Autodesk University 2003
December 2-5, 2003 ◇ MGM Grand Hotel Las Vegas

Going Beyond DCL

Joel Roderick

CP21-3 I know what you're thinking: "Why learn ObjectDCL when there's VBA?" Are you tired of hand coding DCL? Are you wondering if you should just learn VBA and forget all this DCL stuff? This class will introduce you to the basics of ObjectDCL and it's abilities. We will discuss how ObjectDCL compares with DCL, VBA, and ARX, and why ObjectDCL might be perfect for you. Come learn about a great tool that many can benefit from.

About the Speaker:

Joel is currently Project Designer and CAD Technologies Manager at Water Technology, Inc., an aquatic planning, design, and engineering firm in Beaver Dam, Wisconsin. At Water Technology, Joel is responsible for the conceptual design of aquatic recreation / competition facilities & waterparks, as well as all AutoCAD management and customization. For the past 5 years, Joel has been developing Water Technology's proprietary AutoCAD plug-in, Aquatic Desktop using Visual Lisp, VBA, and ObjectDCL.

jroderick@watertechnologyinc.com

The basics of event driven programming…

Definitions:

- ObjectDCL (ODCL) – an arx application that enables the use of ObjectARX style forms with lisp applications
- Form – Synonym for dialog box
- Control – An item on a form (button, list box, combo boxes)
- Event – Happens when the user interacts with a control (action_tile)
- Method – A function to manipulate forms & controls
- Modal form – A form that has focus until closed (most AutoCAD dialogs)
- Modeless form – A form that allows action to be taken outside it's boundaries (Aerial View)
- Dockable form – Similar to modeless with the added ability to be docked (Properties)
- Config Tab – Adds a custom tab to the options command

Files:

- ObjectDCL.arx and ObjectDCL2004.arx – Required to be loaded in AutoCAD. This file is freely distributable and royalty free. This files is necessary to display and handle the forms designed with ObjectDCL
- ObjectDCL.exe – The form editor
- .odc – ObjectDCL project files. These files contain your form designs.
- .ods – Secure ObjectDCL project file. This file is optional and is used as a secure format and cannot be opened in the ODCL editor. Be sure you always keep your .odc files!
- .lsp – AutoLisp / VisualLisp file

File Organization:

The ObjectDCL.arx, .lsp & .odc/.ods files should be located in a folder that is within AutoCAD's search path.

ObjectDCL Projects:

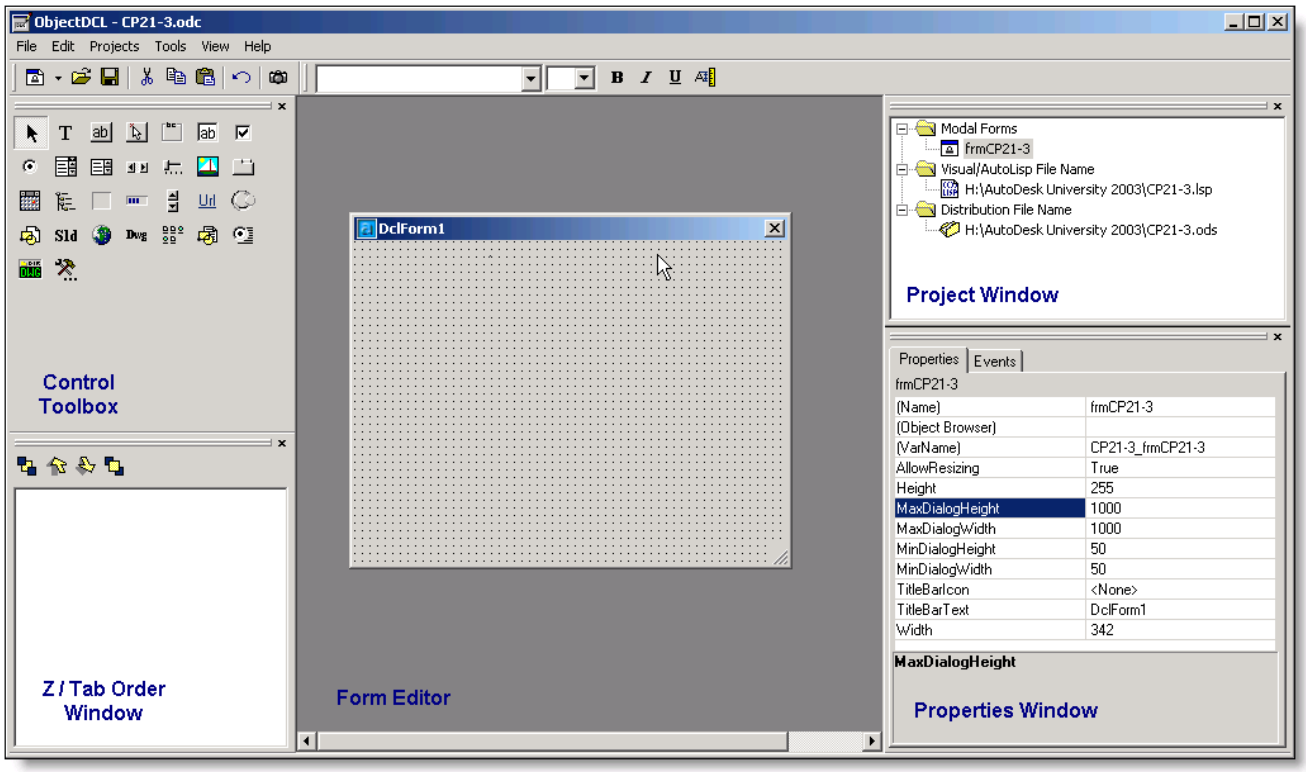
A single ObjectDCL project can contain many forms of many types. So if your application needs more than one form, you can have one project file that contains all of your forms. Project files contain forms only, there is no lisp code saved within these files.

The ObjectDCL Editor…

The basic components of the ODCL editor:

- Form Editor – The main window where you design forms
- Project Window – Where you organize your forms, lisp, and distribution files
- Control Toolbox – Where you get the controls for your form
- Properties Window – Controls the properties of an object
- Z/Tab Order – Controls what happens when user hits TAB & also controls the display order of controls
- Intelligent Help – Explore the different methods and properties of an object
- Picture Folder – Stores images for use within a project
- Property Wizard – Provides a convenient way of changing the properties of a control.
- Font Toolbar – Provides a convenient way of changing the fonts of a control(s).

There is also a toolbar for frequently used commands, but for the purpose of this class, we will be using the menus.



ObjectDCL Boot Camp…

Rather than try to explain many things at one time, I am going to go step by step through the process of creating an ObjectDCL project and explain things along the way. There are too many controls, properties and methods to cover here. This class is intended to help you understand the basic concepts behind ODCL. Here is an outline of the process:

- Create a project
- Add a form
- Associate a lisp file
- Add controls
- Add the code behind the controls

First things first, lets create a project…

There are three basic components of an ODCL project.

- .odc file
- .ods file
- .lsp / .fas / .vlx file

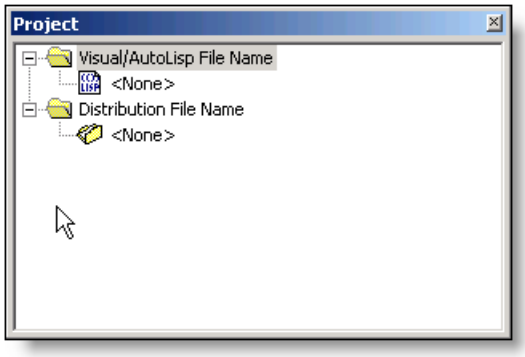
It is recommended that you keep all of these files in the same folder. Also, make sure that the folder you place these files in is part of AutoCAD's search path.

Save the file:

Start ObjectDCL and click File->Save and name the file.

Associate a .lsp and distribution file:

The next step in the process is to associate a lisp file and a distribution file to your ODCL project. To do this, we are going to use the Project window. First you will need to create a lisp file to associate to the project. You can do this with your favorite lisp editor or just plain notepad.



Associate a lsp file:
Double click on <None> under Visual/AutoLisp File Name and browse to your program.

Associate a distribution file:
Distribution files are a form of security. A distribution file has an .ods extension and cannot be opened in the ODCL editor. This prevents other from opening your ODCL project in the ODCL editor and modifying the forms. Double click on <None> under Distribution File Name, browse to your folder, and name the file.

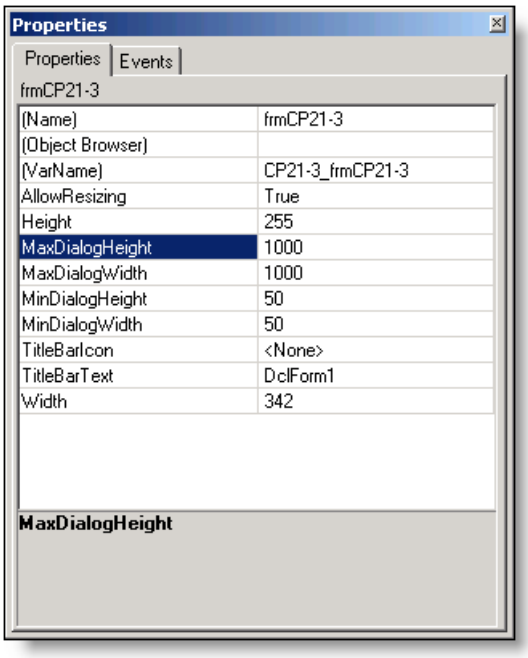
Let the Fun Begin – Designing Forms…

There are 5 types of forms supported by ObjectDCL. All 5 types work basically the same with some caveats. Modeless, dockable forms and config tabs are handled slightly different than modal forms. For example, modeless and dockable forms stay open, so you will need to account for when a user opens a drawing, or switches drawings.

- Modal form – A form that has focus until closed (most AutoCAD dialogs)
- Modeless form – A form that allows action to be taken outside it's boundaries (Aerial View)
- Dockable form – Similar to modeless with the added ability to be docked (Properties)
- Config Tab – Adds a custom tab to the options command
- File Dialog – A customizable file browser

Adding a form:

For the purpose of this class, we will use a modeless form since these are the most common. First lets start by adding a form. Click Projects ->Add Modal Form.

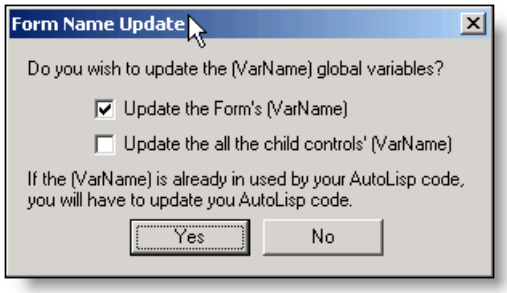


Properties

Now that you have added the form, we need to adjust the properties of the form. To do this we will use the Properties Window. The Properties Window works very similar to the one in VB(A). If you click on a property, a description of what the property does is displayed at the bottom of the Properties window. Most of the properties, for the most part, are self-explanatory.

The VarName property

One property that is special to ODCL is the VarName property. VarName is a global variable that is created when the form is shown in AutoCAD. This variable gives you direct access to whatever object it points to. In this instance VarName points to the form itself. The VarName variable will allow you to make changes to the form from lisp, so you can change the size of the form, disable the form, change the color, etc. all from lisp. The global variables that are created when showing a form are released when the form is closed, so there are no memory issues.



Updating the VarName property

ODCL will name the VarName property based on the ODCL project name and the form name. For example, if your project name is "MyODCL" and the form name is "frmMyODCL" then ODCL will set VarName as "MyODCL_frmMyODCL". ODCL has made it easy to keep your VarNames organized by automating the task of updating the VarName property. Any time you change the name of your form, you will be given the choice of letting ODCL update the VarName property. We will cover more on VarNames later.

Getting from ODCL to AutoCAD…

In this section, we will cover the code required to ensure that ObjectDCL.arx is loaded, load the project and then show the form. Open the lisp file that you associated to your ODCL project in your favorite lisp editor. For the purposes of this class, we will use the Visual Lisp IDE that comes with AutoCAD.

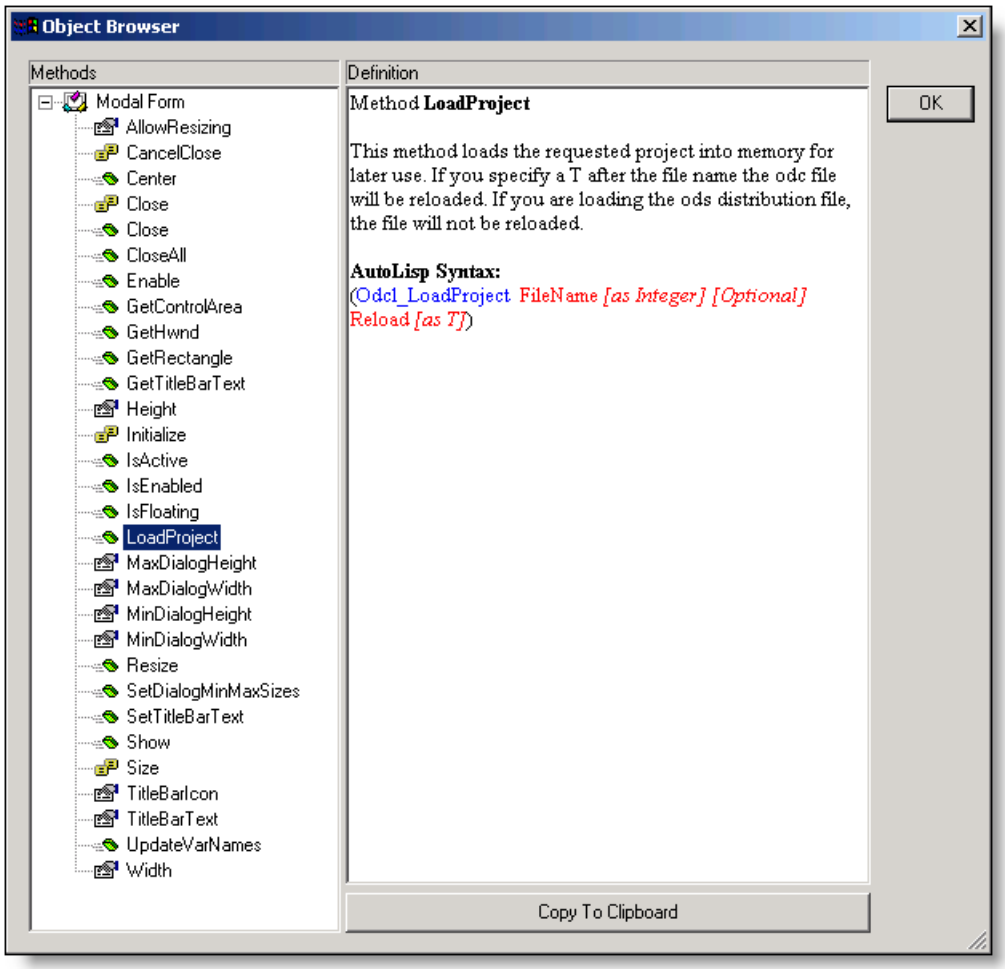
Loading ObjectDCL2004.arx

The code to load ObjectDCL2004.arx is simple. Make sure that this code is added to the lisp file and make sure it is run before you try to show the form. Here is an example:

```
;;function to load objectdcl2004.arx
(defun LoadODCL ()
  (if (not (member "ObjectDCL2004.arx" (arx)))
    (arxload "ObjectDCL2004.arx" "ObjectDCL2004.arx not found.")
  )
)
```

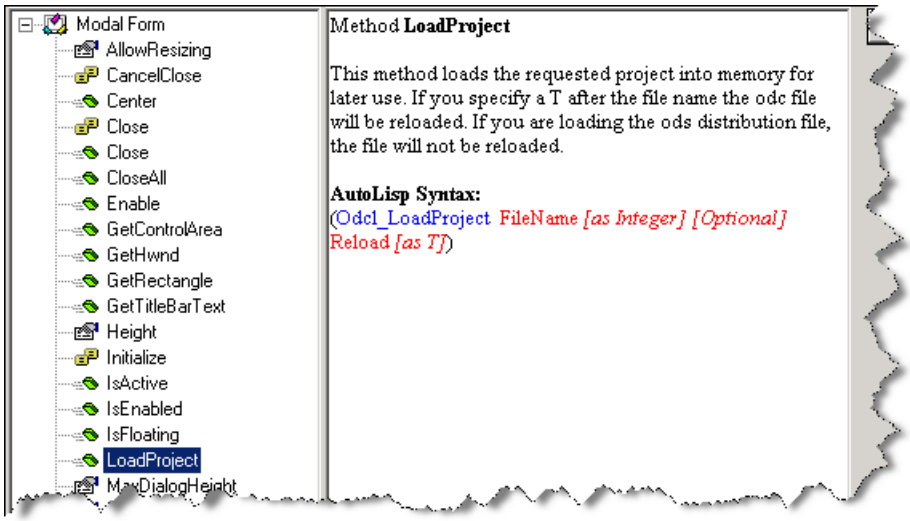
Loading the ODCL Project

To get the code required to actually show the form, we need to go back to the ODCL editor. We are going to use the Intelligent Help and let ODCL do most of the work.



Intelligent Help

To access the Intelligent Help, right click on the form and click Intelligent Help. The Intelligent Help will list all the methods, properties and events that the form supports. As you highlight the different methods and properties, the appropriate code is generated with additional information about that property or method. There are too many properties and methods to cover them all here. The Intelligent Help does a fairly good job of explaining what each method and property does. It's really not important that you know all the methods and properties rather, it's important that you understand the concept behind the Intelligent Help.



LoadProject Method

This method is used to load the ODCL project into memory. You will need to run this method before any others. Rather than having to type all the code, ObjectDCL has provided a convenient button at the bottom of the Intelligent Help to copy the code to the clipboard so you can paste it into the Visual Lisp IDE. Highlight the LoadProject method and click "Copy to Clipboard"

Pasting the code

Now that you have the code saved in your clipboard, go back to the Visual Lisp IDE and paste the code. You will notice that there are some optional arguments. Since they are optional, we can delete that portion, or we can supply the arguments. The arguments are explained in the Intelligent Help (see above). If ObjectDCL2004.arx is loaded, the ODCL functions are recognized by Visual Lisp and are blue.

```
;;load the project
(Odcl_LoadProject  FileName [as Integer] [Optional]  Reload [as T])
```

Reload flag

Since we are in the process of designing the form, we want to use the "Reload" flag so the project is reloaded each time the command is run. Otherwise, the one that is stored in memory will be used. Once you have finished your application, the "Reload" flag can be removed, so that ODCL will use the project stored in memory instead of reloading the project each time the command is run.

```
;;load the project
(Odcl_LoadProject "CP21-3.odc" T)
```

Showing the Form

Now that the code to load the project is complete, we will use the "Show" method to get the form to show up in AutoCAD. Go back to the Object Brower and highlight the "Show" method, then click "Copy to Clipboard".

Paste the code

Switch over to the Visual Lisp IDE and paste the code.

```
;;Show the form
(Odcl_Form_Show CP21-3_frmCP21-3 [Optional] UpperLeftXCoordinate [as Integer] [Optional] UpperLeftYCoordinate [as Integer])
```

Unless you want to show your form at a specific location on the screen, you can delete the optional arguments:

```
;;Show the form
(Odcl_Form_Show CP21-3_frmCP21-3)
```

VarName

ODCL automatically used the variable stored in the VarName property of the form as the argument.

[Object Browser]	
(VarName)	CP21-3_frmCP21-3
AllowResizing	True
Height	228

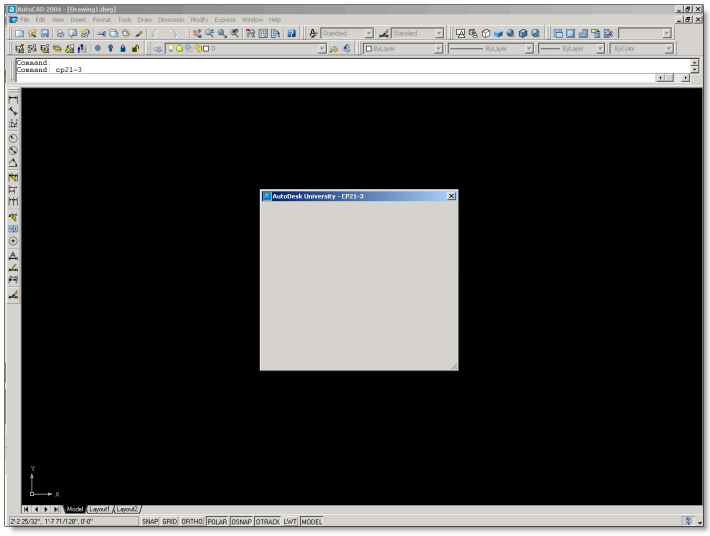
Making a command:

We now have all the code to show the form. Now we need put it all together into a command.

```
(defun c:CP21-3 ()

  ;;load ObjectDCL
  (LoadODCL)

  ;;load the project
  (Odcl_LoadProject "CP21-3.odc" T)

  ;;Show the form
  (Odcl_Form_Show CP21-3_frmCP21-3)

)
```

"De-fun" part

Putting it all together into a command is really no different than any other lisp program. In this case, the command name is "CP21-3". Now all that needs to be done is load the lisp file and type the command.

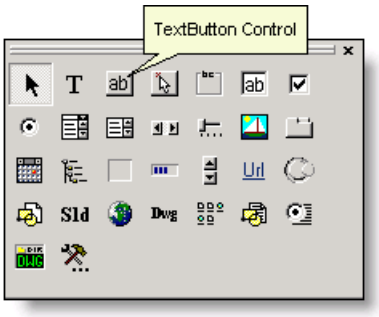


The Form in AutoCAD

You can now run the "CP21-3" command in AutoCAD to test the form. Now that we have them form ready, we can start adding controls. To close the form, click on the X in the upper right corner of the form.

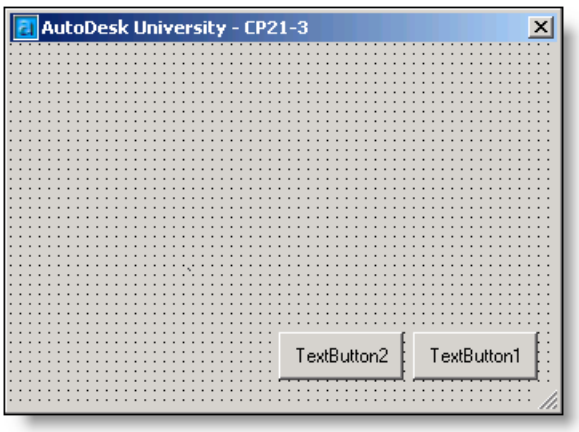
Getting Things Under Control…

To add controls to the form, we are going to use the Control Toolbox. ODCL comes with many standard controls, but also has the ability to use most ActiveX controls. There are too many controls to cover here, however we will be using the Intelligent Help, and they are all explained there.



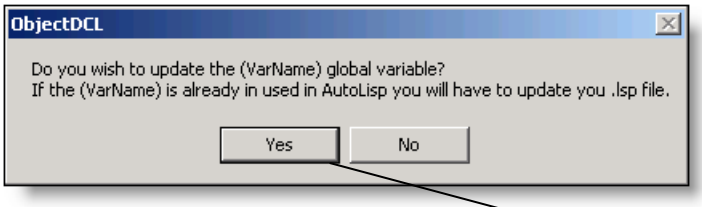
Buttons

Let's start off with the two things most forms need, OK and Cancel buttons. For these buttons we will use the TextButton control. Adding controls is as easy as clicking on the control and specifying the location on the form.



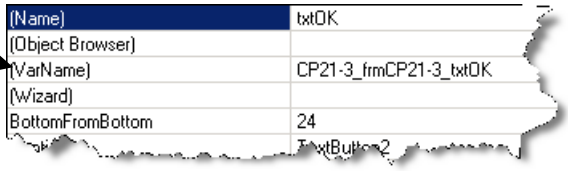
Name Property

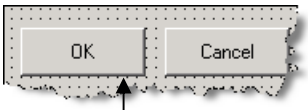
Add two TextButton Controls to the form. After adding the buttons we will need to edit the properties of each button. Highlight the left button and change the "Name" property to txtOK. Repeat the same for the right button, naming it txtCancel

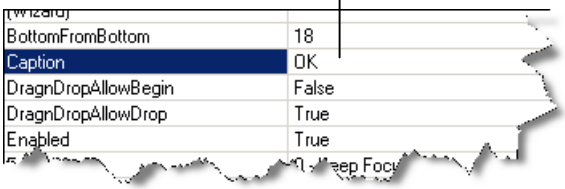


Control VarName Property

Since all controls have a VarName property, ODCL will give you the choice to update the VarName property after changing the Name property. The VarName property of a control gives you direct access to the individual controls on a form.





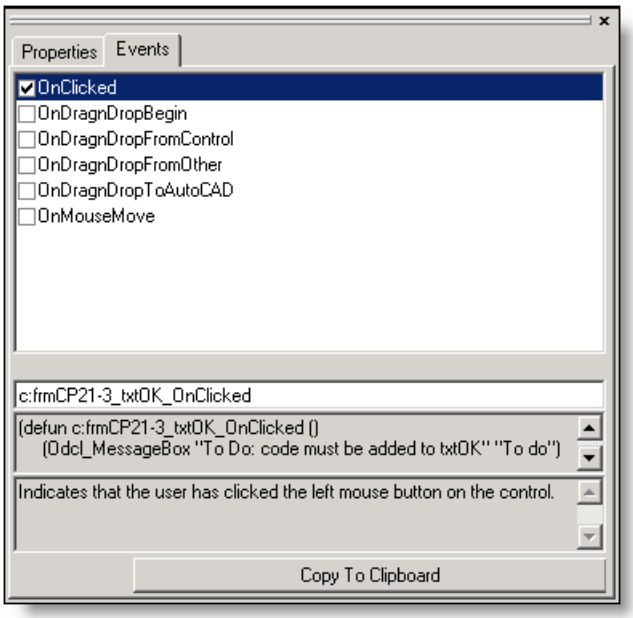


Caption Property

Next, we will change the Caption property of the TextButton Control. The Caption property contains the actual text that is on the control. Change the Caption property to "OK". Repeat the same for the right button naming it "Cancel"

Making the controls alive…

Now that we have our buttons ready, we can add the code to actually make them do something. To accomplish this, we are going to use events. The DCL equivalent of an event would be the (action_tile) function. ODCL has many more events to choose from, and each control and form has it's own set of events.



Adding Events

To add events, we will use the Properties Window in the lower right corner of the editor. Highlight the OK button, and then click the Events tab in the Properties Window. As you can see, there are many events for a TextButton control. The description and code for each event is shown at the bottom of the Intelligent Help. Again, ODCL has provided a convenient "Copy to Clipboard" button at the bottom to save the code to the clipboard.

Where to paste the code

Since each control will have it's own function, it's easy to have many functions within you program. Since these functions are only used when the command in running, we can put the control functions within the main defun and localize the control functions. For modeless and Dockable forms, the event functions need to be available at all times, so these functions should go outside the main defun.

```
;;onclick event for the OK button
(defun c:frmCP21-3_txtOK_OnClicked ()
  (Odcl_MessageBox "To Do: code must be added to txtOK" "To do")
)
```

Adding the OnClicked Event

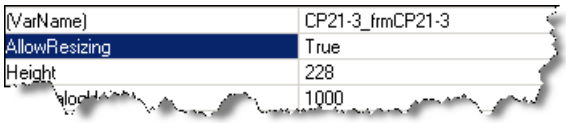
The event we are going to use is OnClicked. This event is fired when the user clicks the control with the left mouse button. Check the box next to OnClicked and then click "Copy to Clipboard". Switch over to the Visual Lisp IDE and paste the code. ODCL puts in "dummy code" for each event that consists of an alert box telling you to add code. We will leave the code as-is for now and come back to it later.

```
(defun c:CP21-3 ()

  ;;load ObjectDCL
  (LoadODCL)

  ;;onclick event for the OK button
  (defun c:frmCP21-3_txtOK_OnClicked ()
    (Odcl_MessageBox "To Do: code must be added to txtOK" "To do")
  )

  ;;onclick event for the Cancel button
  (defun c:frmCP21-3_txtCancel_OnClicked ()
    (Odcl_MessageBox "To Do: code must be added to txtCancel" "To do")
  )

  ;;load the project
  (Odcl_LoadProject "CP21-3.odc" T)

  ;;Show the form
  (Odcl_Form_Show CP21-3_frmCP21-3)

)
```

Next, repeat the process for the cancel button, and paste the code into the Visual Lisp IDE. Your code should look like example to the left. You can now load the lisp program, and run the command. **Be sure to save the ODCL project before running the command, otherwise, the changes will have no effect.** Now that you have added code to the controls, clicking them will trigger the event functions.

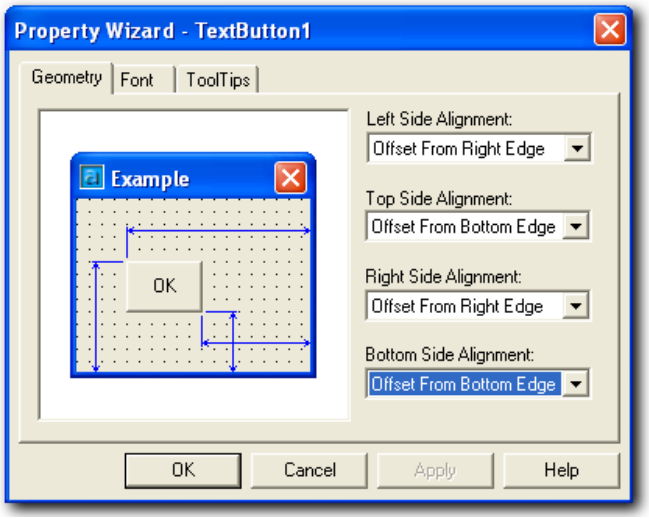
Resizing forms…

ODCL forms can be resized making them much more functional than DCL dialogs. Making your form resizable is easy, however there are some caveats when it comes to making your controls stay in the right place when you resize the form.



AllowResizing Property

Highlight the form and change the AllowResizing property to True. If you save the ODCL project and run the command, you will be able to resize your form, but the controls don't really work the way we want.

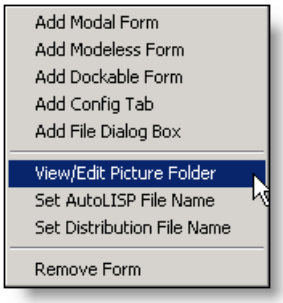


Property Wizard

To control the way controls react when resizing the form, we are going to use the Property Wizard. To access the Property Wizard, double click on the desired control. We want the OK and Cancel buttons to stay in the lower right hand corner of the form when resizing. Change all 4 options on the Geometry tab to ensure that the buttons stays anchored to the lower right corner.

Kicking it up a notch with images…

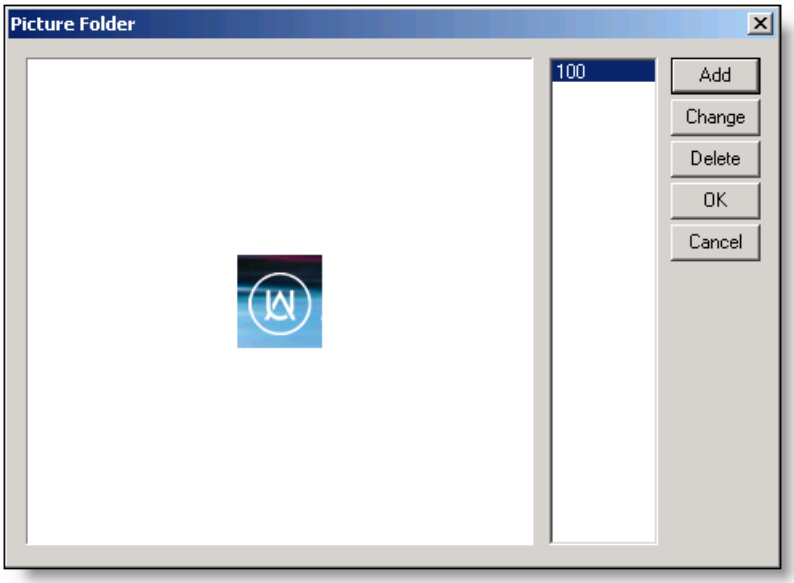
A common question that gets asked is "How can I show my company logo on the form". While this is possible with DCL using a slide and image_show, it is pretty limiting. ODCL provides a way to add images to forms and controls easily. In this section, we will discuss the Picture Folder in ODCL.



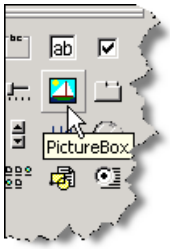
The Picture Folder

ODCL uses the Picture Folder to store images that will be used within the ODCL project. The nice thing about the picture folder is that you don't have to keep the actual image with the project. Once you add your image to the Picture Folder, the image is saved within the ODCL project. To access the Picture Folder, click Projects -> View/Edit Picture Folder.

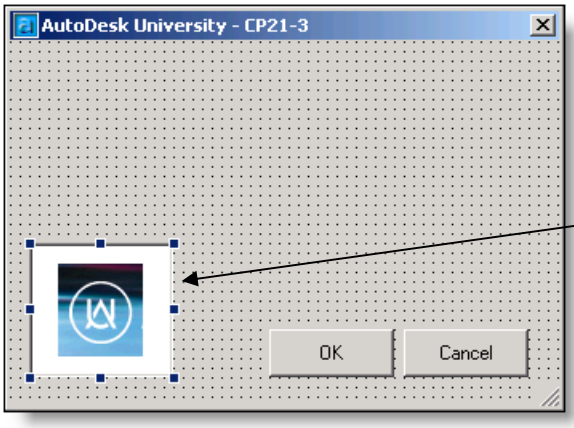
Please note: 3rd Day Software recommends that large jpeg files should not be loaded directly into the picture folder. The jpegs will take up a large amount of memory when the project file is loaded into AutoCAD. Instead use the PictureBox's function called LoadPictureFile instead at run time and this will minimize the amount of memory being used since the loaded picture is released from memory as soon as the dialog box is closed.



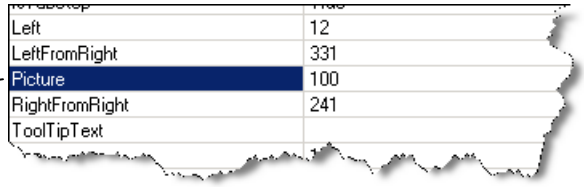
Adding Images to the Picture Folder
Adding an image is quite easy. Now that you have the Picture Folder open, click Add and select an image file. You can add as many images you want. ODCL will assign each image a number. You can now reference the images according to this number.

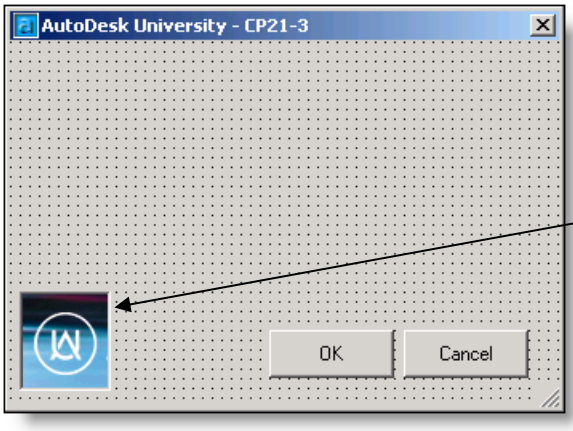


Adding the image to a control
To add the image to the form, we are going to use the PictureBox Control. This control allows you to add most image types to any form. It also has many events and methods you can use to add some nice features to your program, like adding a hyperlink to an image. On the Control Toolbox, click the PictureBox Control, and then specify the location of the control on the form. After adding the control, change the Name property to "picLogo" and answer yes to update the VarName property. Then, you can specify which image you want shown on the control. To do this, we will use the Picture property.



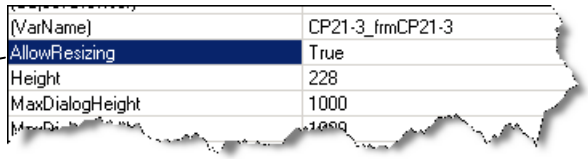
Picture Property
Make sure the PictureBox control is highlighted and change the Picture property to the number associated with the image you want. In this case the image number is 100.

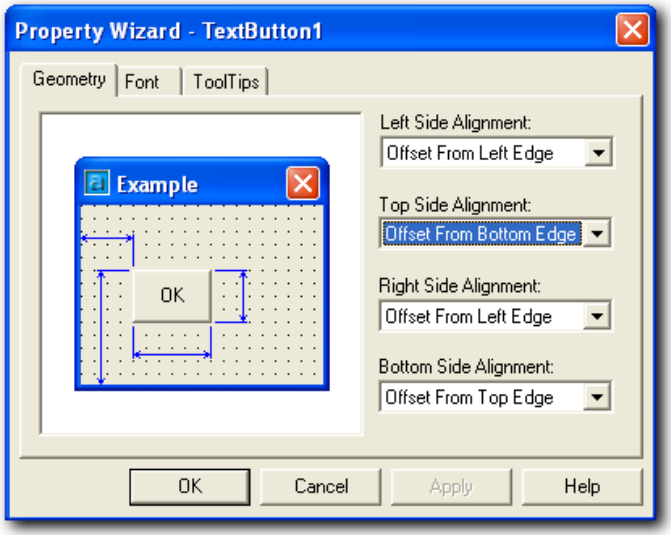




AutoSize Property

To make the control shrink to fit tightly around the image, you can use the AutoSize property. Make sure the PictureBox control is highlighted and change the AutoSize property to true.



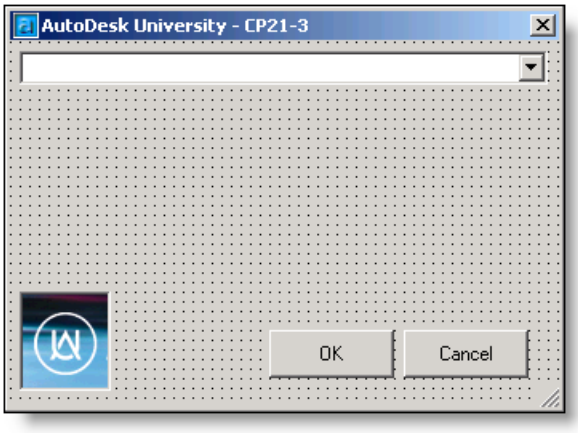


Reminder!

Don't forget to use the Property Wizard to control what happens to the PictureBox control when the form is resized. Also, be sure you save the ODCL project before you run the command in AutoCAD.

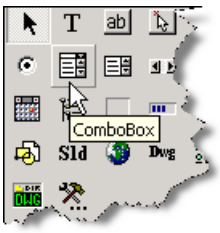
Making something useful….

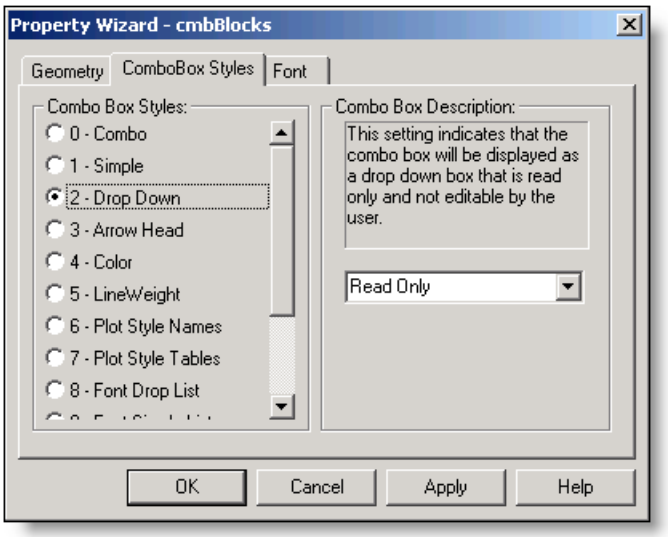
To finish this example off and make something useful out of what we have done so far, let's add a ComboBox control that will list all the blocks within the drawing.



ComboBox Control

A combo box control is used for listing items. On the ControlToolbox, click the ComboBox control and specify the location on the form. Change the Name property of the ComboBox control to "cmbBlocks", and answer yes to update the VarName property.



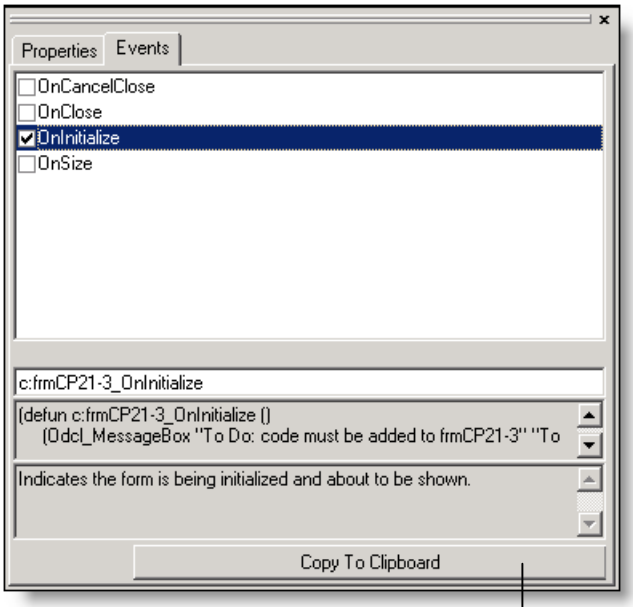


ComboBox Styles
ODCL has many different styles of ComboBoxes. You can change the style of the ComboBox by using the Property Wizard. Double click on the control to see the different options and a description of what they do. For this example, we will use the "Drop Down" style.

Reminder!
Don't forget to use the Property Wizard to control what happens to the ComboBox control when the form is resized.

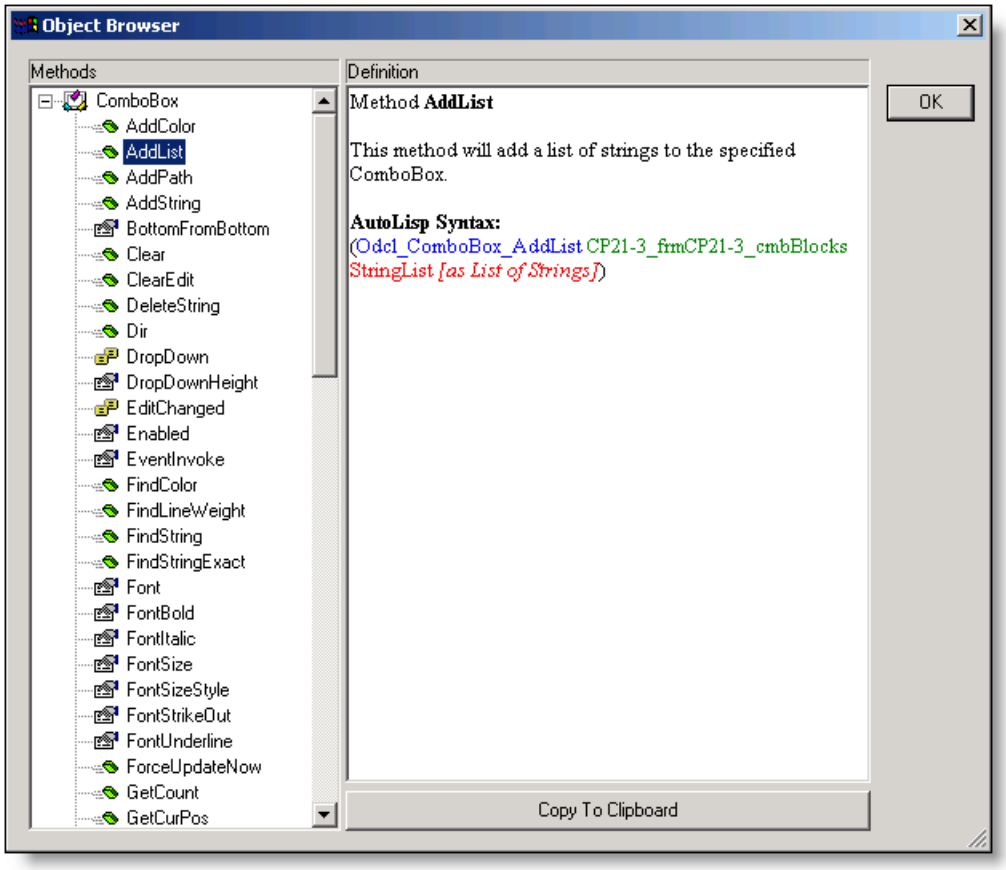
Filling the ComboBox

We want the ComboBox to contain a list of all the blocks within a drawing. To do this, we are going to use the OnInitialize event of the form to populate the ComboBox as the form is shown.



OnInitialize Event
Hightlight the form and switch over to the Events tab on the Properties Window. Check the OnInitialize event and click "Copy to Clipboard", then paste the code into the Visual Lisp IDE.

```
;;oninitialize event function
(defun c:frmCP21-3_OnInitialize ()
  (Odcl_MessageBox "To Do: code must be added to frmCP21-3" "To do")
)
```



AddList Method
To fill the ComboBox, we will need to use the AddList method of the ComboBox. This method takes a list and populates the ComboBox.

Using the Intelligent Help highlight the AddList method of the ComboBox and click "Copy to Clipboard", then paste the code into the Visual Lisp IDE in the OnInitialize event function.

Since we want to list all of the blocks within a drawing, we will need a function to generate that list. Here is an example:

```
;;function to list all the blocks within the drawing
(defun GetBlockList (/ BlockList)
  (vlax-for item (vla-get-blocks (vla-get-activedocument (vlax-get-acad-object)))
    (if (and (not (wcmatch (vla-get-name item) "*Paper_*"))
             (not (wcmatch (vla-get-name item) "*Model_*"))
        )
      (setq BlockList (cons (vla-get-name item) BlockList))
    )
  )
)
```

```
;;oninitialize event function
(defun c:frmCP21-3_OnInitialize ()
  (Odcl_ComboBox_AddList CP21-3_frmCP21-3_cmbBlocks (GetBlockList))
)
```

We are going to run this method when the form is shown, by using the OnInitialize event of the form. Your OnInitialize event function should look like the example code to the left.

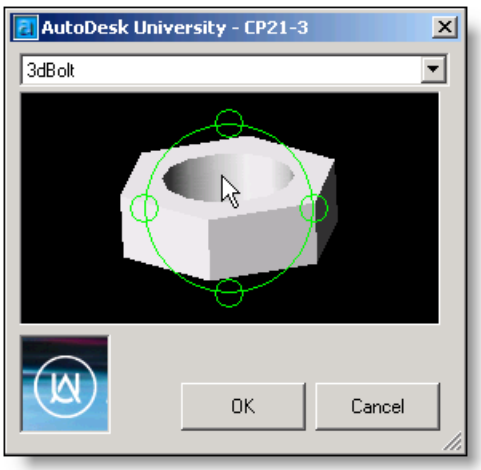
Checkpoint!

By now we have covered quite a bit of information. At this point you should test the form and code to see if everything is working as planned. Below is what your code should look like. Be sure to save your ODCL file before running the command!

```
;;function to load objectdcl2004.arx
(defun LoadODCL ()
  (if (not (member "ObjectDCL2004.arx" (arx)))
    (arxload "ObjectDCL2004.arx" "ObjectDCL2004.arx not found.")
  )
)

(defun c:CP21-3 ()

  ;;load ObjectDCL
  (LoadODCL)

  ;;function to list all the blocks within the drawing
  (defun GetBlockList (/ BlockList)
    (vlax-for item (vla-get-blocks (vla-get-activedocument (vlax-get-acad-object)))
      (if (and (not (wcmatch (vla-get-name item) "*Paper_*"))
               (not (wcmatch (vla-get-name item) "*Model_*"))
          )
        (setq BlockList (cons (vla-get-name item) BlockList))
      )
    )
  )

  ;;oninitialize event function
  (defun c:frmCP21-3_OnInitialize ()
    (Odcl_ComboBox_AddList CP21-3_frmCP21-3_cmbBlocks (GetBlockList))
  )

  ;;onclick event for the OK button
  (defun c:frmCP21-3_txtOK_OnClicked ()
    (Odcl_MessageBox "To Do: code must be added to txtOK" "To do")
  )

  ;;onclick event for the Cancel button
  (defun c:frmCP21-3_txtCancel_OnClicked ()
    (Odcl_MessageBox "To Do: code must be added to txtCancel" "To do")
  )

  ;;load the project
  (Odcl_LoadProject "CP21-3.odc" T)

  ;;Show the form
  (Odcl_Form_Show CP21-3_frmCP21-3)

)
```

Finishing Touches…

To wrap things up, we will add a BlockView control to the form, so we can preview the blocks. A BlockView control will also allow a user to pan, zoom, and orbit within the control. Then we will complete the OK and Cancel button code. Below is what the final command will look like along with the completed code.



Hopefully you now have an understanding of the concepts behind ODCL and can take what you have learned here and apply the same concepts to the other types of forms and controls.

```
;;main function
;;notice that we can localize the event functions
(defun c:CP21-3 (/
                 GetBlockList
                 BlockName
                 c:frmCP21-3_cmbBlocks_OnSelChanged
                 c:frmCP21-3_OnInitialize
                 c:frmCP21-3_txtOK_OnClicked
                 c:frmCP21-3_txtCancel_OnClicked
                )
  ;;load ObjectDCL
  (LoadODCL)

  ;;function to list all the blocks within the drawing
  (defun GetBlockList (/ BlockList)
    (vlax-for item (vla-get-blocks (vla-get-activedocument (vlax-get-acad-object)))
      (if (and (not (wcmatch (vla-get-name item) "*Paper_*"))
               (not (wcmatch (vla-get-name item) "*Model_*"))
          )
        (setq BlockList (cons (vla-get-name item) BlockList))
      )
    )
  )

  ;;onselchanged event function
  (defun c:frmCP21-3_cmbBlocks_OnSelChanged (nSelection sSelText /)
    ;;notice that we can use the arguments to get the text of the selected item
    (setq BlockName sSelText)
    (Odcl_BlockView_DisplayBlock CP21-3_frmCP21-3_blkPreview BlockName)
  )

  ;;oninitialize event function
  (defun c:frmCP21-3_OnInitialize ()
    (Odcl_ComboBox_AddList CP21-3_frmCP21-3_cmbBlocks (GetBlockList))
  )

  ;;onclick event for the OK button
  (defun c:frmCP21-3_txtOK_OnClicked ()
    (Odcl_Form_Close CP21-3_frmCP21-3)
    (command "_.insert" BlockName pause pause pause)
  )

  ;;onclick event for the Cancel button
  (defun c:frmCP21-3_txtCancel_OnClicked ()
    (Odcl_Form_Close CP21-3_frmCP21-3)
  )

  ;;load the project
  (Odcl_LoadProject "CP21-3.odc" T)

  ;;Show the form
  (Odcl_Form_Show CP21-3_frmCP21-3)

)
```

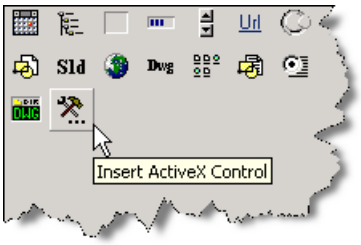
Advanced Topics:

Making your program MDI aware

The ODCL help file explains the technique fairly well. Here is a snippet from the help file:
"To make the dockable and modeless forms MDI aware requires that your code updates the dialog box after every time a different drawing receives focus. The event "OnDocumentActivated" has been added to Dockable and Modeless forms so that you may receive notification from the dialog box that it needs to be updated. Your program is responsible to update what the dialog box displays to the user. Note that the event is not fired when a new drawing is created or a drawing file is opened. With these two cases your lisp code that is auto-loaded should handle the initialization of the dialog box."

ActiveX Controls

ODCL supports the use of most ActiveX controls. You can access "non-ODCL" controls by clicking the "Insert ActiveX Control" button on the Control Toolbox. Once you have an ActiveX control on your form, you can use the same concepts outlined above to manipulate them. If you use an ActiveX control that is not installed on a machine that will be running your program, you will need to register the control on that machine. ODCL also has a function called (Odcl_RegisterActiveXCtrl) that makes this task easier.



Drag and Drop

ODCL supports drag and drop functionallity. The user can drag from a control or AutoCAD to another control or to AutoCAD. Each control that supports drag and drop has four events to support this feature.

- DragnDropBegin - Indicates the user has just begun a drag and drop selection from this control.
- DragnDropToAutoCAD - Indicates the user has just dragged and dropped on to the AutoCAD Drawing from this control.
- DragnDropFromControl - Indicates the user has just dragged and dropped from another control to this control.
- DragnDropFromAutoCAD - Indicates the user has just dragged and dropped a selection from the AutoCAD Drawing to this control.

EventInvoke Property

If you intend to use the (command) function in your program, you will need to make sure that the EventInvoke property is set to 1. A setting of 1 will force ODCL to give focus to the command line allowing the (command) function to be used. A setting of 0 will keep the focus on the control.