**Lab 4 – PaletteSet and Database Events**

In this lab, we will add a PaletteSet and add database events. In the events we will add data about entities added to the drawing to a treeview control in a paletteSet.

Open your Lab3 project and add a UserControl. Go to the Project menu and select “Add User Control”. (Or open the Lab4 project that already has the UserForm and the steps for lab).



Accept the default name (UserControl1.vb)

Display the Toolbox (Ctrl+Alt+X) and add a TreeView control to the UserControl.



Anchor all of the sides of the TreeView control using the properties window as seen in this screenshot:



Copy the steps below to your project. Place the steps after the End Sub of the “addAnEnt” procedure in Lab3. Work through the steps to create the PaletteSet and add the Database Events.

' 1. Add a Reference to PresentationCore. (Use the .NET tab on

' the Add Reference dialog. This is needed for the PaletteSet

' we will declare in step 3.

' 2. Use the Imports Statement for namespace Autodesk.AutoCAD.Windows

' 3. Declare a PaletteSet variable (global) as a PaletteSet. (It will

' only be created once). Add this declaration after AddAnEnt End Sub

' from Lab 3.

' 4. Declare a variable as UserControl1. This is the control created

' in the steps in the Lab4 document. This control will be housed

' by the PaletteSet created in step 3.

' 5. Add an new command named palette. Use the CommandMethod

' attribute and create the Sub that will run when the command is run

' in AutoCAD.

' Note: Put the End Sub after step 10.

' 6. Add an "if then" statement and check to see if the

' PaletteSet declared in step 3 is equal to nothing. It will be

' Nothing the first time the command is run.

' Note: Put the "End If" after step 9

' 7. The PaletteSet is nothing here so we create a a new PaletteSet

' with a unique GUID. Use the New keyword. Make the Name Parameter

' "My Palette". For the ToolID parameter generate a new GUID.

' On the Tools menu select "Create Guid". On the Create GUID

' Dialog select "Registry Format" Select New GUID and the copy.

' Paste the GUID to use as the New Guid. Replace the curley

' braces with double quotes. (The parameter for New Guid is a string)

' 8. Instantiate the UserControl1 variable created in

' step 4. Use the New keyword. (New UserControl1)

' This control houses the tree control.

' 9. Add the UserControl to the PaletteSet. Use the Add method

' of the PaletteSet instantiated in step 7. Use "Palette1" for the

' name parameter and the control instantiated in step 8 for the

' second parameter.

' 10. Display the paletteset by making the Visible property of the

' PaletteSet equal to true. The second time the command is run

' this is the only code in this procedure that will be processed.

' 11. Add a command named "addDBEvents. Use the CommandMethod attribute

' and add the procedure that will run when the commmand is run in AutoCAD

' Note: Put the End Sub after step 20

' 12. use an ""If Then" statement and see if the palette

' created in step 4 Is Nothing.

' Note: put the "End If" after step 15

' 13. Declare an intantiate a editor object. Use the Editor

' property of Application.DocumentManager.MdiActiveDocument

' 14. Use the WriteMessage method of the Editor variable

' created in step 13. Use this for the message parameter

' vbCr + "Please call the 'palette' command first"

' 15. Exit the Sub

' 16. Declare a Database variable and instantiate it by making it

' equal to the Database property of the

' Application.DocumentManager.MdiActiveDocument

' 17. Use the AddHandler statement to had a Database ObjectAppended

' event. Use the ObjectAppended event of the database variable

' created in step 16 for the first parameter (Event). For the second

' parameter (Delegate) use the New statement and create an ObjectEventHandler.

' Use the AddressOf statement and the name of a procedure (callback\_ObjectAppended)

' you will create in step 21.

' 18. Use the AddHandler statement to had a Database ObjectErased

' event. Use the ObjectErased event of the database variable

' created in step 16 for the first parameter (Event). For the second

' parameter (Delegate) use the New statement and create an ObjectErasedEventHandler.

' Use the AddressOf statement and the name of a procedure (callback\_ObjectErased)

' you will create in step 24.

' 19. Use the AddHandler statement to had a Database ObjectReappended

' event. Use the ObjectReappended event of the database variable

' created in step 16 for the first parameter (Event). For the second

' parameter (Delegate) use the New statement and create an ObjectEventHandler.

' Use the AddressOf statement and the name of a procedure (callback\_ObjectReappended)

' you will create in step 32.

' 20. Use the AddHandler statement to had a Database ObjectUnappended

' event. Use the ObjectUnappended event of the database variable

' created in step 16 for the first parameter (Event). For the second

' parameter (Delegate) use the New statement and create an ObjectEventHandler.

' Use the AddressOf statement and the name of a procedure (callback\_ObjectUnappended)

' you will create in step 35.

' 21. Create a Private Sub named callback\_ObjectAppended. This is the Sub that

' will be called when an Object is Appended to the Database. (The name needs to be

' the name used in the Delegate parameter of step 17). The first parameter is an

' object. (Use ByVal and sender as the name of the Object). The second parameter is

' an ObjectEventArgs. (Use ByVal and e as the name of the ObjectEventArgs)

' Note: Put the End Sub after step 23

' 22. Declare a TreeNode variable. (System.Windows.Forms.TreeNode).

' Note: You can save some typing by using Imports and importing the namespace

' Instantiate it using the Add method of the Nodes property of the TreeView on the

' UserForm() created in step 4. Use the ObjectEventArgs passed into the method for

' the string parameter and use the "Type" of DBObject. (e.DBObject.GetType().ToString())

' 23. Make the Tag property of the node created in step 22 equal to the ObjectId of

' the appended object. This will allow us to record it's ObjectId for recognition in

' other events. Use e.DBObject.ObjectId.ToString()

' 24. Create a Private Sub named callback\_ObjectErased. This is the Sub that

' will be called when an Object is erased from the Database. (The name needs to be

' the name used in the Delegate parameter of step 18). The first parameter is an

' object. (Use ByVal and sender as the name of the Object). The second parameter is

' an ObjectErasedEventArgs. (Use ByVal and e as the name of the ObjectErasedEventArgs)

' Note: Put the End Sub before step 32

' 25. use an "If Then Else" statement and check the Erased property of the

' ObjectErasedEventArgs passed into the function. (e.Erased)

' Note: Put the "Else" stament before step 30 and the "End If" after step 31

' 26. Here we will search for an object in the treeview control so it can be removed.

' Create a For Each statement. Use node for the element name and the type is

' Forms.Treenode. The group paramater is the Nodes in the TreeView.

' (myPalette.TreeView1.Nodes)

' Note: put the Next statement below step 29. (before the "Else" statement)"

' 27. Use an "If Then" statement. Test to see if the node Tag is the ObjectId

' of the erased Object. Use the DBObject property of the of the

' ObjectErasedEventArgs passed into the event. (e.DBObject.ObjectId.ToString)

' Note: put the "End If" above the "Next" statement added in step 23

' 28. Remove the node by calling the Remove method. (The entity was

' erased from the drawing).

' 29. Exit the For loop. (Exit For)

' 30. If this is processed it means that the object was unerased. (e.Erased was false)

' Declare a System.Windows.Forms.TreeNode use newNode as the name. Instantiate it by

' using the Add method of the Nodes collection of the TreeView created in previous steps.

' Use the Type of the object for the parameter.

' e.DBObject.GetType().ToString()

' 31. Make the Tag property of the node created in step 30 equal to the ObjectId of

' the unerased object. This will allow us to record it's ObjectId for recognition in

' other events. Use e.DBObject.ObjectId.ToString()

' 32. Create a Private Sub named callback\_ObjectReappended. This is the Sub that

' will be called when an Object is ReAppended to the Database. (The name needs to be

' the name used in the Delegate parameter of step 19). The first parameter is an

' object. (Use ByVal and sender as the name of the Object). The second parameter is

' an ObjectEventArgs. (Use ByVal and e as the name of the ObjectEventArgs)

' Note: Put the End Sub after step 34

' 33. Add the class name of the object to the tree view

' Declare a TreeNode variable. (System.Windows.Forms.TreeNode). Instantiate

' it using the Add method of the Nodes property of the TreeView on the UserForm1

' created in step 4. Use the ObjectEventArgs passed into the method for the string

' parameter and use the "Type" of DBObject. (e.DBObject.GetType().ToString())

' 34. Record its id for recognition later

' Make the Tag property of the node created in step 33 equal to the ObjectId of

' the unerased object. This will allow us to record it's ObjectId for recognition in

' other events. Use e.DBObject.ObjectId.ToString()

' 35. Create a Private Sub named callback\_ObjectUnappended. This is the Sub that

' will be called when an Object is UnAppended from the Database. (The name needs to be

' the name used in the Delegate parameter of step 20). The first parameter is an

' object. (Use ByVal and sender as the name of the Object). The second parameter is

' an ObjectEventArgs. (Use ByVal and e as the name of the ObjectEventArgs)

' Note: Put the End Sub after step 39

' 36. Here we will search for an object in the treeview control so it can be removed.

' Create a For Each statement. Use node for the element name and the type is

' Forms.Treenode. The group paramater is the Nodes in the TreeView.

' (myPalette.TreeView1.Nodes)

' Note: Put the "Next" statement after step 39

' 37. Use and "If Then" statement and see if the node is this the one we want.

' compare the node.Tag to the ObjectId. (use e.DBObject.ObjectId.ToString)

' Note: Put the "End If" after step 39

' 38. If we got here then this is the node for the unappended object.

' call the Remove method of the node.

' 39. Exit the For loop.