEDDIMENSION object using the newly developed “VB_BL” command.


```
Function AlignedDimension1( _
Optional PB As PictureBox = Nothing, _
Optional ANGLE As Integer = 0, _
Optional DX As Integer = 0, _
Optional DY As Integer = 0) As Bitmap
If PB Is Nothing Then PB = PictureBox1


Dim P1 As New List(Of Point)
Dim RESIM As New Bitmap(PB.Width, PB.Height)
Dim CIZ As Graphics =
Graphics.FromImage(RESIM)
CIZ.Clear(Color.White)
CIZ.TranslateTransform(0 + DX, PB.Height -
DY)
CIZ.RotateTransform(ANGLE)
CIZ.ScaleTransform(1, -1)
Dim PEN0 As Pen = New Pen(Color.Black, 2)
Dim PEN1 As Pen = New Pen(Color.Black, 3)
Dim PEN2 As Pen = New Pen(Color.Blue, 2)
PEN2.LineJoin = Drawing2D.LineJoin.Bevel
Dim PEN3 As Pen = New Pen(Color.Red, 2.5)
PEN3.DashPattern = {6, 2}
Dim PEN4 As Pen = New Pen(Color.Magenta, 2)
PEN4.DashPattern = {20, 3, 3, 3, 3}
TEXT_ROTATED_BL(CIZ, PB, "228.47", 23.2,
343, 236, "TC", "ARIAL", 30, Brushes.Black,
, ANGLE, DX, DY)
P1 = New List(Of Point)
P1.Add(New Point(400, 370))
P1.Add(New Point(440, 276))
P1.Add(New Point(440, 277))
P1.Add(New Point(230, 187))
P1.Add(New Point(230, 186))
P1.Add(New Point(190, 280))
CIZ.DrawLines(New
Pen(Brushes.Blue, 2), P1.ToArray)
P1 = New List(Of Point)
P1.Add(New Point(440, 277))
P1.Add(New Point(400, 268))
P1.Add(New Point(406, 254))
CIZ.FillPolygon(Brushes.Blue,
P1.ToArray)
P1 = New List(Of Point)
P1.Add(New Point(230, 187))
P1.Add(New Point(264, 210))
P1.Add(New Point(270, 195))
CIZ.FillPolygon(Brushes.Blue,
P1.ToArray)
P1 = New List(Of Point)
P1.Add(New Point(440, 277))
P1.Add(New Point(307, 220))
CIZ.DrawLines(New
Pen(Brushes.Blue, 2), P1.ToArray)
PB.Image = RESIM
Return RESIM
End Function
```

Figure 20. Visual Basic program produced from ALIGNEDDIMENSION

Receiving RADIALDIMENSION Drawing Information from Autocad by AutoLISP

The RADIALDIMENSION object depicted in Figure 21 is available in the Autocad environment.

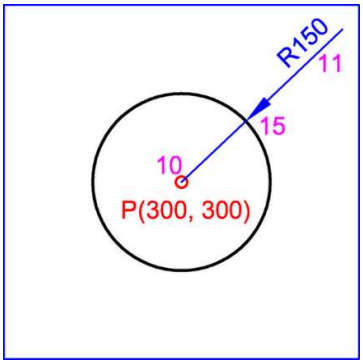

Figure 21. RADIALDIMENSION in Autocad environment

The drawing information in Figure 22 is acquired from the RADIALDIMENSION object using the newly developed "E1" command.

(-1 . <Entity name: 217076f6a10>)	(42 . 150.0)
(0 . "DIMENSION")	(73 . 0)
(330 . <Entity name: 217076eb9f0>)	(74 . 0)
(5 . "6D1")	(75 . 0)
(100 . "AcDbEntity")	(52 . 0.0)
(67 . 0)	(53 . 0.0)
(410 . "Model")	(54 . 0.0)
(8 . "İNCE")	(51 . 0.0)
(100 . "AcDbDimension")	(210 0.0 0.0 1.0)
(280 . 0)	(3 . "VBR")
(2 . "*D4")	(100 . "AcDbRadialDimension")
(10 300.0 300.0 0.0)	(13 0.0 0.0 0.0)
(11 526.39 515.336 0.0)	(14 0.0 0.0 0.0)
(12 0.0 0.0 0.0)	(15 408.686 403.379 0.0)
(70 . 36)	(16 0.0 0.0 0.0)
(1 . "")	(40 . 0.0)
(71 . 5)	(50 . 0.0)
(72 . 1)	
(41 . 1.0)	

Figure 22. Information from the RADIALDIMENSION object

The Visual Basic codes in Figure 23 are acquired from the RADIALDIMENSION object using the newly developed "VB_BL" command.

```
Function RadialDimension1( _
Optional PB As PictureBox = Nothing, _
Optional ANGLE As Integer = 0, _
Optional DX As Integer = 0, _
Optional DY As Integer = 0) As Bitmap
If PB Is Nothing Then PB = PictureBox1

Dim P1 As New List(Of Point)
Dim RESIM As New Bitmap(PB.Width,
PB.Height)
Dim CIZ As Graphics =
Graphics.FromImage(RESIM)
CIZ.Clear(Color.White)
CIZ.TranslateTransform(0 + DX,
PB.Height - DY)
CIZ.RotateTransform(ANGLE)
CIZ.ScaleTransform(1, -1)
Dim PEN0 As Pen = New Pen(Color.Black,
2)
Dim PEN1 As Pen = New Pen(Color.Black,
3)
Dim PEN2 As Pen = New Pen(Color.Blue,
2)
PEN2.LineJoin =
Drawing2D.LineJoin.Bevel
```

```
Dim PEN3 As Pen = New Pen(Color.Red,
2.5)
PEN3.DashPattern = {6, 2}
Dim PEN4 As Pen = New
Pen(Color.Magenta, 2)

PEN4.DashPattern = {20, 3, 3, 3, 3}
TEXT_ROTATED_BL(CIZ, PB, "R150",
403.57, 526, 515, "TC", "ARIAL", 30,
Brushes.Black, , ANGLE, DX, DY)
P1 = New List(Of Point)
P1.Add(New Point(300, 300))
P1.Add(New Point(409, 403))
P1.Add(New Point(563, 550))
CIZ.DrawLines(PEN2, P1.ToArray)
P1 = New List(Of Point)
P1.Add(New Point(409, 403))
P1.Add(New Point(454, 429))
P1.Add(New Point(436, 447))
CIZ.FillPolygon(Brushes.Blue,
P1.ToArray)
PB.Image = RESIM
Return RESIM
End Function
```

Figure 23. Visual Basic program produced from RADIALDIMENSION

Introduction of Software Developed with AutoLISP Programming Language

The "E1" command running in the Autocad environment was developed using the AutoLISP programming language. The information of the drawings provided by Autocad can be displayed with the new "E1" Command. This program supports the preparation of new AutoLISP codes. This command consists of 21 lines. The program content is shown in Figure 1.

The "VB_BL" command running in the Autocad environment was developed using the AutoLISP programming language. The developed program has 767 lines and the file size is 29.3 KB (30.061 bytes). Visual Basic Codes can be generated from Text, Line, Lwpolyline, Circle, Arc, RotatedDimension, AlignedDimension, RadialDimension, AcDb2LineAngularDimension and 2d solid drawings with this command. Multiple selections can be used in this program.

The "TEXT_ROTATED_BL" function was developed in Visual Basic programming environment to write formatted texts in all directions. This Function consists of 46 lines. This program has been developed to support angular texts coming from the graphical environment into the Visual Basic environment.

The bottom left point in the Picturebox object is (X0, Y0) point.

Case Studies with the Developed Software on Machine Parts Applications
In this section, the studies carried out with the developed software are introduced.

PB01_BOLT-VB.dwg

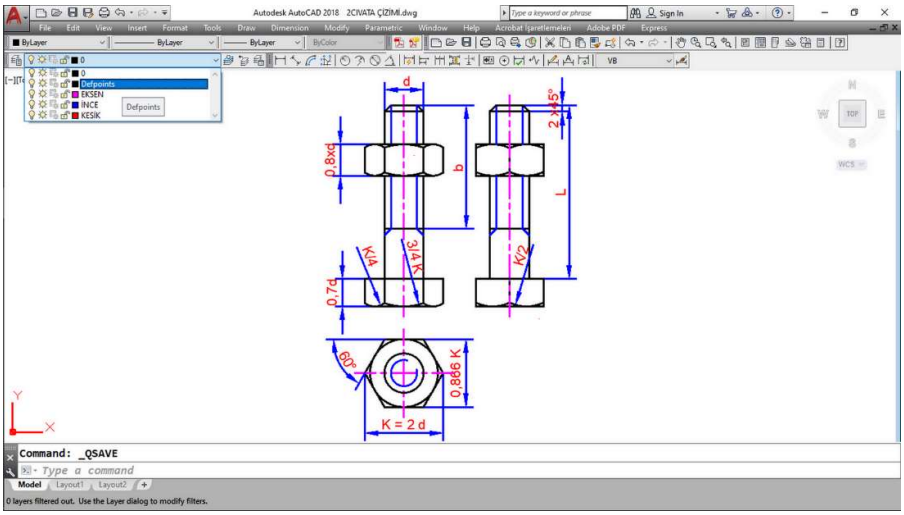

Figure 24. Bolt drawing in AutoCAD environment

Figure 24 shows a bolt drawing in Autocad environment. The size of the file named PB01_BOLT.dwg is 72,6 KB (74.432 bayt) and contains 92 drawing objects.

The developed software "VB_BL" command produced the PB01_BOLT function with 1501 words in 355 lines. Figure 25 demonstrates a drawing created with the Visual Basic function PB01_BOLT.

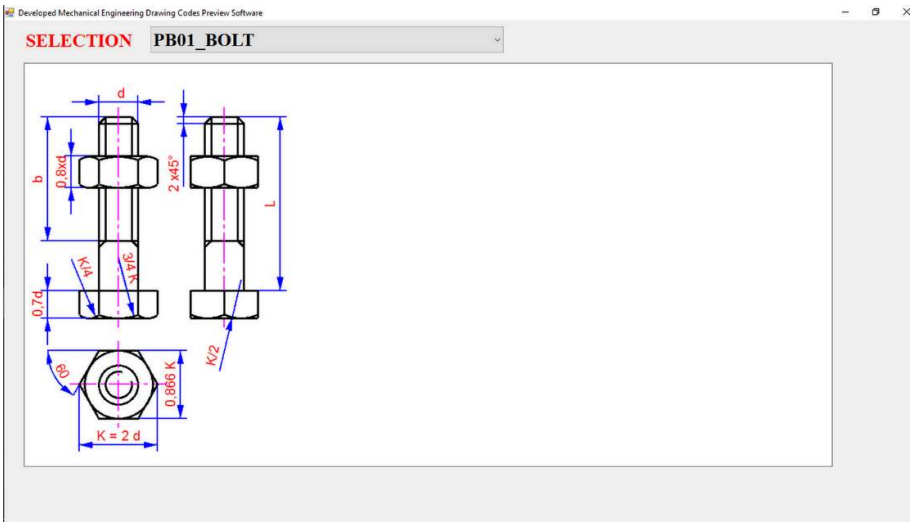

Figure 25. Bolt picture drawn with the code produced by the developed AutoLISP software

PB02_ARC.dwg

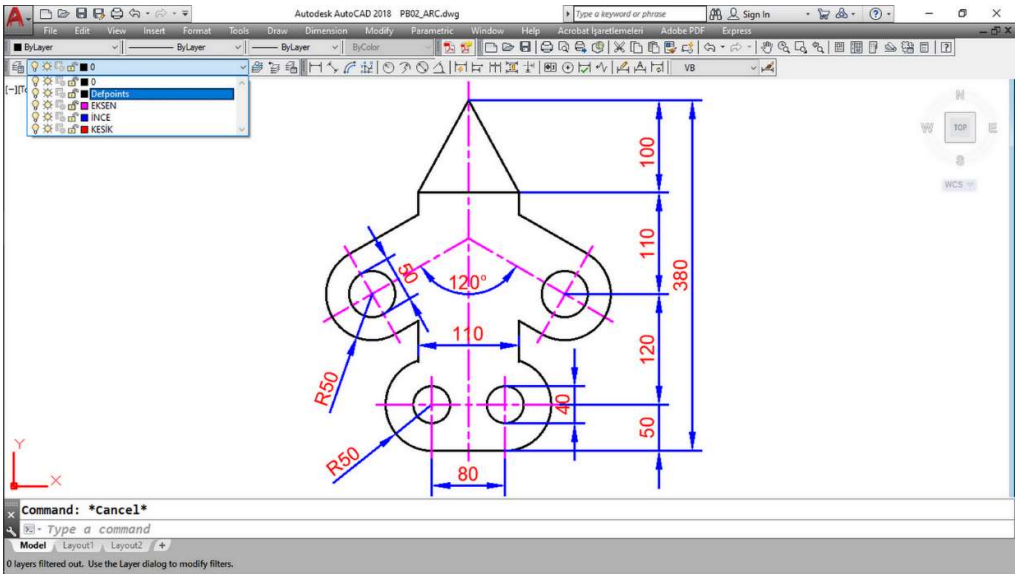

Figure 26. Sample drawing with arcs prepared in Autocad environment

Figure 26 shows PB02_ARC drawing in Autocad environment. The size of the file named PB02_ARC.dwg is 66,0 KB (67.584 bayt). and contains 41 drawing objects.

The developed software “VB_BL” command produced the PB02_ARC function with 1216 words in 298 lines. Figure 27 demonstrates a drawing created with the Visual Basic function PB02_ARC.

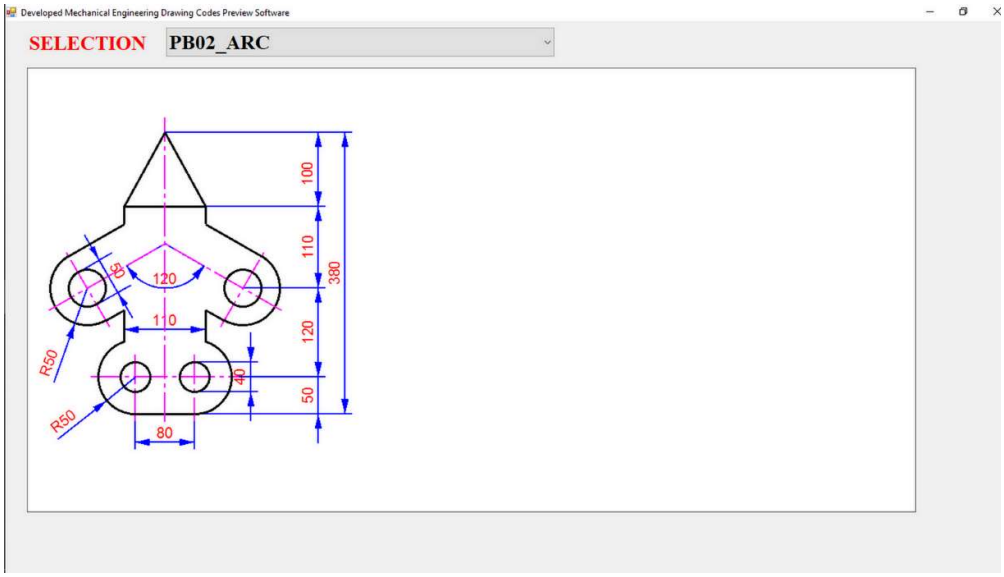

Figure 27. Sample drawing containing arcs drawn with the code produced by the developed AutoLISP software

PB03_SPUR_GEAR.dwg

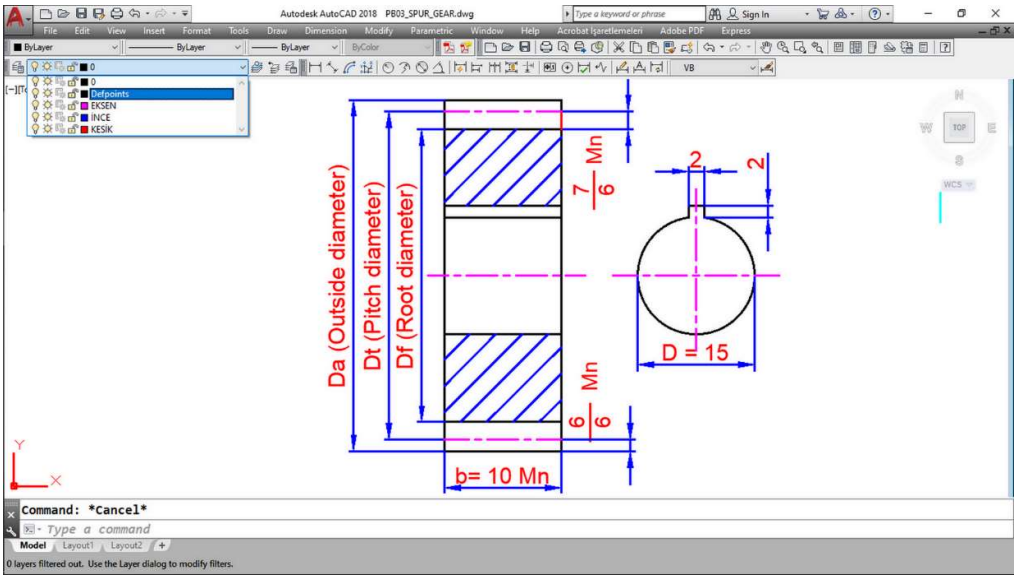

Figure 28. Spur gear drawing in Autocad environment

Figure 28 shows PB03_SPUR_GEAR drawing in Autocad environment. The size of the file named PB03_SPUR_GEAR.dwg is 67,9 KB (69.568 bayt) and contains 52 drawing objects.

The developed software "VB_BL" command produced the PB03_SPUR_GEAR function with 1162 words in 269 lines. Figure 29 demonstrates a drawing created with the Visual Basic function PB03_SPUR_GEAR.

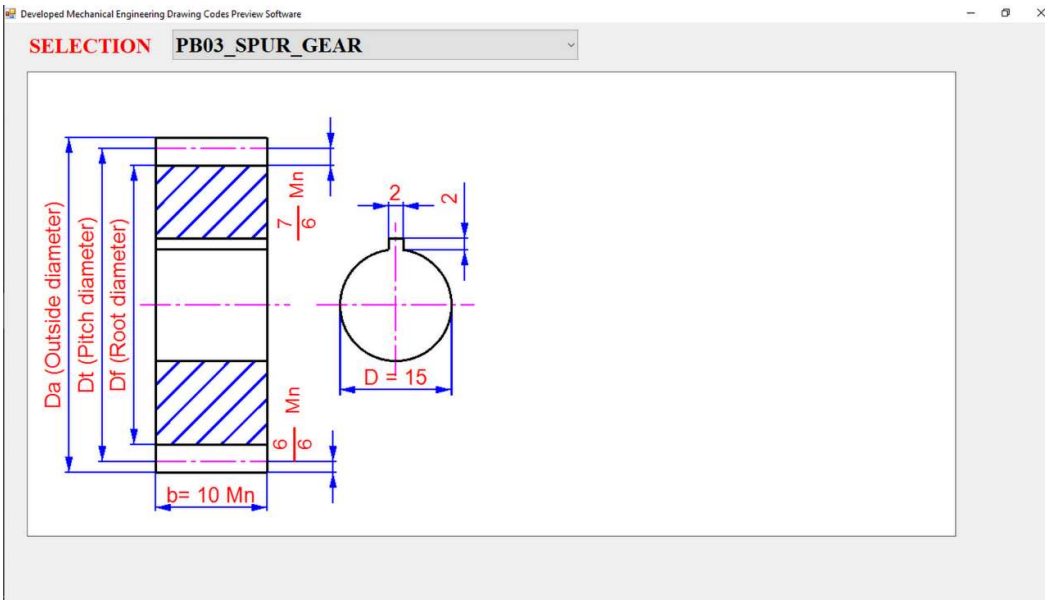

Figure 29. Sample Spur gear drawn with the code produced by the developed AutoLISP software

PB04_PROPELLER-LEFT.dwg

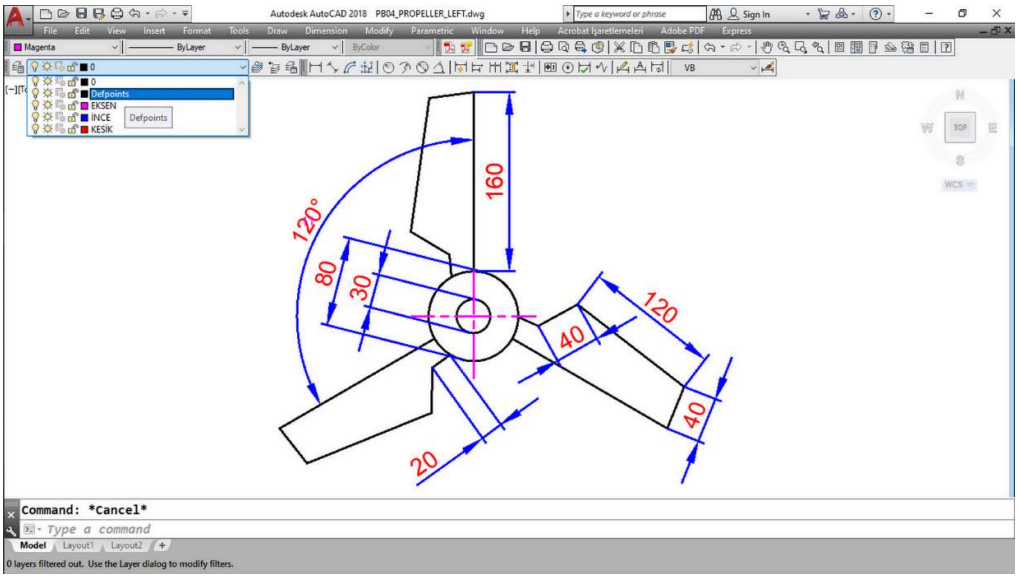

Figure 30. Propeller-Left drawing in Autocad environment

Figure 30 shows PB04_PROPELLER_LEFT drawing in Autocad environment. The size of the file named PB04_PROPELLER_LEFT.dwg is 88,9 KB (91.104 bayt) and contains 16 drawing objects.

The developed software "VB_BL" command produced the PB04_PROPELLER_LEFT function with 991 words in 254 lines. Figure 31 demonstrates a drawing created with the Visual Basic function PB04_PROPELLER_LEFT.

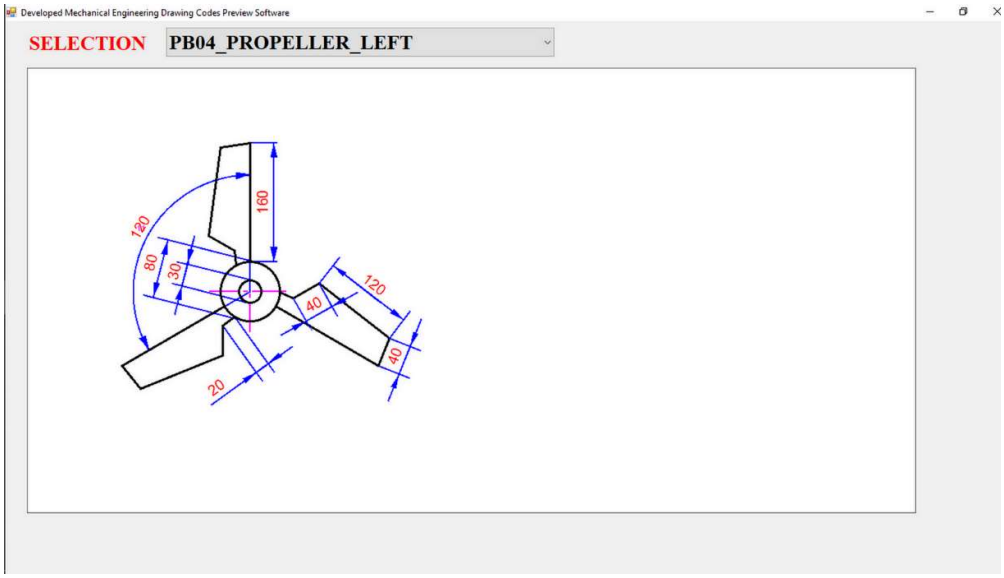

Figure 31. Sample Propeller-Left drawn with the code produced by the developed AutoLISP software

PB04_PROPELLER-RIGHT.dwg

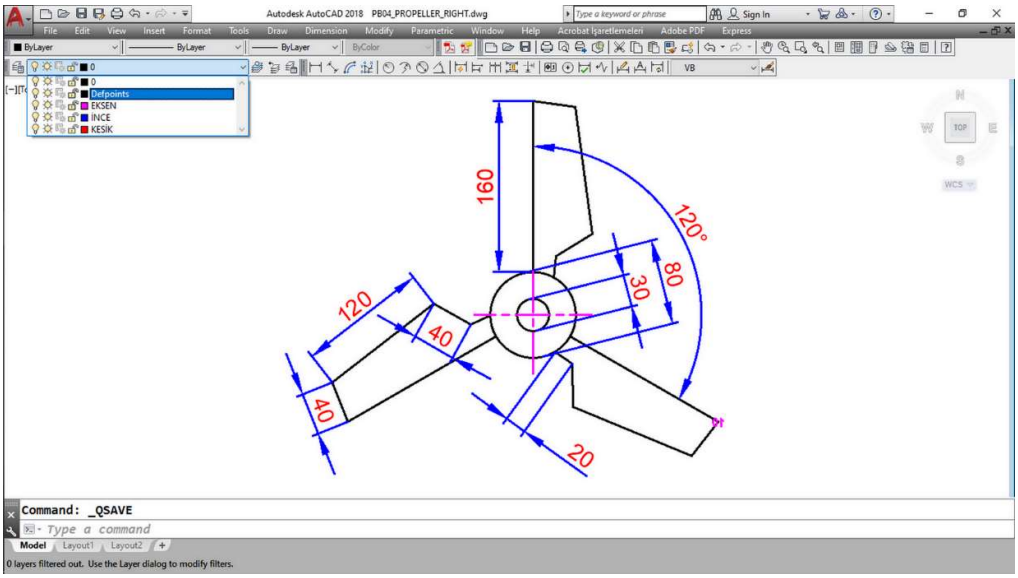

Figure 32. Propeller-Right drawing in Autocad environment

Figure 32 shows PB04_PROPELLER_RIGHT drawing in Autocad environment. The size of the file named PB04_PROPELLER_RIGHT.dwg is 103 KB (105.664 bayt) and contains 16 drawing objects.

The developed software "VB_BL" command produced the PB04_PROPELLER_RIGHT function with 991 words in 254 lines. Figure 33 demonstrates a drawing created with the Visual Basic function PB04_PROPELLER_RIGHT (Zak 2015).

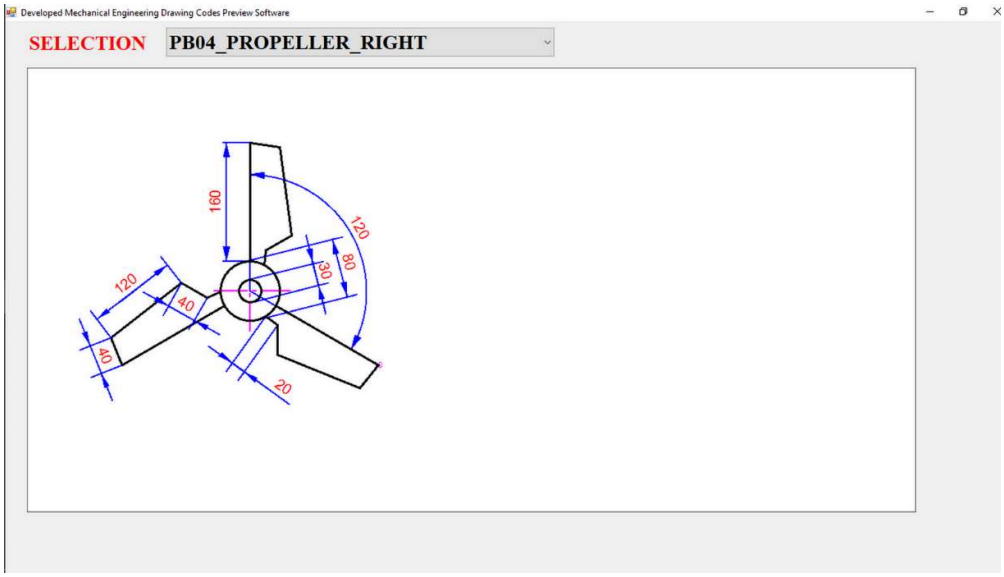

Figure 33. Sample Propeller-Right drawn with the code produced by the developed AutoLISP software

Example of an Animation Program

This section demonstrates the animation application by rotating the drawings. The propellers in Figure 34 rotate in opposite directions.

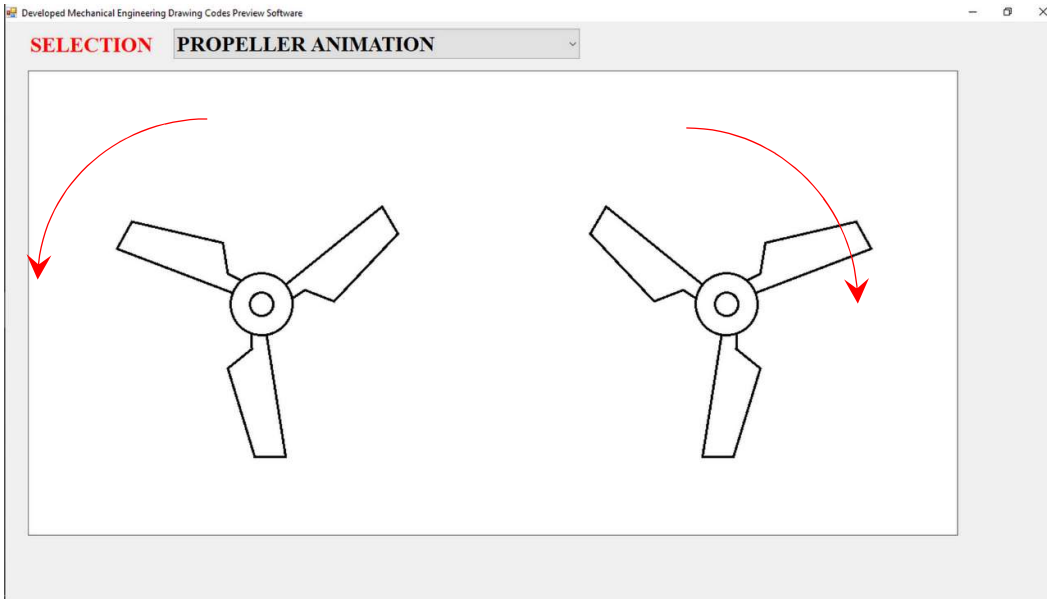

Figure 34. Sample Propeller Animation

Propeller Animation

Animation of drawings is shown in this section. Combobox usage codes are shown in Figure 35, enabling selection of the sub program to run (Grundgeiger 2018). The relevant drawing sub programs are run in the selections 0, 1, 2, 3, and 4.

```
Private Sub ComboBox1_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles ComboBox1.SelectedIndexChanged

Timer1.Enabled = False

If ComboBox1.SelectedIndex = 0 Then PB01_BOLT()

If ComboBox1.SelectedIndex = 1 Then PB02_ARC()

If ComboBox1.SelectedIndex = 2 Then PB03_SPUR_GEAR()

If ComboBox1.SelectedIndex = 3 Then PB04_PROPELLER_LEFT(, , 300, 300)

If ComboBox1.SelectedIndex = 4 Then PB04_PROPELLER_RIGHT(, , 300, 300)

If ComboBox1.SelectedIndex = 5 Then

Timer1.Interval = 10

Timer1.Enabled = True

End If

End Sub
```

Figure 35. Selection-based animation and display of drawings

When Combobox1.SelectedIndex is 5, the timer interval is set to 10 and the timer is turned on as shown in Figure 35.

```
Private Sub Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick
Static ANGLE = 0
ANGLE += 3
PB04_BL_MAIN(, ANGLE)
End Sub
```

Figure 36. Timer1_Tick

Static variables in the Visual Basic programming language do not lose their last value while inside a function in Figure 36. As a result, the ANGLE variable continues to increase in 3 degree increments. Drawings are prepared by the PB04_BL_MAIN drawing function in accordance with the ANGLE variable. As a consequence, the graph is displayed by rotating in respect to the variables Angle and Interval.

```
Function PB04_BL_MAIN( _
Optional PB As PictureBox = Nothing, _
Optional ANGLE As Integer = 0, _
Optional DX As Integer = 0, _
Optional DY As Integer = 0) As Bitmap
If PB Is Nothing Then PB = PictureBox1

Dim P1 As New List(Of Point)
Dim RESIM As New Bitmap(PB.Width, PB.Height)
Dim CIZ As Graphics = Graphics.FromImage(RESIM)
CIZ.Clear(Color.White)
CIZ.TranslateTransform(0 + DX, PB.Height - DY)
CIZ.RotateTransform(ANGLE)
CIZ.ScaleTransform(1, -1)
Dim PEN0 As Pen = New Pen(Color.Black, 2)
Dim PEN1 As Pen = New Pen(Color.Black, 3)
Dim PEN2 As Pen = New Pen(Color.Blue, 2)
PEN2.LineJoin = Drawing2D.LineJoin.Bevel
Dim PEN3 As Pen = New Pen(Color.Red, 2.5)
PEN3.DashPattern = {6, 2}
Dim PEN4 As Pen = New Pen(Color.Magenta, 2)
PEN4.DashPattern = {20, 3, 3, 3, 3}

PB04_PROPELLER_LEFT(CIZ, , -ANGLE, 300, 300)
PB04_PROPELLER_RIGHT(CIZ, , ANGLE, 900, 300)
PB.Image = RESIM
Return RESIM
End Function
```

Figure 37. PB04_BL_MAIN

The command "CIZ.TranslateTransform(0 + DX, PB.Height - DY)" transforms the picturebox object's bottom left point to X0, Y0. By using the "CIZ.RotateTransform(ANGLE)" command, the graph can be rotated at the chosen angle. The Y+ axis will be oriented upwards with the "CIZ.ScaleTransform(1, -1)" command. The graphical contents of the "PB04_PROPELLER_LEFT" and "PB04_PROPELLER_RIGHT" functions are combined in the graphics environment CIZ as shown in Figure 37.


```
Function PB04_PROPELLER_LEFT( _
CIZ As Graphics, _
Optional PB As PictureBox = Nothing, _
Optional ANGLE As Integer = 0, _
Optional DX As Integer = 0, _
Optional DY As Integer = 0) As Bitmap
If PB Is Nothing Then PB = PictureBox1

Dim P1 As New List(Of Point)
Dim GR = CIZ.Save
CIZ.ResetTransform()
CIZ.TranslateTransform(0 + DX, PB.Height - DY)
CIZ.RotateTransform(ANGLE)
CIZ.ScaleTransform(1, -1)
Dim PEN0 As Pen = New Pen(Color.Black, 2)
Dim PEN1 As Pen = New Pen(Color.Black, 3)
Dim PEN2 As Pen = New Pen(Color.Blue, 2)
PEN2.LineJoin = Drawing2D.LineJoin.Bevel
Dim PEN3 As Pen = New Pen(Color.Red, 2.5)
PEN3.DashPattern = {6, 2}
Dim PEN4 As Pen = New Pen(Color.Magenta, 2)
PEN4.DashPattern = {20, 3, 3, 3, 3}

P1 = New List(Of Point)
P1.Add(New Point(34.64, -20))
P1.Add(New Point(34.64, -20))
P1.Add(New Point(173.21, -100))
P1.Add(New Point(148.48, -131.44))
P1.Add(New Point(37.44, -85.93))
P1.Add(New Point(36.91, -45.93))
P1.Add(New Point(20.68, -34.24))
CIZ.DrawLines(PEN1, P1.ToArray)
P1 = New List(Of Point)
P1.Add(New Point(-34.64, -20))
P1.Add(New Point(-34.64, -20))
P1.Add(New Point(-173.21, -100))
P1.Add(New Point(-188.07, -62.86))
P1.Add(New Point(-93.14, 10.54))
P1.Add(New Point(-58.23, -9))
P1.Add(New Point(-39.99, -0.79))
CIZ.DrawLines(PEN1, P1.ToArray)
CIZ.DrawEllipse(PEN1, -40, -40, 80, 80)
CIZ.DrawEllipse(PEN1, -15, -15, 30, 30)

P1 = New List(Of Point)
P1.Add(New Point(0, 40))
P1.Add(New Point(0, 40))
P1.Add(New Point(0, 200))
P1.Add(New Point(39.59, 194.31))
P1.Add(New Point(55.69, 75.39))
P1.Add(New Point(21.32, 54.93))
P1.Add(New Point(19.32, 35.03))
CIZ.DrawLines(PEN1, P1.ToArray)
CIZ.Restore(GR)
End Function
```

Figure 38. PB04_PROPELLER_LEFT

The software in Figure 38 was produced from Autocad drawing by the software developed in this study. It is necessary to use the graphic definition CIZ as a function variable in order to display two drawings in a single picturebox.


```
Function PB04_PROPELLER_RIGHT( _
CIZ As Graphics, _
Optional PB As PictureBox = Nothing, _
Optional ANGLE As Integer = 0, _
Optional DX As Integer = 0, _
Optional DY As Integer = 0) As Bitmap
If PB Is Nothing Then PB = PictureBox1

Dim P1 As New List(Of Point)
Dim GR = CIZ.Save
CIZ.ResetTransform()
CIZ.TranslateTransform(0 + DX, PB.Height - DY)
CIZ.RotateTransform(ANGLE)
CIZ.ScaleTransform(1, -1)
Dim PEN0 As Pen = New Pen(Color.Black, 2)
Dim PEN1 As Pen = New Pen(Color.Black, 3)
Dim PEN2 As Pen = New Pen(Color.Blue, 2)
PEN2.LineJoin = Drawing2D.LineJoin.Bevel
Dim PEN3 As Pen = New Pen(Color.Red, 2.5)
PEN3.DashPattern = {6, 2}
Dim PEN4 As Pen = New Pen(Color.Magenta, 2)
PEN4.DashPattern = {20, 3, 3, 3, 3}

P1 = New List(Of Point)
P1.Add(New Point(34.64, -20))
P1.Add(New Point(34.64, -20))
P1.Add(New Point(173.21, -100))
P1.Add(New Point(188.07, -62.86))
P1.Add(New Point(93.14, 10.54))
P1.Add(New Point(58.23, -9))
P1.Add(New Point(39.99, -0.79))
CIZ.DrawLines(PEN1, P1.ToArray)
CIZ.DrawEllipse(PEN1, -40, -40, 80, 80)
CIZ.DrawEllipse(PEN1, -15, -15, 30, 30)

P1 = New List(Of Point)
P1.Add(New Point(-34.64, -20))
P1.Add(New Point(-34.64, -20))
P1.Add(New Point(-173.21, -100))
P1.Add(New Point(-148.48, -131.44))
P1.Add(New Point(-37.44, -85.93))
P1.Add(New Point(-36.91, -45.93))
P1.Add(New Point(-20.68, -34.24))
CIZ.DrawLines(PEN1, P1.ToArray)

P1 = New List(Of Point)
P1.Add(New Point(0, 40))
P1.Add(New Point(0, 40))
P1.Add(New Point(0, 200))
P1.Add(New Point(-39.59, 194.31))
P1.Add(New Point(-55.69, 75.39))
P1.Add(New Point(-21.32, 54.93))
P1.Add(New Point(-19.32, 35.03))
CIZ.DrawLines(PEN1, P1.ToArray)
CIZ.Restore(GR)
End Function
```

Figure 39. PB04_PROPELLER_RIGHT

The software in Figure 39 was produced from Autocad drawing by the software developed in this study. It is necessary to use the graphic definition CIZ as a function variable in order to display two drawings in a single picturebox.

Assessment and Recommendations

Supporting software with graphics is of great importance. Graphics make programs more understandable and their presentation more effective. Programs written on some subjects cannot achieve their purpose without drawing support. Such situations reveal the necessity of using graphic codes in software. Graphics production with code is very difficult without a drawing resource, and unexpected erroneous and inadequate results occur. Therefore, our recommendation is to prepare the necessary graphics in a program such as Autocad.

Persons who prepare the drawings do not need to know code. This will ensure the collective work of Autocad experts and software developers who do not know how to code in software development. It will be more efficient, easy and reliable for programmers to write code by taking graphic information from previously prepared drawings. Despite all these advantages, drawing large graphics and then converting them into codes requires attention, effort and time.

In this study, the new VB_BL command, which converts AutoCAD drawings to Visual Basic codes with the AutoLISP programming language, has been added to the Autocad environment. Thus, very long and complex software codes can be obtained from all defined drawing objects in a very short time. Text, Line, Lwpolyline, Circle, Arc, RotatedDimension, AlignedDimension, RadialDimension, AcDb2LineAngularDimension and 2D solid drawing objects are converted to Visual Basic codes. In transcoding, it is sufficient to select the desired drawings using the VB_BL command. Thus, the drawing codes function can be obtained in a very short time in text format. Visual preparation of drawings provides quality graphic codes. There are no user errors in transcoding from drawings.

Bolt, Arc, Spur_Gear, Propeller-Left, Propeller-Right drawings were made in Autocad. Visual Basic codes of these drawings were obtained in a very short time by using the developed software. The use of the codes obtained in the Visual Studio environment is shown. PROPELLER Animation is provided by combining the codes and making additional arrangements and the application is demonstrated.

With the VB_BL software developed in this study, it is expected that the development of graphical engineering software will be easier and faster. In the development of software that requires drawing and animation, such as engines, hydraulics, pneumatics, measuring instruments and gears, valuable programs can be prepared with less effort and time by using the drawings prepared in the Autocad environment. Visual information pages containing graphics can be easily designed in software. In addition, it may be possible to develop more valuable programs with rich graphics support. It can be recommended that this work, which is done with the AutoLISP programming language, should be done for different purposes such as other programming languages or production.

References

Ambrosius, L. (2015). *AutoCAD Platform Customization: User Interface, AutoLISP*, VBA, and Beyond. John Wiley & Sons.

Ayyildiz, M., Cicek, A., & Kara, F. (2010). Parametric gear wheel applications in computer aided design, *Journal of the Faculty of Engineering and Architecture of Gazi University* **25**(3), 643-651.

Grundgeiger, D. (2018). *Programming Visual Basic. NET*, O'Reilly.

Kashid, S., & Kumar, S. (2014). An expert system for selection of components of compound die, *Journal of Advanced Manufacturing Systems* **13**(03), 181-195.

Khalili, K., Ahmadi-brooghani, S. Y., & Rakhshkhorshid, M. (2007). CAD Model Generation Using 3D Scanning, *Advanced Materials Research* **23**, 169-172.

Molotnikov, V., & Molotnikova, A. (2023). Elements of Programming in the AutoLISP language, *In Theoretical and Applied Mechanics*, Springer, Cham, 645-666.

Naranje, V., & Kumar, S. (2014). A knowledge based system for automated design of deep drawing die for axisymmetric parts, *Expert Systems with Applications* **41**(4), 1419-1431.

Singh, R., & Sekhon, G. (1999). An expert system for optimal selection of a press for a sheet metal operation, *Journal of Materials Processing Technology* **86**(1-3), 131-138.

Stojkovic, Z., & Stankic, Z. (2006). AutoCAD-based concept for estimating lightning protection zone of transmission lines and structures, *International journal of electrical engineering education* **43**(4), 299-317.

Türkyılmaz, T. (1997). *Calculations And Drawings Of Bending Beams in Autocad Environment*, Unpublished Master's Thesis, Gazi University, Institude of Educational Sciences.

Zak, D. (2015). *Programming with Microsoft Visual Basic 2015*, Cengage Learning.

ABOUT THE AUTHOR

Telat Türkyılmaz is an assistant professor at Bilecik Şeyh Edebali University Vocational School, Department of Machinery and Metal Technologies. His research interests include CAD, CAM, Autolisp, Visual Basic, Technical Software Development, Wind-Solar energy, Hybrid Energy.