

Drawing 3D Entities with Visual Basic

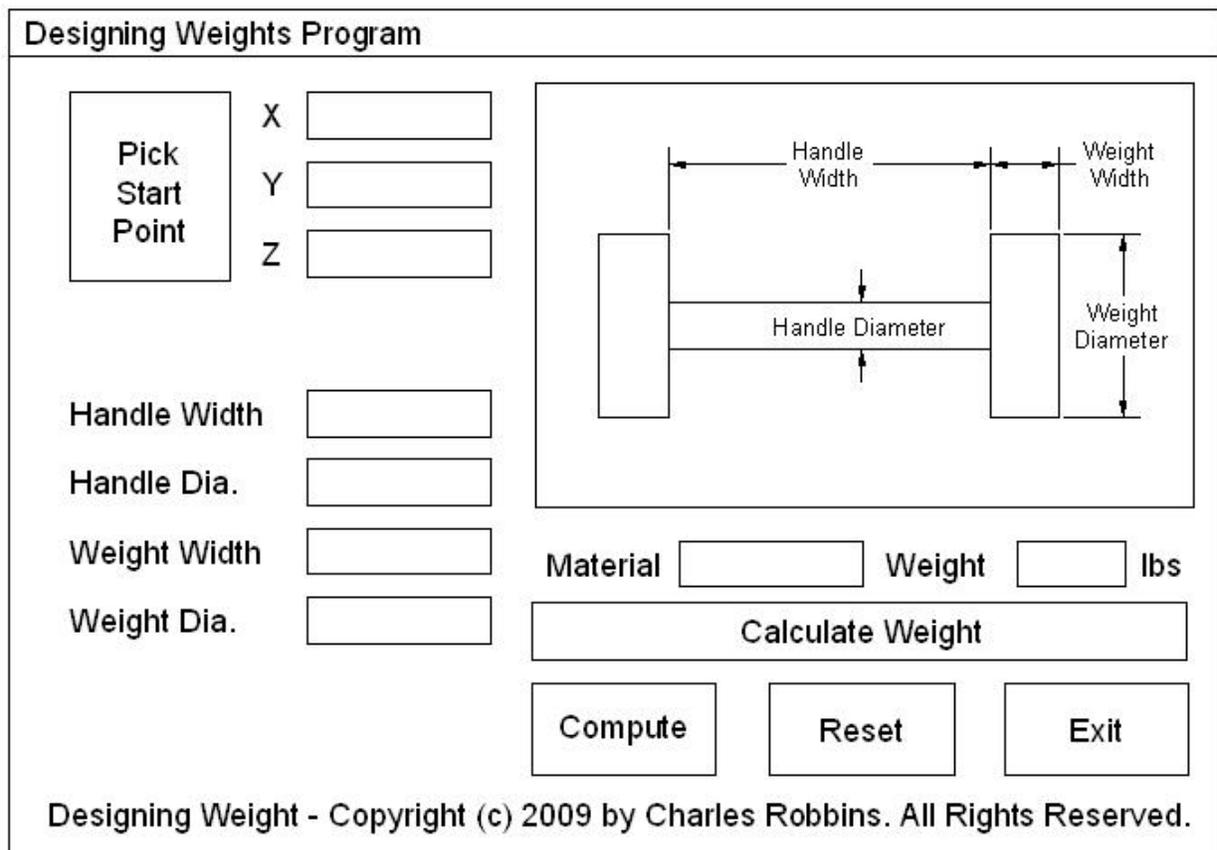
In this chapter, you will learn how to use the following VBA functions to World Class standards:

- **Beginning a New Visual Basic Application**
- **Opening the Visual Basic Editor in AutoCAD**
- **Laying Out a User Input Form in Visual Basic**
- **Creating and Inserting an Image into a Form in Visual Basic**
- **Insert a Label into a Form**
- **Insert a Textbox into a Form**
- **Insert a Listbox into a Form**
- **Insert Command Buttons into a Form**
- **Adding a Copyright Statement to a Form**
- **Adding Comments in Visual Basic to Communicate the Copyright**
- **Declaring Variables in a Program with the Dimension Statement**
- **Setting Variables in a Program**
- **Assigning Values to the Variables**
- **Inputting the Code to Draw 3D Objects in Visual Basic**
- **Union Two Solids in Visual Basic**
- **Populating the Material Listbox**
- **Resetting the Data with the cmdClear Command Button**
- **Exiting the Program with the cmdExit Command Button**
- **Exiting the Program with the cmdExit Command Button**
- **Executing a Subroutine with the cmdPickPoint Command Button**
- **Executing a Subroutine with the cmdDraw Command Button**
- **Inserting a Module into a Visual Basic Application**
- **Running the Program**

Beginning a New Visual Basic Application

In this chapter, we will continue to learn how to use the Visual Basic Application (VBA) program to create a form and then to generate a 3D drawing automatically. We reiterate many elements of the previous lesson, but now we have the capability to 3D entities in AutoCAD Model Space.

At the beginning of every chapter, we will start a new Visual Basic Application project, use a sketch to determine the extent of what the program will do, create the form and then write the code. Once the code is finished, we will run the program and an orthographic drawing will appear on the graphical display.



Designing Weights Program

Pick Start Point
X
Y
Z

Handle Width
Handle Dia.
Weight Width
Weight Dia.

Material Weight lbs

Calculate Weight

Compute **Reset** **Exit**

Designing Weight - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.

Figure 11.1 – Rough Sketch of the Designing Weights Form

Remember, that all programming projects begin with one or more sketches, with one portraying the part, detail, or assembly and the other being the user input form. In this Visual Basic Project, Designing Weights, we will be running a user input form inside the AutoCAD application, so we need to sketch the structure of this special dialogue box. We will name the Input form, **Designing Weights**. We will place seven textboxes on the left side of the form to input the starting point, the handle width, the handle diameter, the weight width, and the weight diameter. We add a list box to allow the user to pick Steel or Aluminum and a label to hold the answer when we pick the calculate weight command button. On the right side of the form, we

will place an image of the dumbbell. We will have five command buttons, **Draw**, **Clear**, **Exit**, **Pick Start Point** and **Calculate Weight**. On the bottom of the form, we will write the copyright statement using another label. On this presentation, we can help ourselves by being as accurate as possible, by displaying sizes, fonts, colors and any other specific details which will enable us to quickly create the form. From the beginning of inserting the form into the project, we need to refer to our sketch. The sketch of the form is shown in Figure 11.1.

Remember, we should train new programmers initially in the art of form building. When using the editor, we insert and size the form, and selecting the Controls Toolbox, we will place all the various input tools and properly label them. Whenever we place an input tool, the properties window will display a list of every attribute associated with the tool, and we will take every effort to arrange the tool by performing such actions as naming, labeling and sizing the visual input device.

Opening the Visual Basic Editor in AutoCAD

Opening the Visual Basic Editor in AutoCAD is essential to creating the program to automate the drawing process. In this version of the World Class CAD – Visual Basic Applications for AutoCAD, we are using AutoCAD 2009, but we just finished using all the programs in this text with a group programming in AutoCAD 2002. Their drawings were automatically made just as efficiently as if they were using the most recent version of the Autodesk software.

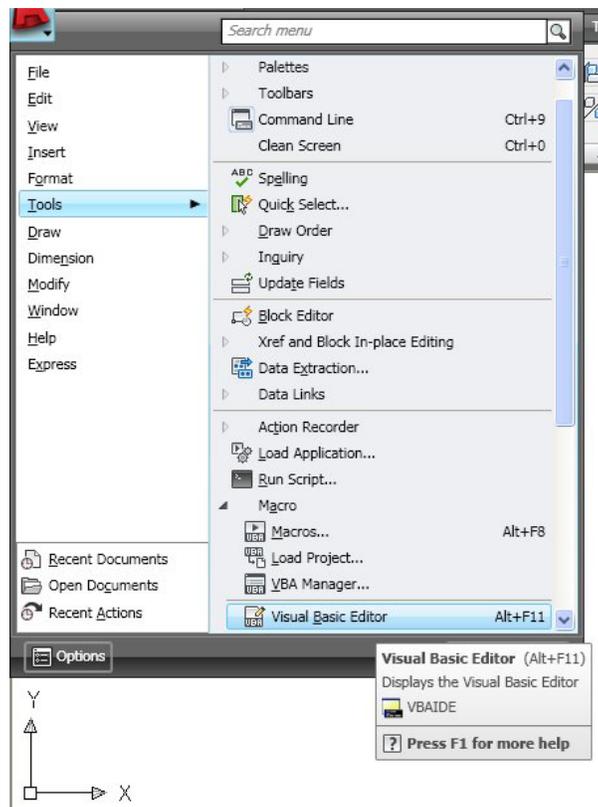


Figure 11.2 – Launching the Visual Basic Editor

Select Tools on the AutoCAD main menu; pick Macro and then choose the Visual Basic Editor. Look to the right of the phrase, Visual Basic Editor and the shortcut keys Alt – F11 is noted. For quick launching of the editor, press Alt – F11

The Visual Basic Editor will appear on the computer desktop as a new program application. Looking down on the computer’s Taskbar, we can see the AutoCAD and Microsoft Visual Basic Editor program tabs. Just single click either program tab to switch between any applications. However, if we close the AutoCAD drawing, unlike a stand alone version of Visual Basic, the Visual Basic Editor will also close.

For those individuals with previous Visual Basic experience, the Visual Basic Editor in AutoCAD has the same layout as in other VB programs. The Menu Bar contains tools for our use as well as the four toolbars shown in Figure 11.4, which are Standard, Debug, Edit and Userform. Presently, only the Standard toolbar is showing. On the left side of the workspace is the Project menu, which shows the files pertaining to this project. Below the Project menu is the Properties pane. If we remember the Properties tool in AutoCAD, using this device will be simple.

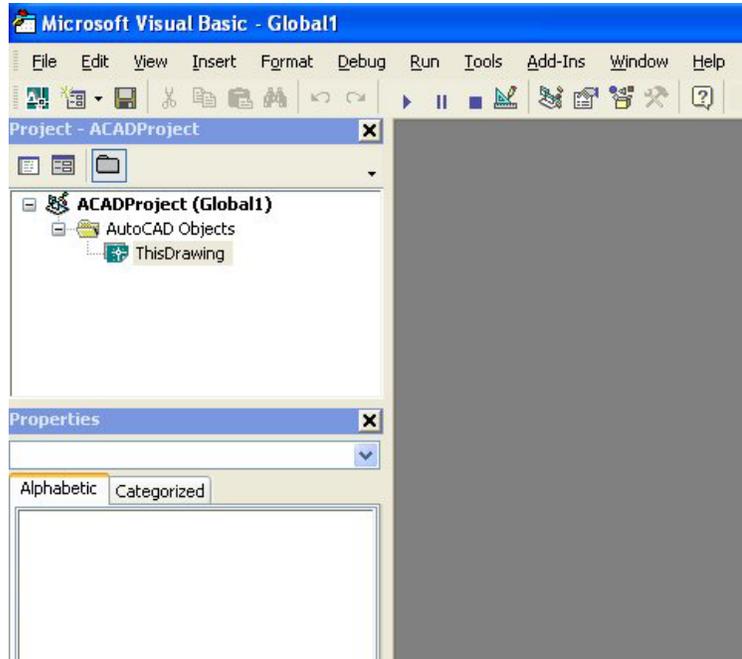


Figure 11.3 – The Visual Basic Editor

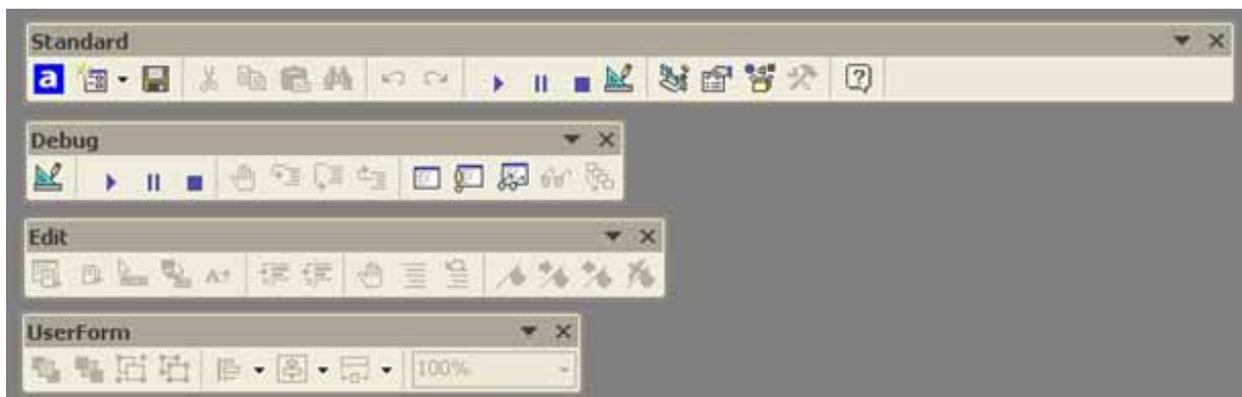


Figure 11.4 – Toolbars in the Visual Basic Editor

With the Visual Basic Editor open, select **File** on the Menu Bar and select **Save Project**. Remember, we have a folder on either the desktop or in the My Documents folder called “VBA Programs”. Save the project with the filename “Designing Weights”. The file has an extension called *dvb* which means DCL and Visual Basic programs as shown in Figure 11.5.

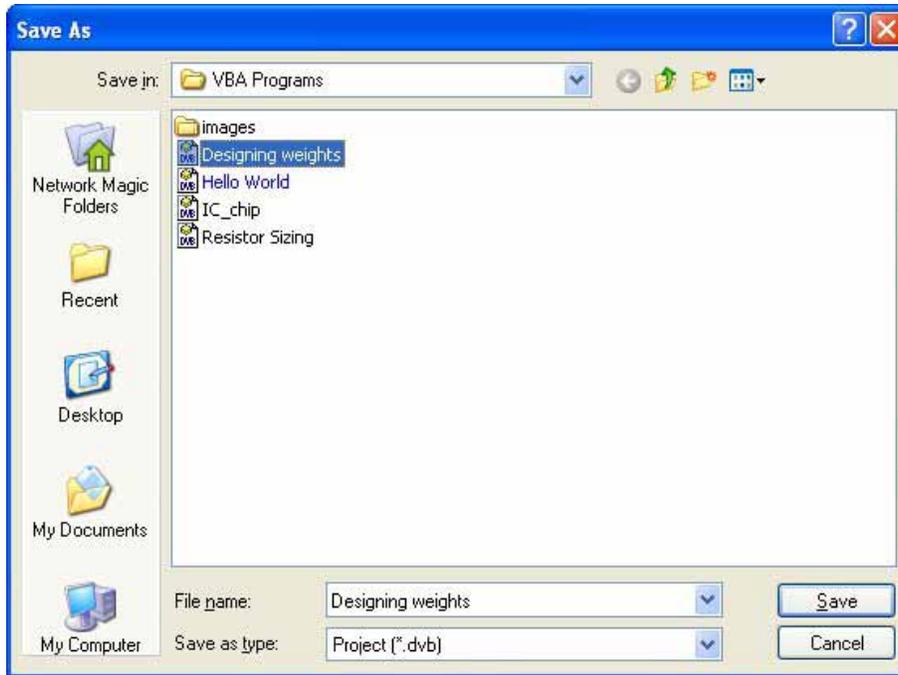


Figure 11.5 – Saving the Designing Weights Program

Laying Out a User Input Form in Visual Basic

Now that we have an idea of what the dialogue box in our program will look like, select the **Insert UserForm** button on the Standard toolbar to insert a new form as shown in Figure 11.6. Instantaneously, the once grey work area is changed to contain our UserForm1. A Form folder with Userform1 is now in the Project menu and the Properties pane contains the attributes associated with UserForm1. (See Figure 11.7)

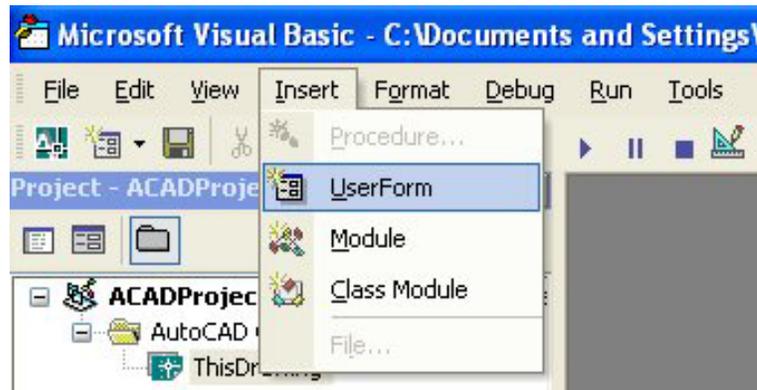


Figure 11.6 – Inserting a User Form

Change the name of the user form to frmICchip. We use the frm prefix in front of all of the form names in Visual Basic. Change the background of the form to light blue by setting the BackColor in the Properties Pane on the left side of the Visual Basic Application window to “&H80000013&”.

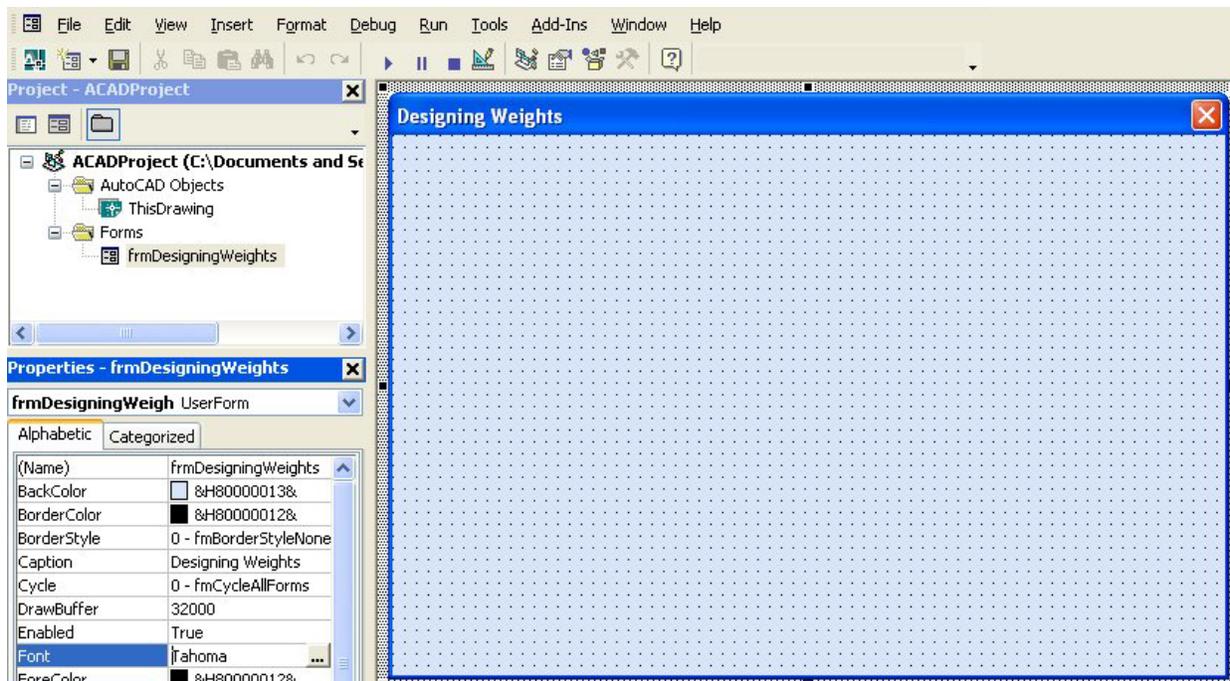
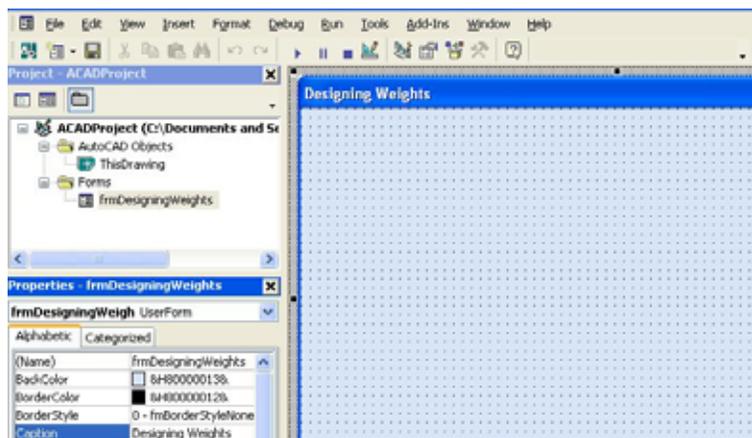


Figure 11.7 – Creating the Designing Weights Form in Visual Basic

Next, we will change the **Caption** in the Properties pane to **Integrated Circuit Drawing** to agree with the sketch in Figure 11.8. Go ahead and change the form in two other aspects, Height and Width.



Alphabetic	
(Name)	frmDesigningWeights
BackColor	&H80000013&
Caption	Designing Weights
Height	325
Width	490

Figure 11.8 – Setting the Caption and other Properties

The form will change in size to the height and width measurement. The background color will change to a light blue. There are many more attributes in the Properties pane that we will use on future projects.

In previous chapters, we set the Font and Font size for the labels, textboxes and command buttons after creating those specific interfaces. If we set the Font to Tahoma and the Font size to 14 on the form, then all of the labels, textboxes and command buttons that we insert from the Control Toolbox will already be set to those attributes.

On the left side of the Visual Basic Editor, locate the property that controls the font and font size in the Properties window. When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Tahoma. Click on the three dotted button to open the Visual Basic Font window.

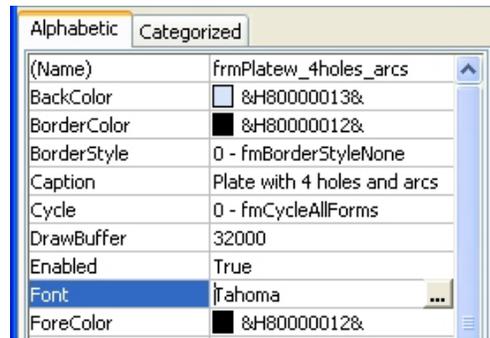


Figure 11.9 – Changing the Font to Tahoma

We will select the Tahoma font, Regular font style and 14 size for this project to agree with the initial sketch if the user input form. When we adjust the attributes for the label, these changes do not alter globally for the other objects on the form. If we wish to underline the text or phrase in the label, add a check to the Underline checkbox in the Effects section of the Font window. When we finish making changes to the font property, select the OK command button to return to the work area.

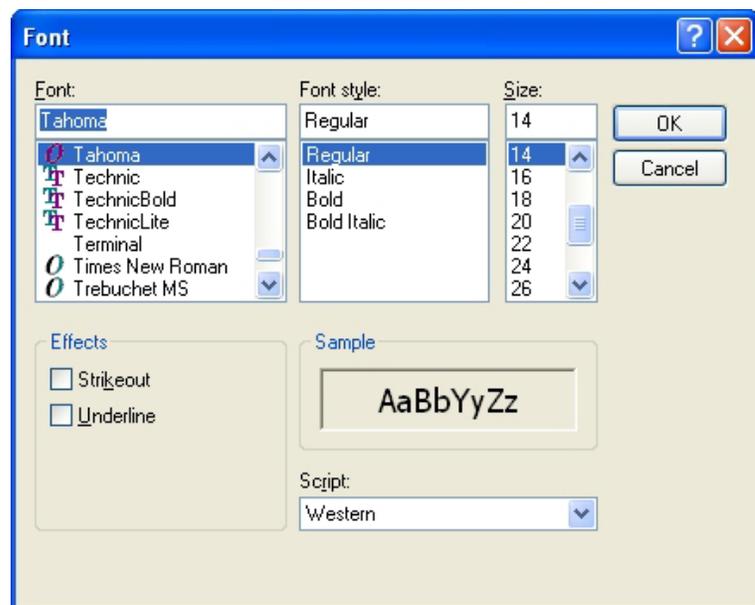


Figure 11.10 – The Font Window in Visual Basic

Creating and Inserting an Image into a Form in Visual Basic

As in previous chapters, this form will have a picture of the part that we will create automatically, so we need to make a drawing of part in AutoCAD. Dimension the drawing as we do in any other drawing, but we will use the Edit Text tool to remove the actual dimension and write in the word that matched the textbox label. In Figure 11.11, we show dimensions that associate with the handle width, the handle diameter, the weight width, and the weight diameter and starting point textboxes with the image. When the drawing is finished, we need to save the drawing as an image file. Use the **Saveimg** command to save file on the VBA Programs folder. Create a folder named Images in the VBA Programs folder and save the file as the same name as the program for matching purposes, designing_weights. We saved the file as a Bitmap with a width of 400 pixels and a height of 190 pixels.

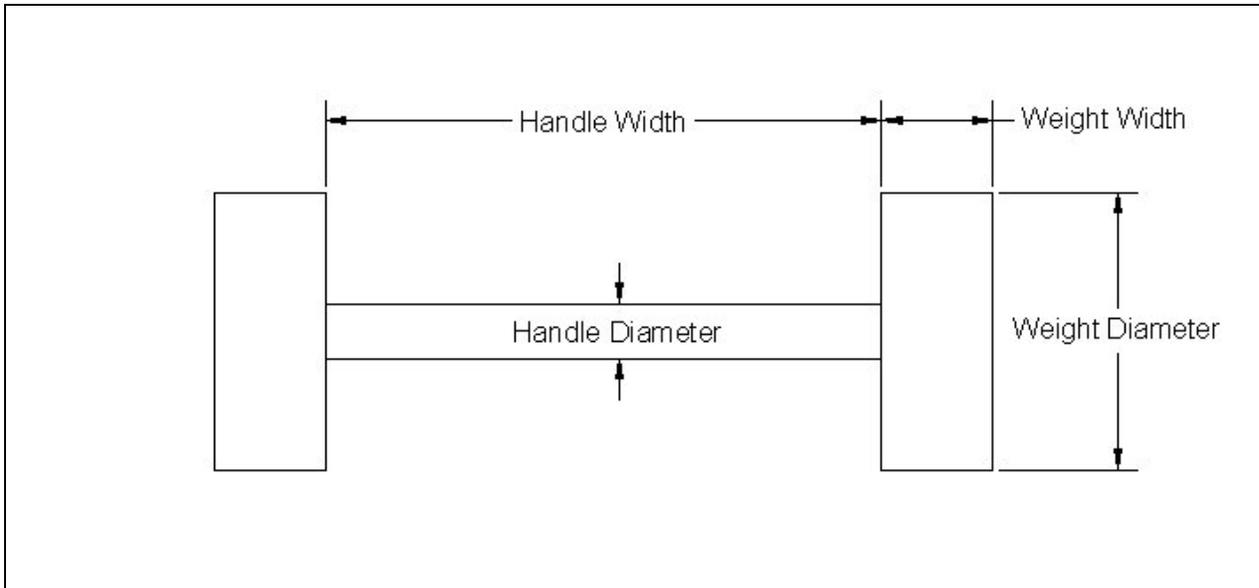


Figure 11.11 – Creating the Designing Weights Form Image in AutoCAD

On the control toolbox, select the Image tool and then draw a rectangular box on the form in the upper right corner as shown in Figure 11.13. After outlining the size of the image, we will direct the program to the folder and filename of the digital image. In the Properties – Image pane, select the attribute named Picture. With the mouse, select the three dot box in the empty cell to the right of Picture. The Load Picture window appears on the screen. Go to the VBA Programs folder and then the Images folder. Select the file, designing_weights and it will appear in the picture frame.

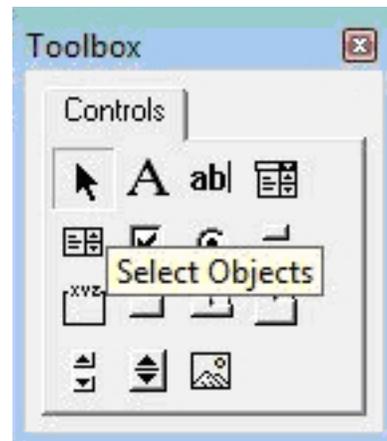


Figure 11.12 – The Control Toolbox

In the Properties pane set the image name to imgWeights, the width to 300 and the height to 144. The image will finally appear as shown in Figure 11.14.

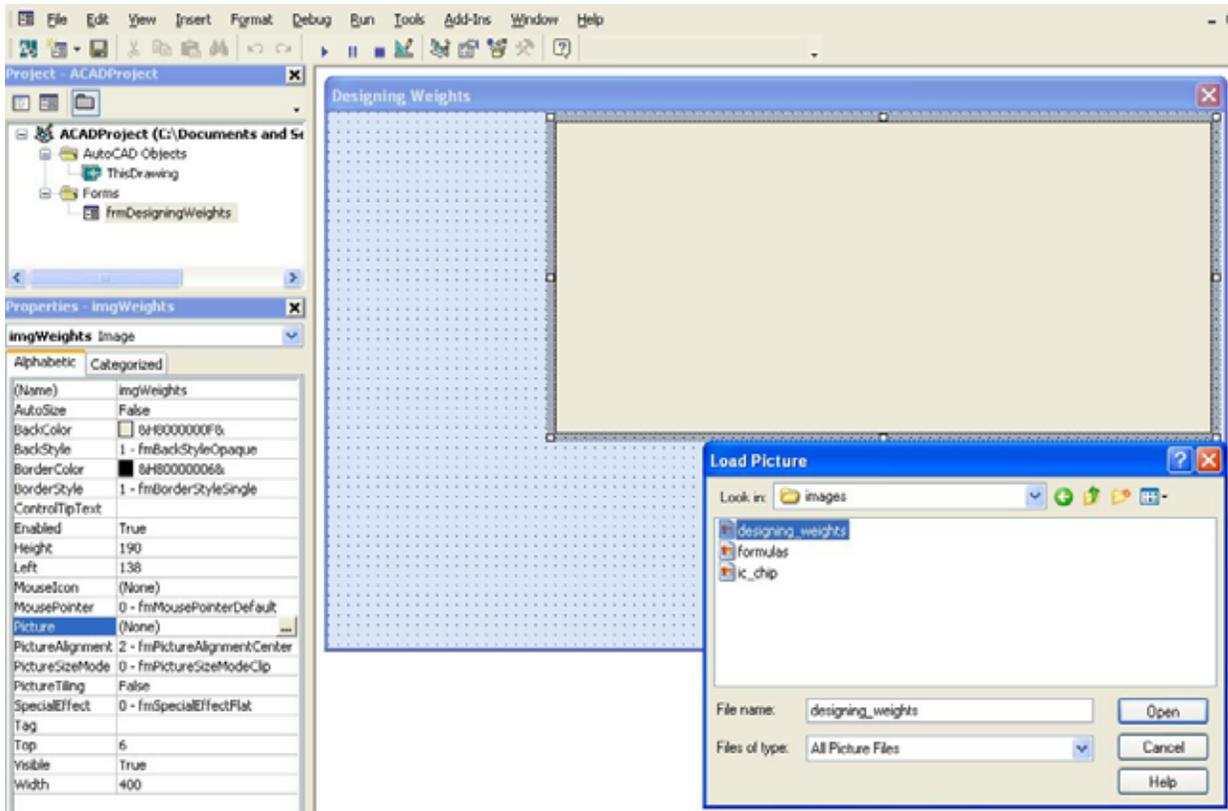


Figure 11.13 – Placing an Image on the Form

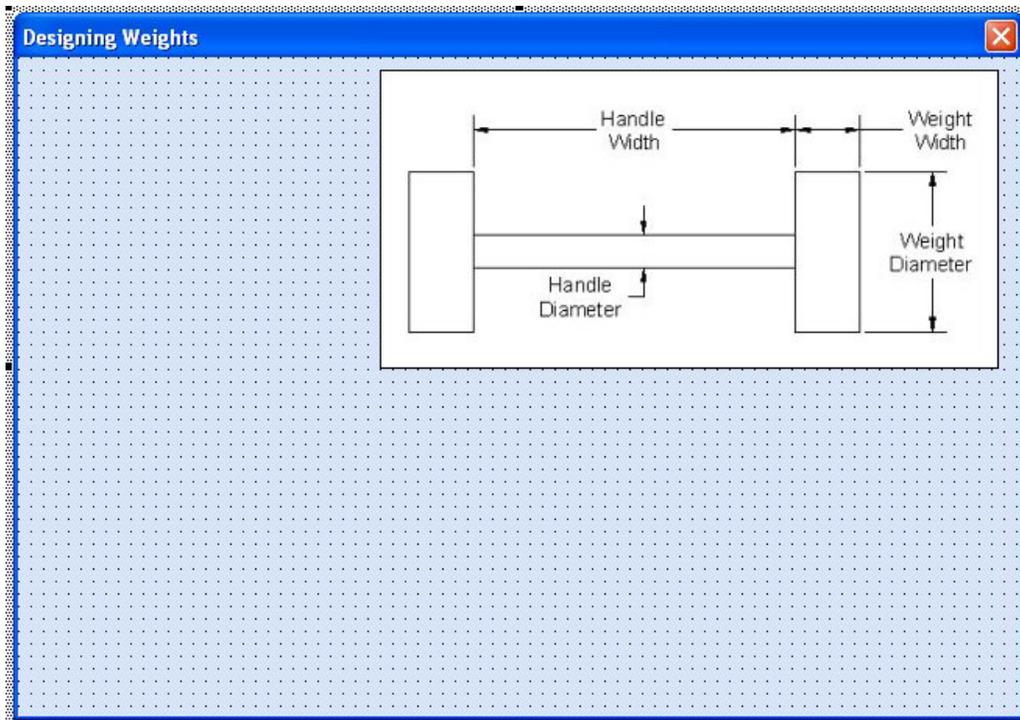


Figure 11.14 – Placing an Image on the Form

Inserting a Label into a Form

A good form is easy to figure out by the user, so when we are attempting to provide information on the window that will run in AutoCAD; we add labels to textboxes to explain our intent. Press the Label (A) button on the Control Toolbar to add a label. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted label box as shown in the sketch.

When the first label is done, the background color of the label matches the background color of the form. In many cases that effect is visually pleasing to the eye, versus introducing another color. Both color and shape will direct the user in completing the form along with the explanation we place on the window to guide the designer in using the automated programs. Use colors and shape strategically to communicate well.

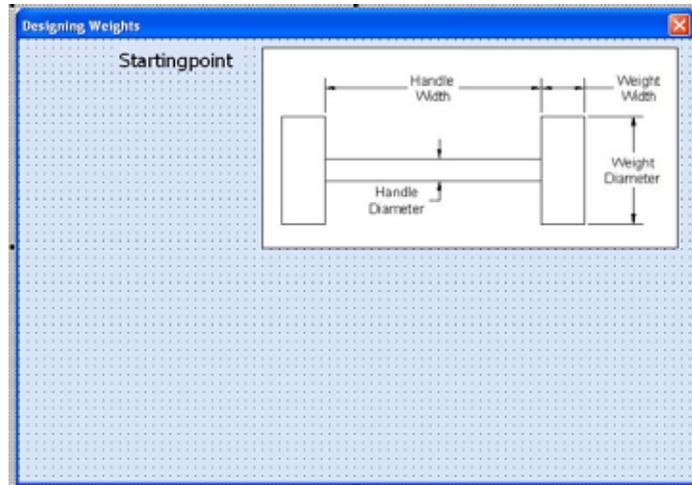


Figure 11.15 – The Finished Label on the Form

For the first label, set the name as **lblStartingpoint** and the caption as Startingpoint. The width of the textbox is 90 and the height is 18. For labels on the top side of the textbox, set the TextAlign attribute to center justification.

Inserting a Textbox into a Form

A textbox is used so that a user of the computer program can input data in the form of words, numbers or a mixture of both. Press the TextBox (ab) button on the Control Toolbar to add a textbox. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted textbox as shown in Figure 11.16.

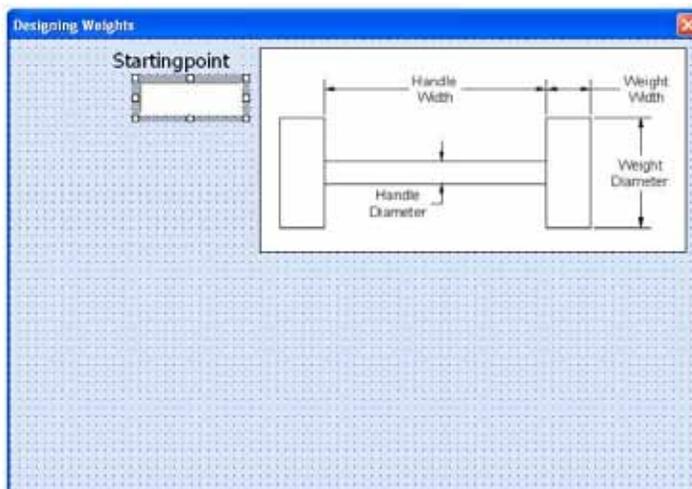


Figure 11.16 – Placing a TextBox on the Form

We will name the TextBox using the three letter prefix followed by the name or phrase of the tool. For our first textbox, the name is **txtSpX**.

Alphabetic	
(Name)	txtSpX
Height	24
Width	72

We place a Label using a common Visual Basic naming convention **lblSpX** just to the left of the Textbox. The Caption for the Label will be **X**. On all of the labels that are just to the left of the Textboxes, we will align the text to the right by setting the **TextAlign** property to right align.

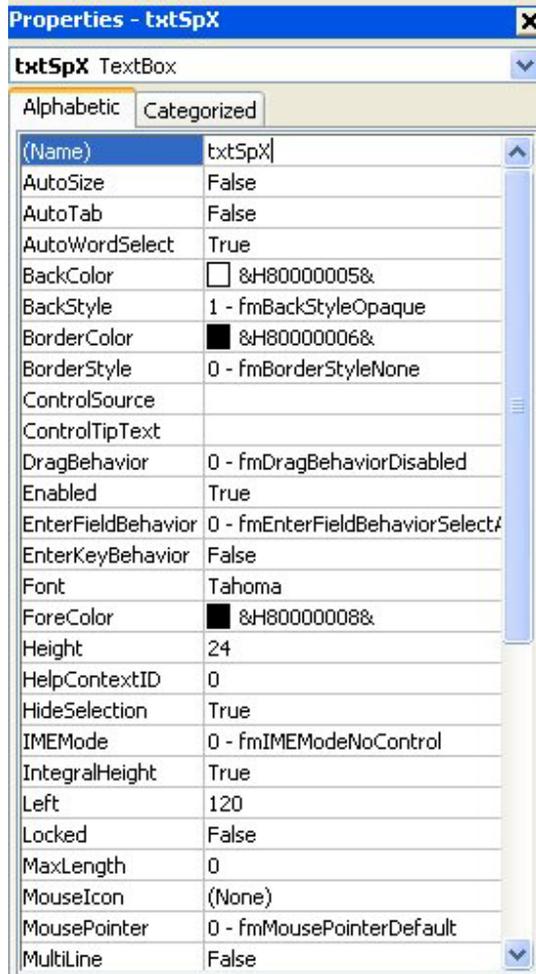


Figure 11.17 – Changing the (Name) to txtName

We will add another TextBox named **txtSpY** under the first one and the Label to the left of the textbox is called **lblSpY**. The Caption for the Label will be **Y**.

We will add yet another TextBox named **txtSpZ** under the first one and the Label to the left of the textbox is called **lblSpZ**. The Caption for the Label will be **Z**.

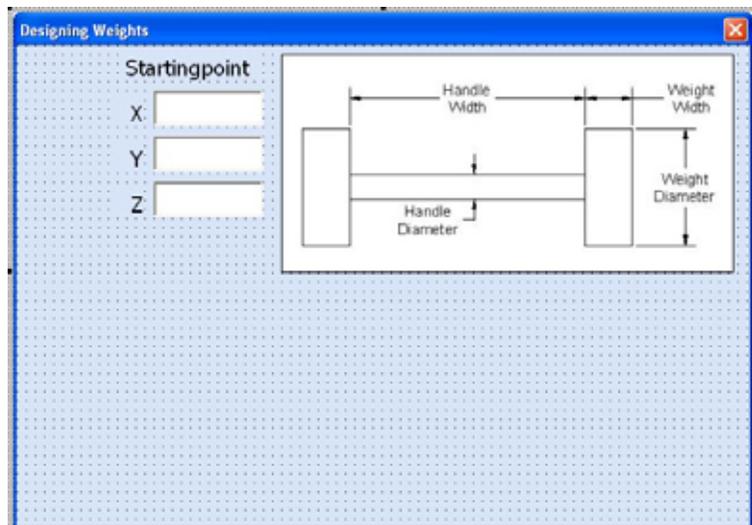


Figure 11.18 – Adding the Y and Z Textboxes

We will add four more textboxes named

txtHandleWidth,
txtHandleDiameter,
txtWeightWidth
txtWeightDiameter

under the X, Y and Z textboxes. The labels to the left of the textbox are called

lblHandleWidth,
lblHandleDiameter,
lblWeightWidth
lblWeightDiameter

The Captions for the Labels are shown in Figure 11.19.

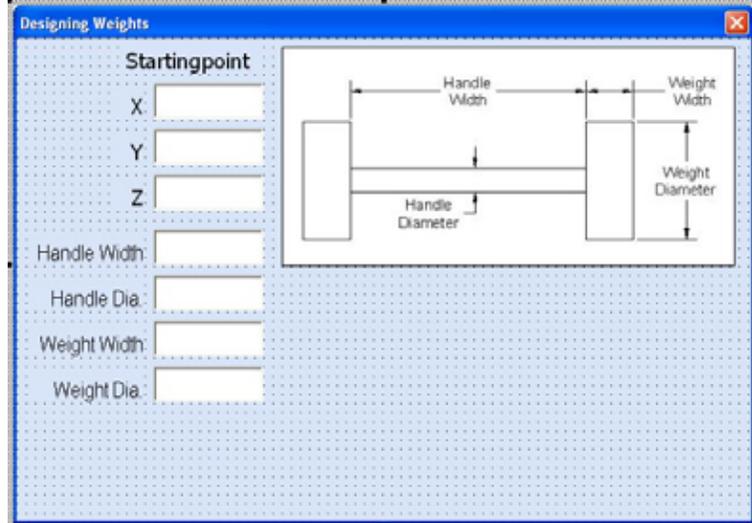


Figure 11.19 – Adding Four More Textboxes

Inserting a Listbox into a Form

We use a Listbox to allow the Designing Weights program the ability to select the type of material that the manufacturer of the dumbbell will use. Select the Listbox tool on the Control toolbar as shown in figure 11.20 and place the Listbox on the form with the Material label as shown in figure 11.21. In other chapters, we will use a combo box, where the user can select from the list, or type in their own entry.

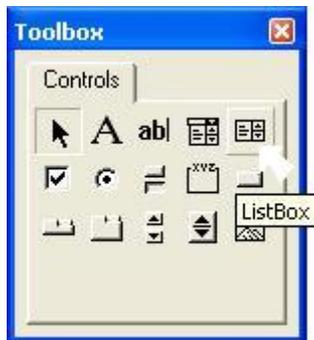


Figure 11.20– Adding a Listbox

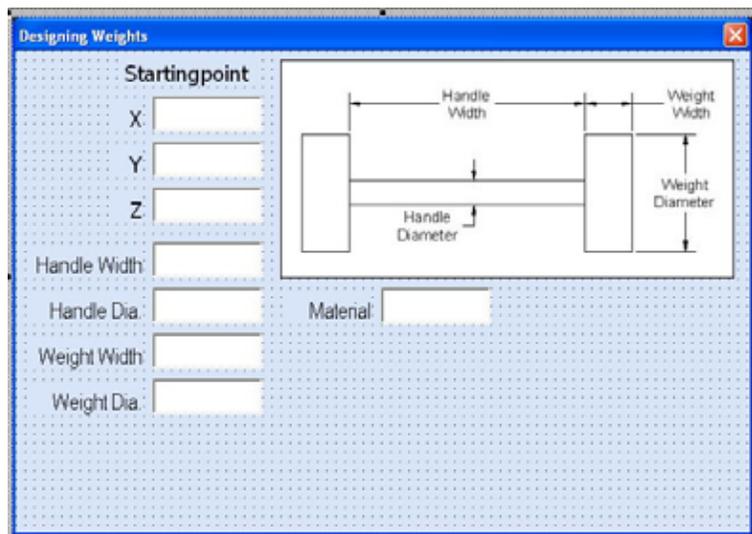


Figure 11.21– Adding a Listbox into the Form

To the right of the Material Listbox, we add the Weight label with a blank caption. Call this label **lblWeight**. Make the border style set to one to have a block line around the empty label. Place a label with the caption “Weight” to the left of the blank label and a label with the caption “lbs” to the right of the blank label. When the user presses the Calculate Weight command button, a value in pounds will display in the label’s caption.

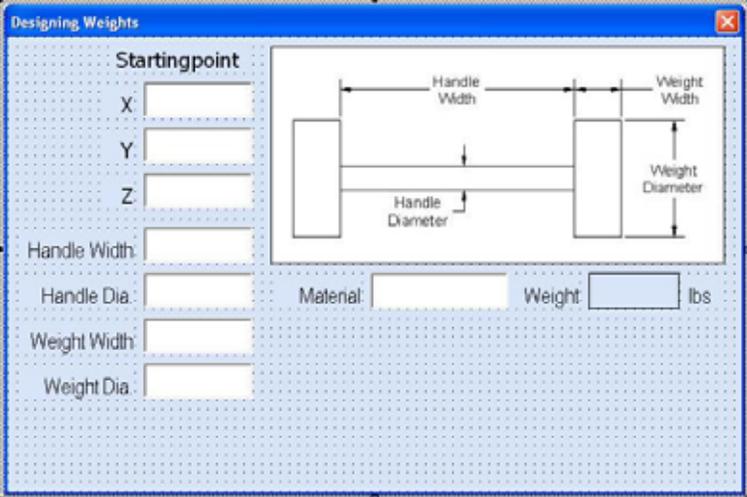


Figure 11.22– Adding a Label to Report the Weight

Inserting a Command Buttons into a Form

A command button is used so that a user will execute the application. Press the Command button on the Control Toolbar to add a command button. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the command button as shown in Figure 11.23.

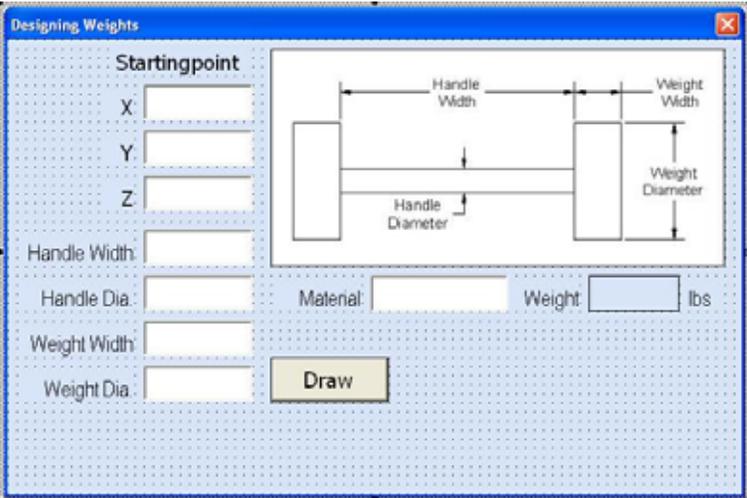


Figure 11.23 – Insert a Command Button onto a Form

We will name the command button using the name is **cmdDraw**.

Alphabetic	
(Name)	cmdDraw
Caption	Draw
Font	Tahoma
Height	30
Width	72

The font we want for the Command Button is 18 point, Tahoma. When highlighting the row for Font, a small command button with three small dots appears to the right of the font name of Arial. Click on the three dotted button to open the Visual Basic Font window. Make the changes as we did before and press OK to save the property.

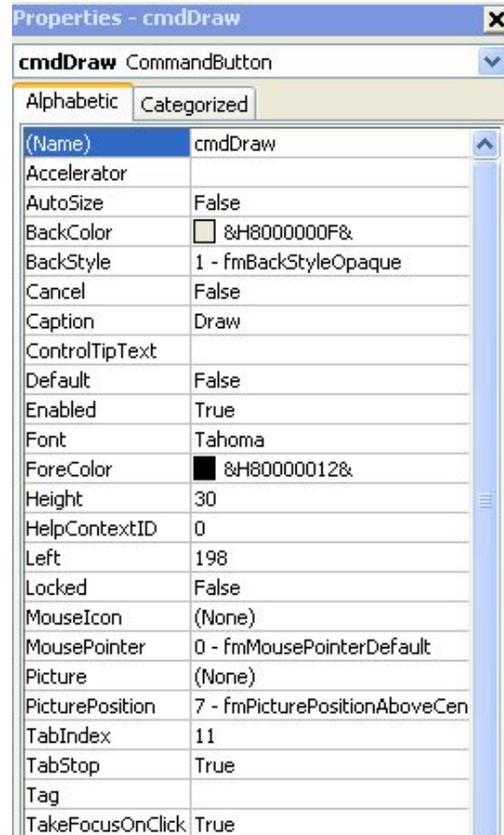


Figure 11.24 – Changing the (Name) to cmdDraw

Add a second Command button; named cmdClear is for clearing the Starting point's X, Y, Z coordinates, the handle width, the handle diameter, the weight width, and the weight diameter textboxes. The third command button is to exit the program. When the user presses the Exit command button, the application closes and full control of the manual AutoCAD program returns to the user. Notice the equal spacing between the command buttons gives a visually friendly appearance.

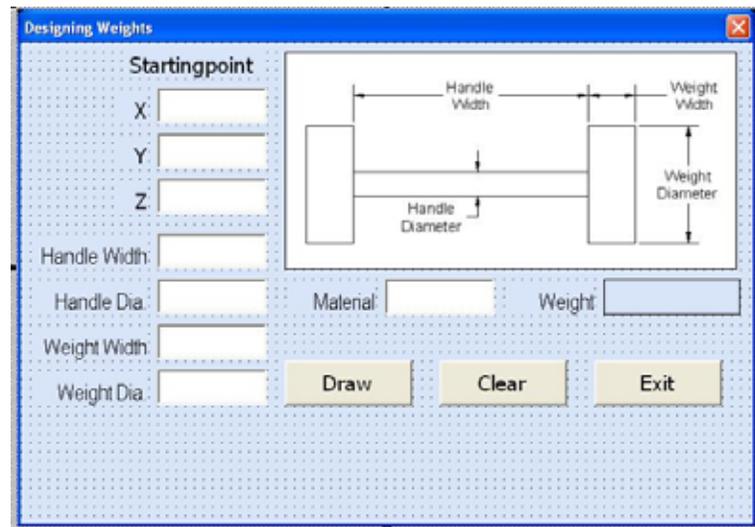


Figure 11.25 – Insert Two More Command Buttons

The fourth command button is the Calculate Weight button. We create a button that is long and underlines the material Listbox and the weight label. We will change the font to size 12 and the font style to bold as shown in figure 11.26.

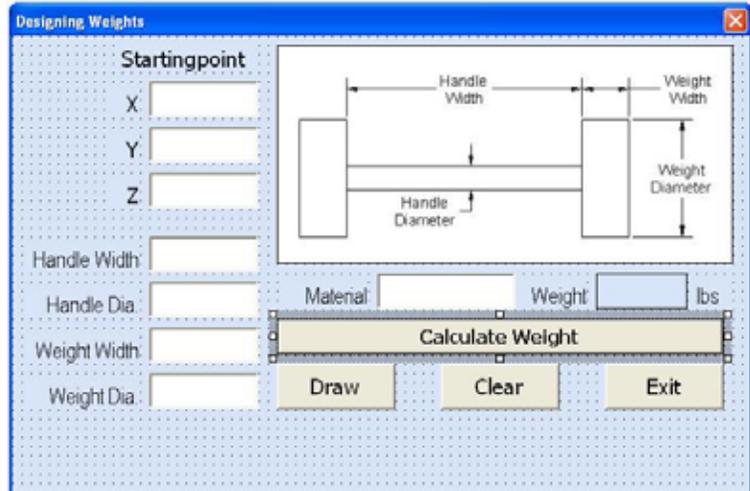


Figure 11.26 – Insert the Fourth Command Button

The fifth command button is Pick Point, which we will name cmdPickPoint. We draw the button as shown in Figure 11.27 and after typing Pick for the caption, press Shift Enter and type Point on the second line. Center the text. When we code for this command button, we will allow the user to select a point on the graphical display and the X, Y and Z coordinates will appear in their specific textboxes.

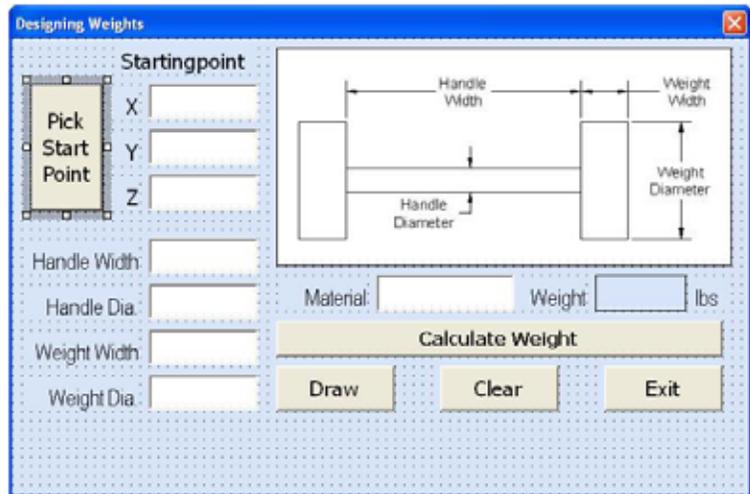


Figure 11.27 – Insert the Fourth Command Button

Adding a Copyright Statement to a Form

At the beginning of a new program, we will expect to see an explanation or any special instructions in the form of comments such as copyright, permissions or other legal notices to inform programmers what are the rules dealing with running the code. Comments at the opening of the code could help an individual determine whether the program is right for their application or is legal to use. The message box is a great tool when properly utilized to inform someone if they are breaking a copyright law when running the code.

Finish the form with the following copyright information.

Designing Weight - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.

If there are special rules or instructions that the user needs to know, place that information on the bottom of the form.

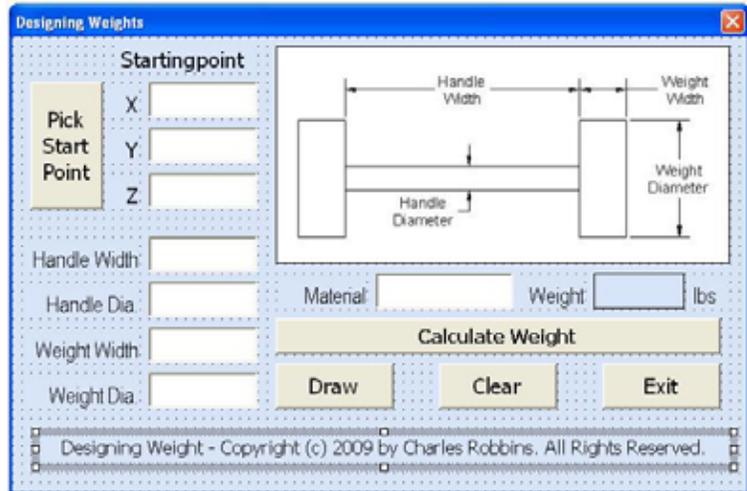


Figure 11.28 – Adding a Copyright Statement

Now that the form is complete, we will begin to write the code that actually interfaces the content of the form using logic and computations to draw the Designing Weights in the AutoCAD graphical display. We will begin the program with comments and place addition phrases throughout the program to assist ourselves or others in the future when modifying the code.

Adding Comments in Visual Basic to Communicate the Copyright

The comments we placed in the first three lines of the program will inform the individual opening and reading the code, but those user that may run the application without checking, the label on the bottom of the form with the copyright information is a great tool to alert the client to the rules of the program and what will the application do. If we require compensation then we would add that comment to the form, also.

So, to begin the actual coding of the program, double click on the Draw command button to enter the programming list. At the top of the program and before the line of code with **Sub DrawWeights ()**, place the following comments with the single quote (') character. Remember, the single quote character (') will precede a comment and when the code is compiled, comments are ignored.

Type the following line of code:

```
'Designing Weight - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.  
'This program will draw a set of 3D dumbbells in Modelspace
```

```
Sub DrawWeights ()
```

Declaring Variables in a Program with the Dimension Statement

When we are going to use a number, text string or object that may change throughout the life of the code, we create a variable to hold the value of that changing entity. In Visual Basic, the dimension or dim statement is one of the ways to declare a variable at the script or procedure level. The other two ways are the Private and Public statements, which we will use in later chapters.

Type the following lines of code after the comment.

'assign variables

```
Dim ObjCylinder1 As Acad3DSolid
Dim ObjCylinder2 As Acad3DSolid
Dim ObjCylinder3 As Acad3DSolid
Dim ObjDrawingObject As AcadEntity
Dim Startingpoint(0 To 2) As Double
Dim P1(0 To 2) As Double
Dim P2(0 To 2) As Double
Dim P3(0 To 2) As Double
Dim HandleWidth As Double
Dim HandleDiameter As Double
Dim WeightWidth As Double
Dim WeightDiameter As Double
```

In our program, we will declare a variable to enable us to draw three 3D solids, a variable for the drawing objects, a variable for each vertex and a variable for the handle width, the handle diameter, the weight width and the weight diameter.

The vertices are declared as double integers (As Double) with an array of zero to two (0 to 2). The vertex StartingPoint(0) represents the X coordinate, the StartingPoint(1) represents the Y coordinate and StartingPoint(2) represents the Z coordinate. This is the first program that we have written that will modify the Z coordinate.

We will declare points P1 through P3 for the vertices in the drawing in Figure 11.26. Lastly, we declare HandleWidth, HandleDiameter, WeightWidth and WeightDiameter as double integers (As Double).

When selecting variable names, they should be a word or a phrase without spaces that represents the value that the variable contains. If we want to hold a value of one's date of birth, we can call the variable, DateofBirth. The keywords Date and Birth are in sentence case with the first letter capitalized. There are no spaces in the name. Some programmers use the underscore character (_) to separate words in phrases. This is acceptable, but a double underscore (__) can cause errors if we do not detect the repeated character.

Assigning Values to the Variables

After we declare the variables and before we start drawing, we will assign the variables from the input the user types in the textboxes on the launched user form and then assign values to each of the vertices in the set of construction points.

Type the following code right below the declared variables.

'set variables

```
Startingpoint(0) = txtSpX.Text  
Startingpoint(1) = txtSpY.Text  
Startingpoint(2) = txtSpZ.Text  
HandleWidth = txtHandleWidth.Text  
HandleDiameter = txtHandleDiameter.Text  
WeightWidth = txtWeightWidth.Text  
WeightDiameter = txtWeightDiameter.Text
```

'point assignments and math

```
P1(0) = Startingpoint(0)  
P1(1) = Startingpoint(1)  
P1(2) = Startingpoint(2) - HandleWidth / 2 - WeightWidth / 2  
P2(0) = Startingpoint(0)  
P2(1) = Startingpoint(1)  
P2(2) = Startingpoint(2)  
P3(0) = Startingpoint(0)  
P3(1) = Startingpoint(1)  
P3(2) = Startingpoint(2) + HandleWidth / 2 + WeightWidth / 2
```

As we can see below, the first seven variables equal the values from the textbox. After that, we assign each point's X, Y and Z coordinates a number either from the variable or from a mathematical calculation that we arrive from the sketch in Figure 11.29. Point 2 is the same as the startingpoint and we use the variables `HandleWidth` and `WeightWidth` to measure the distance from one point to another.

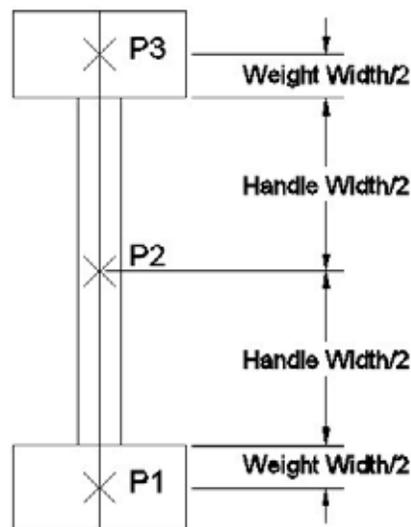


Figure 11.29 – Sketch of the Weight with Formulas

Inputting the Code to Draw 3D Objects in Visual Basic

Now we want to enter the code that will actually draw the 3D cylinders in the AutoCAD Model Space. We use the Set function to draw a line by typing **Set ObjCylinder1** and then we tell the computer that it will draw in Modelspace by adding a cylinder with the center point of P1, with the radius of **WeightDiameter / 2** and with a height of **WeightWidth**. The first cylinder is called **ObjCylinder1** and the second **ObjCylinder2** and the third **ObjCylinder3**. Then we need to union the 3D entities together, we do not need to make a selection set holding those solids.

Go ahead and type the following comments and drawing code:

'draw the first cylinders

```
Set ObjCylinder1 = ThisDrawing.ModelSpace.AddCylinder(P1, WeightDiameter / 2, WeightWidth)
ObjCylinder1.Update
```

'draw the second cylinders

```
Set ObjCylinder2 = ThisDrawing.ModelSpace.AddCylinder(P2, HandleDiameter / 2, HandleWidth)
ObjCylinder2.Update
```

'draw the third cylinders

```
Set ObjCylinder3 = ThisDrawing.ModelSpace.AddCylinder(P3, WeightDiameter / 2, WeightWidth)
ObjCylinder3.Update
```

Union Two Solids in Visual Basic

We can union two solids together with the **ObjCylinder1.Boolean acUnion** expression and by adding the second cylinder **ObjCylinder2**. The new combined solid is called **ObjCylinder1**. Then we can union the cylinder **ObjCylinder1** to **ObjCylinder3** with the same code except changing the cylinder to **ObjCylinder3**.

The comment and code appears as follows:

'union the cylinders

```
ObjCylinder1.Boolean acUnion, ObjCylinder2
ObjCylinder1.Boolean acUnion, ObjCylinder3
```

To end this Visual Basic subroutine, we will type a comment saying so. In the future, this will be more elaborate, but for now we will just get used to announcing the natural divisions of the script.

Type the following code:

End the program

End Sub

Populating the Material Listbox

To populate the Material Listbox with the aluminum and steel choices, we use the `.additem` function in the `With lstMaterial` expression. After adding the items to the list, end the `with` statement. Type the code as shown below.

```
Private Sub UserForm_activate()  
With lstMaterial  
    .AddItem "Aluminum"  
    .AddItem "Steel"  
End With
```

Resetting the Data with the cmdClear Command Button

To clear the textboxes containing the user input, we will first set the textbox for `txtSpX.Text` property to a "0" entry by using the equal sign "=" . This makes the property equal zero as a default. We do this also for the Y and Z coordinates. We will set the textboxes for `txtHandleWidth.Text`, `txtHandleDiameter.Text`, `txtWeightWidth.Text`, `txtWeightDiameter.Text` and `lblWeight.Caption` property to a blank entry by using the equal sign "=" and the null string "", and this will make that property blank. Notice that after the control object name the dot (.) separates the suffix which is the name of the property for that object.

Key the following code as a new subroutine `Private Sub cmdClear_Click()`.

```
Private Sub cmdClear_Click()  
'clear the form  
    txtSpX.Text = "0"  
    txtSpY.Text = "0"  
    txtSpZ.Text = "0"  
    txtHandleWidth.Text = ""  
    txtHandleDiameter.Text = ""  
    txtWeightWidth.Text = ""  
    txtWeightDiameter.Text = ""  
    lblWeight.Caption = ""  
End Sub
```

Exiting the Program with the cmdExit Command Button

To exit this program, we will unload the application and end the program. Type the following code:

```
Private Sub cmdExit_Click()  
'unload and end program  
    Unload Me  
    End  
End Sub
```

Executing a Subroutine with the cmdPickPoint Command Button

For the cmdPickPoint button, we will write SelectPoint which is a subroutine to allow the user to select a point on the graphical display and the X, Y and Z coordinates will be placed in the appropriate textboxes. A user can now type a starting point manually or has the choice to pick the initial point with their mouse.

```
Private Sub cmdStartpoint_Click()  
    SelectPoint  
End Sub
```

The SelectPoint subroutine starts with hiding the Foundation window and the prompt "Pick a Start Point: " is written on the command line. Once the single point is selected, the txtSpX textbox is given the first value of the starting point. After all three txtSP textboxes are filled, the Foundation window reappears.

```
Sub SelectPoint()  
Me.Hide  
Dim StartPoint As Variant  
    StartPoint = ThisDrawing.Utility.GetPoint(, vbCr & "Pick a Start Point: ")  
    txtSpX = StartPoint(0)  
    txtSpY = StartPoint(1)  
    txtSpZ = StartPoint(2)  
Me.Show  
End Sub
```

Executing a Subroutine with the cmdDraw Command Button

In this program, we use a subroutine which is executed by the Draw command button, so type the following code to execute the subroutine, DrawWeights.

```

Private Sub cmdDraw_Click()
'draw the dumbbell in 3d
  DrawWeights
End Sub

```

below is the entire program for creating the Designing Weights. Notice, we will write a small subroutine to calculate the weight of the dumbbell. We should have enough experience to declare and set variables, and write a formula that takes the volume of 3 cylinders and multiplies the answer by the density of the metal. Steel has a density of 0.283 pounds per cubic inch and aluminum has a density of 0.1 pounds per cubic inch.

```

'Designing Weight - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.
'This program will draw a set of 3D dumbbells in modelspace

Private Sub cmdCalculateWeight_Click()
'Assign variables
  Dim Density As Double
  Dim Answer As Double
  Dim WD As Double
  Dim WW As Double
  Dim HD As Double
  Dim HW As Double

'Set variables
  WD = txtWeightDiameter.Text
  WW = txtWeightWidth.Text
  HD = txtHandleDiameter.Text
  HW = txtHandleWidth.Text
  If IstMaterial.Text = "Aluminum" Then Density = 0.1
  If IstMaterial.Text = "Steel" Then Density = 0.283
'Computer the answer and list in the weight label caption
'Round the answer to one a max of one decimal
  Answer = Round((2 * (3.14159 * (WD / 2) ^ 2 * WW) + (3.14159 * (HD / 2) ^ 2 * HW)) * Density, 1)
  lblWeight.Caption = Answer
End Sub

Private Sub UserForm_activate()
With IstMaterial
  .AddItem "Aluminum"
  .AddItem "Steel"
End With
End Sub

Private Sub cmdDraw_Click()
'draw the top of the chip
  DrawWeights
End Sub

```

```
Sub DrawWeights()
```

```
'assign variables
```

```
Dim ObjCylinder1 As Acad3DSolid  
Dim ObjCylinder2 As Acad3DSolid  
Dim ObjCylinder3 As Acad3DSolid  
Dim ObjDrawingObject As AcadEntity  
Dim Startingpoint(0 To 2) As Double  
Dim P1(0 To 2) As Double  
Dim P2(0 To 2) As Double  
Dim P3(0 To 2) As Double  
Dim HandleWidth As Double  
Dim HandleDiameter As Double  
Dim WeightWidth As Double  
Dim WeightDiameter As Double
```

```
'set variables
```

```
Startingpoint(0) = txtSpX.Text  
Startingpoint(1) = txtSpY.Text  
Startingpoint(2) = txtSpZ.Text  
HandleWidth = txtHandleWidth.Text  
HandleDiameter = txtHandleDiameter.Text  
WeightWidth = txtWeightWidth.Text  
WeightDiameter = txtWeightDiameter.Text
```

```
'point assignments and math
```

```
P1(0) = Startingpoint(0)  
P1(1) = Startingpoint(1)  
P1(2) = Startingpoint(2) - HandleWidth / 2 - WeightWidth / 2  
P2(0) = Startingpoint(0)  
P2(1) = Startingpoint(1)  
P2(2) = Startingpoint(2)  
P3(0) = Startingpoint(0)  
P3(1) = Startingpoint(1)  
P3(2) = Startingpoint(2) + HandleWidth / 2 + WeightWidth / 2
```

```
'draw the first cylinders
```

```
Set ObjCylinder1 = ThisDrawing.ModelSpace.AddCylinder(P1, WeightDiameter / 2, WeightWidth)  
ObjCylinder1.Update
```

```
'draw the second cylinders
```

```
Set ObjCylinder2 = ThisDrawing.ModelSpace.AddCylinder(P2, HandleDiameter / 2, HandleWidth)  
ObjCylinder2.Update
```

'draw the third cylinders

```
Set ObjCylinder3 = ThisDrawing.ModelSpace.AddCylinder(P3, WeightDiameter / 2, WeightWidth)
ObjCylinder3.Update
```

'union the cylinders

```
ObjCylinder1.Boolean acUnion, ObjCylinder2
ObjCylinder1.Boolean acUnion, ObjCylinder3
```

'End the program

End Sub

Private Sub cmdStartpoint_Click()

SelectPoint

End Sub

Sub SelectPoint()

Me.Hide

Dim StartPoint As Variant

StartPoint = ThisDrawing.Utility.GetPoint(, vbCr & "Pick a Start Point: ")

txtSpX = StartPoint(0)

txtSpY = StartPoint(1)

txtSpZ = StartPoint(2)

Me.Show

End Sub

Private Sub cmdExit_Click()

'unload and end program

Unload Me

End

End Sub

Private Sub cmdClear_Click()

'clear the form

txtSpX.Text = ""

txtSpY.Text = ""

txtSpZ.Text = ""

txtHandleWidth.Text = ""

txtHandleDiameter.Text = ""

txtWeightWidth.Text = ""

txtWeightDiameter.Text = ""

lblWeight.Caption = ""

End Sub

Next, we will insert a module to launch the form.

Inserting a Module into a Visual Basic Application

Insert a Module by selecting Insert on the Menu Bar and select Module as shown in Figure 11.30. In the Project Menu, double click on the Module and type the following code.

```
Sub DrawICChip ()
'draw the weight
frmDesigningWeights.Show
End Sub
```

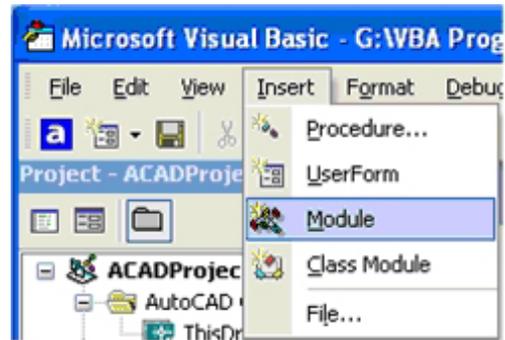


Figure 11.30 – Inserting a Module

The line of code, frmICchip.Show will display the form at the beginning of the program.

Running the Program

After noting that the program is saved, press the F5 to run the Designing Weights application. The Designing Weights window will appear on the graphical display in AutoCAD as shown in Figure 11.31.

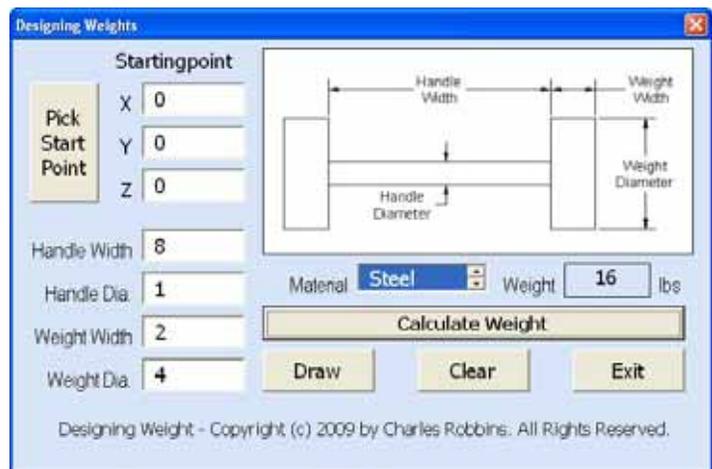


Figure 11.31 – Launching the Program

Type the following data or something similar into the textboxes and list box and select the Calculate the Weight and Draw Command buttons to execute the program.

X	0
Y	0
Z	0
Handle Width	8
Handle Dia.	1
Weight Width	2
Weight Dia.	4
Material	Steel

Figure 11.32 – Input Data

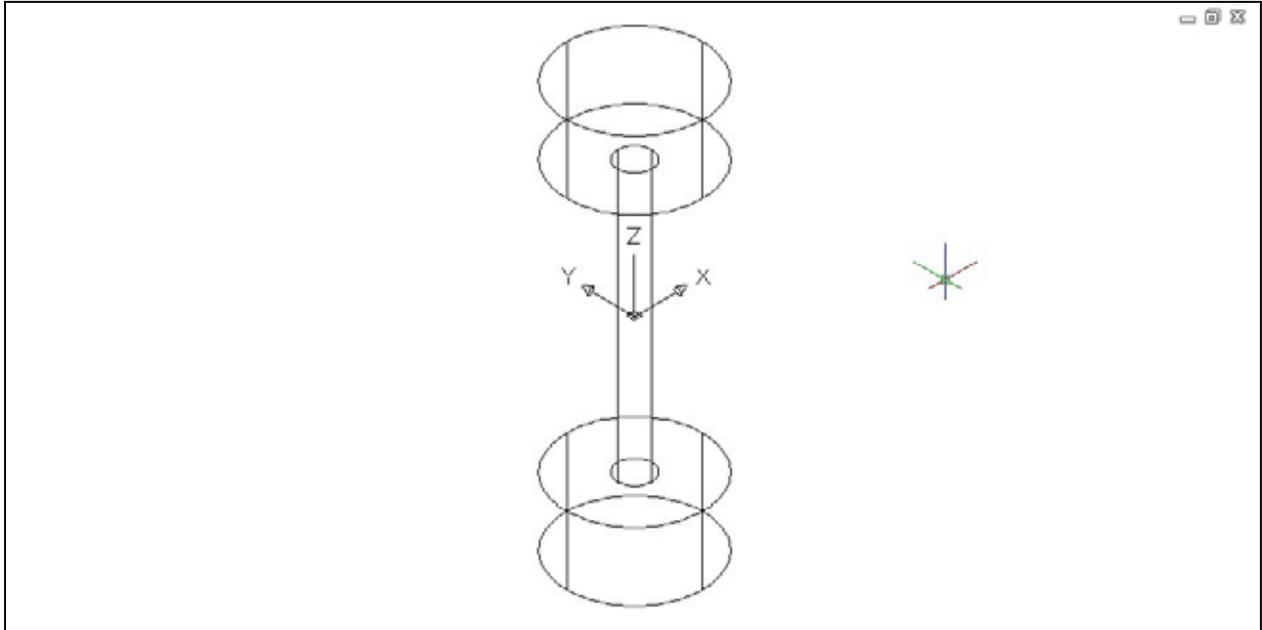


Figure 11.33 – The Finished Draw

There are many variations of this Visual Basic Application we can practice and draw many single view orthographic drawings. While we are practicing with forms, we can learn how to use variables, make point assignments and draw just about anything we desire. These are skills that we want to commit to memory.

*** World Class CAD Challenge 5-13 * - Write a Visual Basic Application that draws a dumbbell in 3D by a inputting data into a form. Complete the program in less than 120 minutes to maintain your World Class ranking.**

Continue this drill four times making other shapes with 3D primitives, each time completing the Visual Basic Application in less than 120 minutes to maintain your World Class ranking.

