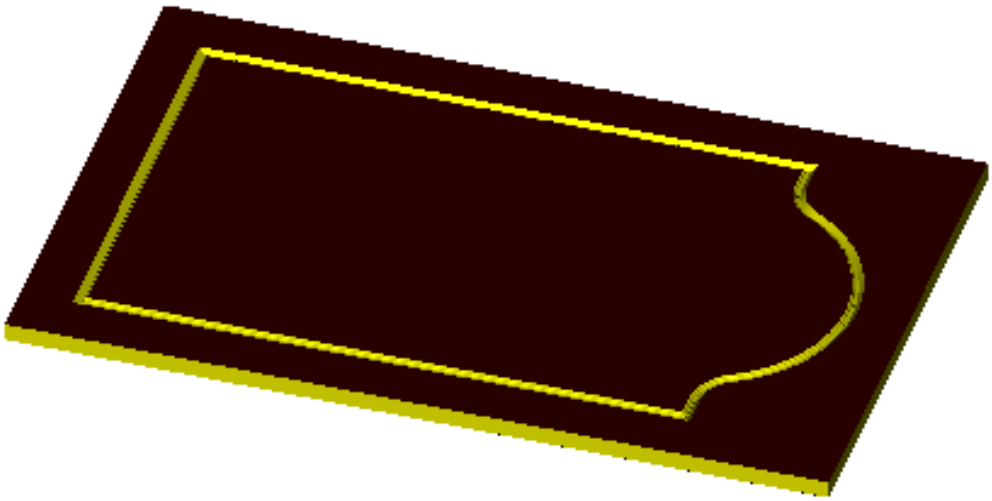
AlphaCAM

VBA

Router Example

The Project

In this project we are going to draw a door front with an arched panel from information supplied by the user and then machine the door complete.



Creating the New Project

Open a new VBA Project by selecting **New VBA Project** from the **VBA Macros** option in the **Utils** pull down menu. This will open the VBA editor and create a new empty project. Double click on the name property in the Properties Window and rename the project to ***CathedralDoor***. From the **File** pull down menu select **Save** and save the project as ***CathedralDoor.arb*** in the following folder ***C:\licomdir\VBMacros\StartUp\VBA Training*.**

Creating the Form

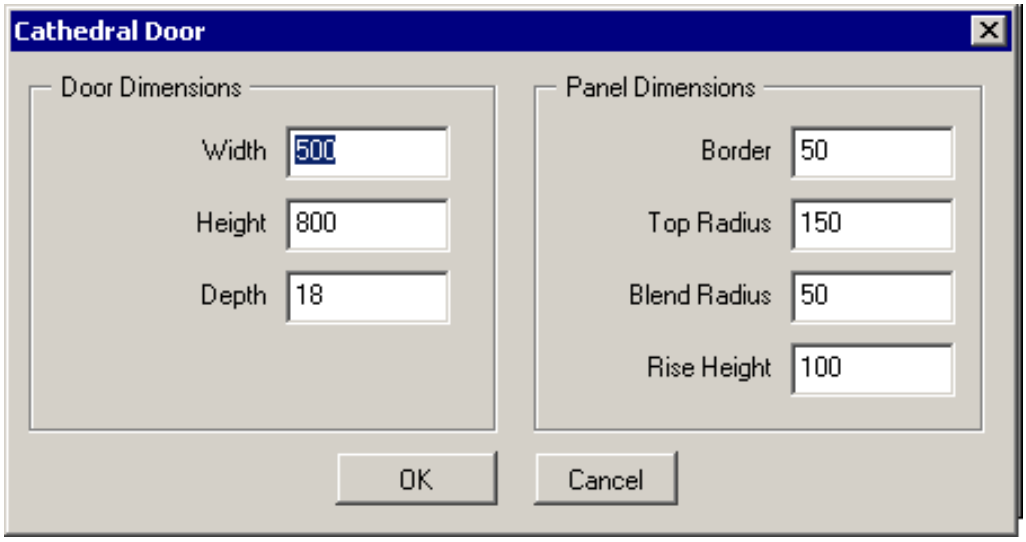
Insert a new form into your project by selecting the **Insert** pull down menu and clicking on **UserForm**.

The new form will need to contain the following controls

- 2 Frames
- 7 Text Boxes
- 7 Labels
- 2 Command Buttons

New controls are added to the form by single clicking on the desired control in the **Toolbox** and then single clicking on the **Userform**. If your **Toolbox** is not visible it can be made visible by selecting **Toolbox** from the **View** pull down menu.

The form you are going to create will look like this



Set the caption property for each control as shown in the image above.

To make these new controls easy to identify it is best if they are given names that relate to what they are, and what you want to use them for. To set the name property for each control you single click on the control you wish to name in the **UserForm Window** and then double click on the name property in the **Properties Window**. This will allow you easy access to each control at any time from within the project.

For this project we are going to use the name properties for the controls as follows

- Userform ~ frmMain
- Frame1 ~ fraDoor
- Label1 ~ lblWidth
- Textbox1 ~ txtWidth
- Label2 ~ lblHeight
- Textbox2 ~ txtHeight
- Label3 ~ lblDepth
- Textbox3 ~ txtDepth
- Frame2 ~ fraPanel
- Label4 ~ lblBorder
- Textbox4 ~ txtBorder
- Label5 ~ lblTopRadius
- Textbox5 ~ txtTopRadius
- Label6 ~ lblBlendRadius
- Textbox6 ~ txtBlendRadius
- Label7 ~ lblRiseHeight
- Textbox7 ~ txtRiseHeight
- Command1 ~ cmdOK
- Command2 ~ cmdCancel

It is quite likely that you will not have inserted the controls onto the form into a suitable order for moving around the form using the tab key on the keyboard. The order can be changed by using the following procedure.

Set the Tab Order for the form ***frmMain*** by right clicking on the background of the form and selecting **Tab Order** from the **pop up window**.

fraDoor
fraPanel
cmdOK
cmdCancel

Set the Tab Order for the frame ***fraDoor*** by right clicking on the background of the frame and selecting **Tab Order** from the **pop up window**.

lblWidth
txtWidth
lblHeight
txtHeight
lblDepth
txtDepth

Set the Tab Order for the frame ***fraPanel*** by right clicking on the background of the frame and selecting **Tab Order** from the **pop up window**.

lblBorder
txtBorder
lblTopRadius
txtTopRadius
lblBlendRadius
txtBlendRadius
lblRiseHeight
txtRiseHeight

Writing the Code

The first thing we need to do is create a new menu so that we can run our project. To do this we need to insert a new ***Module*** into the project. This is done by selecting the ***Insert*** pull down menu and clicking on ***Module***. The module we are creating is special module because AlphaCAM needs to access it on initialisation to add a new menu. For this to happen the module has to have the name ***Events***. To rename the module double click on the name property in the properties window and type in the word Events. In the code window for the *Events* module we are going to add two new functions, one to add the new menu and one to show the ***Userform*** when the new menu item is selected. The code is as follows ~

```
Public Function InitAlphacamAddIn(acamversion As Long) As Integer
    Dim fr As Frame
    Set fr = App.Frame
    With fr
        ' set up strItemName  and strMenuName as new string variables
        Dim strItemName As String, strMenuName As String
        strItemName = "Cathedral Door": strMenuName = "VBA Training Macro"
        ' create the new menu
        .AddMenuItem2 strItemName, "ShowfrmMain", acamMenuNEW, strMenuName
    End With
    InitAlphacamAddIn = 0
End Function


Function ShowFrmMain()
    ' show the main form
    Load frmMain
    frmMain.Show
End Function
```

To see the new menu you will need to save the project, then close and restart **AlphaCAM**. This will allow **AlphaCAM** to read your new ***Module*** and add the new menu.

Test the new command by selecting ***Cathedral Door*** from the ***VBA Training Macro*** pulldown menu.

At the moment there is no code associated with the form so the only way to exit the form is to select the **'X'** in the top right corner of the form.

To continue editing the project we will need to reopen the project. Do this by selecting **Open VBA Project** from **VB Macros** option in the **Utils** pulldown menu.

Each control on the form and the form itself can have its own unique piece of code. The easiest way to access the code for each control is by double clicking on the desired control in the ***Userform Window***. This will open a new window called the ***Code Window*** and insert an empty subroutine based on the control's default event.

We are going to set some default values for the form by using the form's ***Activate*** event.

Double click on the background of the form in the ***Userform Window***.

This will open the ***Code Window*** with the form's default event, which is the ***Click Event*** as shown below.

```
Private Sub UserForm_Click()

End Sub
```

To change this to the ***Activate*** event there is a drop down box at the top right corner of the ***Code Window*** listing all the events available for the currently active control. Click on this and position your mouse over the word ***Activate*** and then click again. This will create the following new code for you.

```
Private Sub UserForm_Activate ()

End Sub
```

Modify the ***UserForm_Activate*** sub so that it looks like the following code

```
Private Sub UserForm_ Activate ()

    If Len(txtWidth) = 0 Then
        ' set defaults for door
        txtWidth = 500
        txtHeight = 800
        txtDepth = 18

         ' set defaults for panel
        txtBorder = 50
        txtTopRadius = 175
        txtBlendRadius = 50
        txtRiseHeight = 100
    End If

    ' set focus to first text box and highlight
    txtWidth.SetFocus
    txtWidth.SelStart = 0
    txtWidth.SelLength = 999

End Sub
```

Write the code for the ***Click Event*** for the command button ***cmdCancel*** so that it will end the project. The easiest way to create the new ***sub*** is to double click on the word ***frmMain*** in the **Project Explorer Window.** This will open the ***Userform*** so that you can double click on the ***Cancel*** button on the form. This will create the following new code for you.

```
Private Sub cmdCancel_Click()

End Sub
```

Modify this new sub so that it looks like the following code

```
Private Sub cmdCancel_Click()

   End      ' end VBA macro

End Sub
```

To enable the ***Esc*** key on the keyboard to act in the same way as the ***cmdCancel*** button you can set the ***Cancel*** property for the ***cmdCancel*** button to ***True***.

Save the project and switch to AlphaCAM so that you can test the new code by selecting ***Cathedral Door*** from the ***VBA Training Macro*** pulldown menu.

To write the main code to draw and machine the door panel it would be possible to put all the code in the click event for the ***cmdOK*** button. The main problem with doing this is that in a large project it would become difficult to follow and debug the code. It would also mean that you may have to write the same piece of code more than once because a similar feature is required. To overcome this problem it is best to create a new ***Module***, and write common functions into it. Calls to these functions can then be made from any point in the project.

Insert a new module into the project by selecting Module from the Insert Pulldown menu, and set its name property to ***Main***.

In this module we are going to write a new Subroutine to draw and machine the door. This can be achieved in two ways, either select ***Procedure*** from ***Insert*** menu or type the following code directly into the module.

```
Public Sub CreateCathedralDoor()

End Function
```

This function will need to know the dimensions for the door as the user has typed them into the form. To do so, we edit the name of the function to include the variables we want to pass to it. This is shown below.

```
Public Sub CreateCathedralDoor( _
                dblHeight As Double, dblWidth As Double, dblDepth As Double, _
                dblBorder As Double, dblRiseHeight As Double, _
                dblBlendRadius As Double, dblTopRadius As Double)

End Function
```

To make a call to this function and send the required values to it edit the click event for the ***cmdOk*** button to hide the form and make a call to the function. The easiest way to create the new ***sub*** is to double click on the word ***frmMain*** in the **Project Explorer Window.** This will open the userform so that you can double click on the *OK* button on the form. This will create the following new code for you.

```
Private Sub cmdOK_Click()

End Sub
```

Modify this new sub so that it looks like the following code

```
Private Sub cmdOK_Click()

    frmMain.Hide      ' hide the form
    DoEvents              ' stop the project from processing until it has completed all previous tasks

    ' call the subroutine to create the cathedral door
   CreateCathedralDoor CDbl(txtHeight), CDbl (txtWidth), CDbl (txtDepth), _
            CDbl (txtBorder), CDbl (txtRiseHeight), CDbl (txtBlendRadius), CDbl (txtTopRadius)


End Sub
```

The following features will need to be edited into the ***CreateCathedralDoor*** Subroutine.

1. Clear the memory
2. Define the active drawing
3. Create a work volume
4. Create a material
5. Draw the outside of the door
6. Machine the outside of the door
7. Draw the panel
8. Machine the panel

Double click on the word ***Main*** in the **Project Explorer Window** to see the code for the module ***Main***. Make the following edits to the ***CreateCathedralDoor*** Subroutine.

1. Clear the Memory

```
' clear the memory
App.new
```

2. Defining the Active Drawing

```
' define the active drawing
Dim drw as Drawing
Set drw = App.ActiveDrawing
```

3. Creating the work volume

```
' create the work volume
Dim WorkVol As Path
Set WorkVol = drw.CreateRectangle(0, 0, dblHeight, dblWidth)
WorkVol.SetWorkVolume 0, -dblDepth
```

4. Creating the material

```
' create the material
Dim Material As Path
Set Material = drw.CreateRectangle(-1, -1, dblHeight + 1, dblWidth + 1)
Material.SetMaterial 0, -dblDepth
```

5. Creating the outside of the door using fast geometry

```
' create the outside of the door
Dim tempFastGeo As FastGeometry
Dim DoorGeo As Path
Set tempFastGeo = Drw.CreateFastGeometry
With tempFastGeo
   .Point 0, dblWidth / 2
   .Point 0, dblWidth
   .Point dblHeight, dblWidth
   .Point dblHeight, 0
   .Point 0, 0
   .Point 0, dblWidth / 2
   Set DoorGeo = .Finish
End With
DoorGeo.ToolSide = acamLEFT
```

Switch to AlphaCAM and test the project by running it from the pulldown menu.

5. Machining the outside of the door ~ for this we are going to add a new module called ***MachiningRoutines*** and make calls to functions in the new module.

Add the following code to the ***MachiningRoutines*** module

```
' function to select router Tool
Public Function GetRouterTool(ToolName As String) As MillTool
    ' define local variables
    Dim Tool As MillTool
    ' clear the Tool object
    Set Tool = Nothing
    Do
        ' set up error trapping
        On Error Resume Next
        ' select the Tool
        Set Tool = App.SelectTool(ToolName)
        ' if tool not found show normal tool select dialog box
        If Tool Is Nothing Then
            ' ask user to select tool
            Set Tool = App.SelectTool("$User")
        Else
            ' do nothing as the tool was found
        End If
        ' cancel error trapping
        On Error GoTo 0
        ' if the tool was not found and the user has
        ' not selected a tool loop back to the start
        ' to force the user to select a tool
    Loop While Tool Is Nothing
    ' set the return tool for the function
    Set GetRouterTool = Tool
End Function
```

Adding the following code to the ***CreateCathedralDoor*** Subroutine in the module ***Main*** will call the ***GetRouterTool*** function and select the specified tool.

```
' machine outside of door
' select the tool
Dim Tool As MillTool
Dim strToolName As String
strToolName = App.Frame.PathOfThisAddin & "\rtools.alp\Router-20mm.art"
Set Tool = Nothing
Set Tool = GetRouterTool(strToolName)
```

Add the following code to the module ***MachiningRoutines***.

```
' public function to generate Rough/Finish toolpaths
Public Function CreateRoughFinishPaths( _
                    GeosToMachine As Paths, _
                    Optional dblSafeRapid As Double = 0, _
                    Optional dblRapidDownto As Double = 0, _
                    Optional dblMaterialTop As Double = 0, _
                    Optional dblFinalDepth As Double = 0, _
                    Optional dblStock As Double = 0, _
                    Optional intMcComp As Integer = acamCompTOOLCEN, _
                    Optional intXyCorners As Integer = acamCornersROUND, _
                    Optional intCoolant As Integer = acamCoolNONE) As Paths
    ' define local variables
    Dim Drw As Drawing
    Dim Md As MillData
    ' setup local variables
    Set Drw = ActiveDrawing
    Set Md = App.CreateMillData
    ' setup the milling data
    With Md
        .SafeRapidLevel = dblSafeRapid
        .RapidDownTo = dblRapidDownto
        .MaterialTop = dblMaterialTop
        .FinalDepth = dblFinalDepth
        .Stock = dblStock
        .McComp = intMcComp
        .XYCorners = intXyCorners
        .Coolant = intCoolant
        ' select the geometries to be machined
        GeosToMachine.Selected = True
        ' create the toolpaths
        Set CreateRoughFinishPaths = .RoughFinish
    End With
End Function
```

Adding the following code to the ***CreateCathedralDoor*** subroutine in the module ***Main*** will call the ***CreateRoughFinishPaths*** function and machine the outside edge of the door with the previously selected tool.

```
' create machining data for outside of door
Dim GeosToMachine As Paths
Dim Tps As Paths, Tp As Path
Set GeosToMachine = Nothing
Set GeosToMachine = Drw.CreatePathCollection
GeosToMachine.Add DoorGeo
Set Tps = Nothing
Set Tps = CreateRoughFinishPaths( _
            GeosToMachine, _
            dblSafeRapid:=50, _
            dblRapidDownto:=5, _
            dblMaterialTop:=0, _
            dblFinalDepth:=-dblDepth - 1, _
            intMcComp:=acamCompMC)

' apply lead in and lead out
For Each Tp In Tps
    Tp.SetLeadInOutAuto acamLeadBOTH, acamLeadBOTH, 1.2, 1.2, 45, False, False, 0
Next Tp
```

Switch to AlphaCAM and test the project by running it from the pulldown menu.

6. Creating the panel

Adding the following code to the ***CreateCathedralDoor*** subroutine in the module ***Main*** to create 2 profiles for the panel.

```
' create the panel using fast geometry
Dim PanelGeo1 As Path, PanelGeo2 As Path
Dim dblPanelxStart As Double, dblPanelyStart As Double
Dim dblPanelxFin As Double, dblPanelyFin As Double
Dim tempPaths As Paths
dblPanelxStart = dblBorder
dblPanelyStart = dblBorder
dblPanelxFin = dblHeight - dblBorder – dblRiseHeight
dblPanelyFin = dblWidth – dblBorder
Set tempFastGeo = Drw.CreateFastGeometry
With tempFastGeo
    .Point dblPanelxStart, (dblWidth / 2) + 10
    .Point dblPanelxStart, dblPanelyStart
    .Point dblPanelxFin, dblPanelyStart
    .LineToArc dblBlendRadius, True, False, 90
    .KnownArc dblTopRadius, False, dblHeight - dblBorder - dblTopRadius, dblWidth / 2
    .ArcToLine dblBlendRadius, True, False, 90
    .Point dblPanelxFin, dblPanelyFin
    .Point dblPanelxStart, dblPanelyFin
    .Point dblPanelxStart, (dblWidth / 2) - 10
    Set PanelGeo2 = .Finish
End With
Set tempPaths = PanelGeo2.Offset(6, acamLEFT)
Set PanelGeo1 = tempPaths.Item(1)
PanelGeo2.ToolSide = acamLEFT
```

7. Machining the panel

Adding the following code to the ***CreateCathedralDoor*** subroutine in the module ***Main*** will call the ***GetRouterTool*** function and select the specified tool.

```
' select the tool
strToolName = App.Frame.PathOfThisAddin & "\rtools.alp\Profile Tool 1.art"
Set Tool = Nothing
Set Tool = GetRouterTool(strToolName)
```

Adding the following code to the ***CreateCathedralDoor*** subroutine in the module ***Main*** will call the ***CreateRoughFinishPaths*** function and machine the panel of the door with the previously selected tool.

```
' create the toolpaths
Set GeosToMachine = Nothing
Set GeosToMachine = Drw.CreatePathCollection
GeosToMachine.Add PanelGeo1
Set Tps = Nothing
Set Tps = CreateRoughFinishPaths( _
            GeosToMachine, _
            dblSafeRapid:=50, _
            dblRapidDownto:=5, _
            dblMaterialTop:=0, _
            dblFinalDepth:=-5)
```

Adding the following code to the ***CreateCathedralDoor*** subroutine in the module ***Main*** will apply a manual sloping lead in out to the toolpaths created by ***CreateRoughFinishPaths.***

```
' apply leadin / out to the profile paths
Dim Efirst As Element, Elast As Element
Dim dblXs As Double, dblYs As Double
Dim dblXf As Double, dblYf As Double
For Each Tp In Tps
   Set Efirst = Tp.GetFirstElem
   If Efirst.IsRapid Then Set Efirst = Efirst.GetNext
   Set Elast = Tp.GetLastElem
   dblXs = Efirst.StartXG
   dblYs = Efirst.StartYG
   dblXf = Elast.EndXG
   dblYf = Elast.EndYG
   Tp.SetLeadInOutManual acamLeadLINE, acamLeadLINE, True, True, _
      dblXs, dblYs + 20, dblXf, dblYf - 20
Next Tp
```

The outer profile of the panel needs to machined using the 3D engraving process, so we will add a new function to the ***MachiningRoutines*** and call it from the ***CreateCathedralDoor*** subroutine.

Add the following code to the module ***MachiningRoutines*** to define the new function.

```
' public function to generate 3D Engrave toolpaths
Public Function Create3dEngravePaths( _
                    GeosToMachine As Paths, _
                    Optional dblSafeRapid As Double = 0, _
                    Optional dblRapidDownto As Double = 0, _
                    Optional dblMaterialTop As Double = 0, _
                    Optional dblFinalDepth As Double = 0, _
                    Optional dblStock As Double = 0, _
                    Optional intXyCorners As Integer = acamCornersROUND, _
                    Optional intCoolant As Integer = acamCoolNONE, _
                    Optional dblEngraveCornerAngleLimit As Double = 180, _
                    Optional dblChordError As Double = 0.05, _
                    Optional dblStepLength As Double = 0.1) As Paths
    ' define local variables
    Dim Drw As Drawing
    Dim Md As MillData
    ' set local variables
    Set Drw = ActiveDrawing
    Set Md = App.CreateMillData
    ' setup milling data
    With Md
        .SafeRapidLevel = dblSafeRapid
        .RapidDownTo = dblRapidDownto
        .MaterialTop = dblMaterialTop
        .FinalDepth = dblFinalDepth
        .StepLength = dblStepLength
        .ChordError = dblChordError
        .EngraveType = acamEngraveGEOMETRIES
        .EngraveCornerAngleLimit = dblEngraveCornerAngleLimit
        ' select the geometries
        GeosToMachine.Selected = True
        ' create the toolpaths
        Set Create3dEngravePaths = .Engrave
    End With
End Function
```

Adding the following code to the ***CreateCathedralDoor*** subroutine in the module ***Main*** will call the ***GetRouterTool*** function and select the specified tool.

```
' select the tool
strToolName = App.Frame.PathOfThisAddin & "\rtools.alp\Profile Tool 2.art"
Set Tool = Nothing
Set Tool = GetRouterTool(strToolName)
```

Adding the following code to the ***CreateCathedralDoor*** subroutine in the module ***Main*** will call the ***Create3dEngravePaths*** function and 3D engrave the panel of the door with the previously selected tool.

```
' create the toolpaths
Set GeosToMachine = Nothing
Set GeosToMachine = Drw.CreatePathCollection
GeosToMachine.Add PanelGeo2
Set Tps = Nothing
Set Tps = Create3dEngravePaths( _
            GeosToMachine, _
            dblSafeRapid:=50, _
            dblRapidDownto:=5, _
            dblMaterialTop:=0, _
            dblFinalDepth:=-5)
```

Adding the following code to the ***CreateCathedralDoor*** subroutine in the module ***Main*** will apply a manual sloping lead in out to the toolpaths created by ***Create3dEngravePaths.***

```
' apply leadin / out to the profile paths
For Each Tp In Tps
   Set Efirst = Tp.GetFirstElem
   If Efirst.IsRapid Then Set Efirst = Efirst.GetNext
   Set Elast = Tp.GetLastElem
   dblXs = Efirst.StartXG
   dblYs = Efirst.StartYG
   dblXf = Elast.EndXG
   dblYf = Elast.EndYG
   Tp.SetLeadInOutManual acamLeadLINE, acamLeadLINE, True, True, _
      dblXs, dblYs + 20, dblXf, dblYf - 20
Next Tp
```

Switch to AlphaCAM and test the project by running it from the pulldown menu.

Add a new function to the module ***Main*** to check if the active drawing has any geometries in it. If it has, show a warning to allow the user to save any unsaved data if they want to.

```
Function FileNew()
    ' function to test if active drawing has any geometries
    ' and show a warning that any unsaved data will be lost
    Dim MsgText As String
    MsgText = "This will open a new drawing, press OK to continue"
    Dim MsgBoxReturn As Integer
    If App.ActiveDrawing.GetGeoCount > 0 Then
        MsgBoxReturn = MsgBox(MsgText, vbOKCancel)
        If MsgBoxReturn = vbOK Then
            App.New
        Else
          End     ' exit VBA macro
        End If
    End If
End Function
```

Modify the sub *ShowFrmMain* in the module ***Events*** to include a call to the new function.

```
Sub ShowFrmMain ()
    ' run function to test if the active drawing has any geometries
    FileNew
  '  show main dialog box
   Load frmMain
   frmMain.Show
End Sub
```

Add a new function to the module **Main** to refresh the screen.

```
Function Refresh()
    With App.ActiveDrawing
        .ThreeDViews = True
        .Options.ShowRapids = False
        .Options.ShowTools = False
        .Redraw
    End With
End Function
```

Modify the sub *ShowFrmMain* in the module **Events** to include a call to the new function.

```
Sub ShowFrmMain()
    ' run function to test if the active drawing has any geometries
    NewDrawing
    '  show main dialog box
    Load frmMain
    frmMain.Show
     ' run function to refresh the screen
    Refresh
End Sub
```