# Module1: Objects and their Properties

Amongst Ken Iverson's many aphorisms was the cautionary dictum not to labour too hard on any particular explanation of any particular fine point of APL notation (such as `0⍴0`) because it is as likely to be an indication of an unsuccessfully hurdled hurdle behind the tutor as an especially difficult conceptual leap for that audience. This course is not so good: it labours every nook and cranny. It is a labour of love, but nevertheless every page can be justly criticised on some grounds by sharp students. I just hope somebody learns something true and useful from it. Hold your breath .. here goes …

> Simula 67 introduced in 1967 most of the key concepts of **object-oriented (OO) programming** - objects, classes, subclasses (involving inheritance) and virtual procedures. Most (but not all) of the objects discussed below have a graphical (visual) manifestation. The first **graphical user interface (GUI)** was designed by Xerox Corporation's Palo Alto Research Center in the 1970s. It was not until the 1980s and the emergence of the Apple Macintosh that GUIs started to prevail, and not until the appearance of Microsoft Windows in the early 1990s that GUIs became ubiquitous.

Modern computer applications, including operating systems themselves, are designed using object-oriented architecture. Microsoft's graphical user interface has evolved into an archetypically object-oriented collection of buttons, forms, and other progressively more complex active constructs. Most of Dyalog's 71 or so Microsoft-based GUI objects are virtually tangible, arguably real and thoroughly useful.

1.1 For a comprehensive list of primitive GUI objects available in Dyalog APL version 10, see [Help][GUI Help][Objects Overview]. Or type `Type` into the APL session and hit **F1**. Or see the invaluable Dyalog APL *Object Reference* manual. Also load workspace `WINTRO` and follow the 56 lessons. And load `WTUTOR` and follow 37 tutorials in there. Then load `WTUTOR95` and follow the 18 extra tutorials in there. Alternatively, begin your investigations at http://www.dyalog.com [Products]. But first follow Modules 1-3 below ☺.

## § 1.1 Object Spaces

### §§ 1.1.1 Creating vanilla Namespaces with ⎕NS

```
CVec ⎕NS ''          ⍝ Creates an empty Namespace, named in CVec
```

1.1.1.1 Create an empty namespace called MySpace.

```
)CS Name             Changes Space into Namespace named Name
```

1.1.1.2 Change space into your new namespace and verify `⊢(⎕NL ⍳4)≡0 0⍴''`

Read this as: "It happens to be true that `⎕NL ⍳4` matches an empty character matrix." Compare with `⊢1∨0` which reads: "It is necessarily true that `1∨0` ."

```
)CS                  Changes Space to the Root space named #
```

1.1.1.3 Change back to the Root space using system command `)CS` and examine the result of `⎕NL 9`

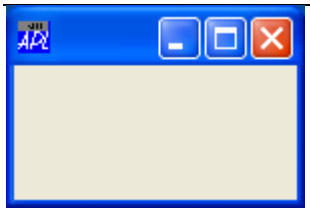
### §§ 1.1.2 Creating GUI Object Spaces with ⎕WC

```
CVec2 ⎕WC CVec1      ⍝ Creates GUI Object name in CVec2 ; Type in CVec1
```

1.1.2.1 Create an object of `Type Form` with name MyForm. The structure of the right argument of this system function can be much more complicated (essentially *Name-Value* pairs) as we shall see later.

```
)OBS                 Displays a list of global objects
```

1.1.2.2 Use system command `)OBS` to verify the existence of MyForm although its existence is manifest:



1.1.2.3 Try moving and resizing your `Form` with the mouse.

Other objects have less obvious existence, such as the `SysTrayItem` object created by typing

```
'STI'⎕WC'SysTrayItem'
```

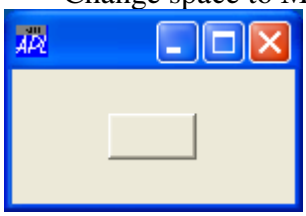
But notice the new APL icon that has appeared in the system tray at bottom right of your task bar.

### §§ 1.1.3 Changing Space with ⎕CS

| | |
|---|---|
| `⎕CS CVec` | ⍝ Changes current space to object named in `CVec` |

An object of a particular `Type` can only exist as the child of parents of particular `Type`s. For example, a `Button` can be the child of a `Form` but not of a `SysTrayItem` object.

1.1.3.1 Change space to MyForm and create an object of `Type Button` with name MyButton.



The `Button` has no `Caption` because this property has not yet been specified.

1.1.3.2 Change into MyButton space and verify `⎕NL ⍳9↳0 0⍴''` then change back to the Root space (`#`).

## § 1.2 Properties of Object Spaces

Objects have properties. The properties determine the specific appearance and behaviour of individual objects. For example, the `Size` property, which is common to many objects, determines the height and width of the object.

### §§ 1.2.1 Examining Properties of an Object with ⎕WG

Some information about MyForm can be discovered by right-clicking on the word MyForm in the APL Session and then selecting the menu item [Properties]. However, the value of a specific individual property is found by means of the property name.

| | |
|---|---|
| `Arr ← CVec2 ⎕WG CVec1` | ⍝ Gets the value of property `CVec1` of object `CVec2` |

1.2.1.1 Get the `Size` of MyForm. Go into MyForm space and get the `Size` of MyButton.

| | |
|---|---|
| `)PROPS` | Reports the properties in the space of the current object |

1.2.1.2 Display the list of properties in the MyForm space and the MyButton space. These are the intrinsic properties of `Form`s and `Button`s respectively. Examine the values of various properties. Return to `#`.

### §§ 1.2.2 Setting Properties of an Object with ⎕WS

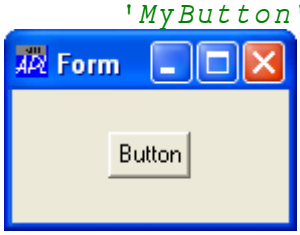
| | |
|---|---|
| `CVec ⎕WS CVec_Arr` | ⍝ Sets prop-value pair `CVec_Arr` of object `CVec` |

For example,

```
'MyForm'⎕WS'Size' (6.5 11)
```

sets the `Size` of MyForm to 6.5% by 11% of the full screen size.

```
      'MyForm'⎕WS'Caption' 'Form' ⋄ ⎕CS 'MyForm'
      'MyButton' ⎕WS'Caption' 'Button'
```

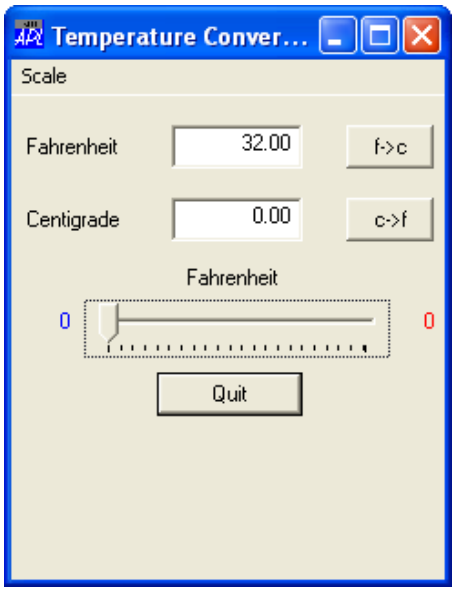



1.2.2.1 Try getting and setting the values of various properties of `Form`s and `Button`s. Return to `#` and erase your `Form` with `)ERASE` or `⎕EX`.

Note that it is possible to get (`⎕WG`) and set (`⎕WS`) many properties in a single call, and to set many properties at create time (`⎕WC`). Indeed some properties, such as the `Points` property, *must* be set at create time, as must `Type`. Most properties have sensible default values and do not need to be explicitly set on most occasions.

### §§ 1.2.3 Building complex Objects

`⎕WC` returns a shy result of the name of the object just created (Larg). Here are some lines that illustrate the flexibility of the right argument (Rarg) of `⎕WC` and `⎕WS`. Note that *Name-Value* pairs do not need the *Name* if specified in the default property order (see for example the `MenuBar` and `Menu` below).




```
w←,⊂'Type' 'Form'
w,←⊂'Caption' 'Temperature Converter'
w,←⊂'Size'(266 238)
w,←⊂'Coord' 'Pixel'
⎕CS'Temp'⎕WC w
⎕CS'MB'⎕WC'MenuBar'
⎕CS'M'⎕WC'Menu' 'Scale'
'C'⎕WC'MenuItem' 'Centigrade'
'F'⎕WC'MenuItem' 'Fahrenheit'
⎕CS'#' ⋄ ⎕CS'Temp' ⋄ num←'FieldType' 'Numeric'
'LF'⎕WC'Label' 'Fahrenheit'(16 8)(24 72)
'F'⎕WC'Edit' ''(16 88)('Decimals' 2)num
'F'⎕WS('ValidIfEmpty' 1)('Value' 32)('Size' 24 72)
'F2C'⎕WC'Button' 'f->c'(16 184)(24 48)
'LC'⎕WC'Label' 'Centigrade'(56 8)(24 72)
'C'⎕WC'Edit' ''('Decimals'(2))num
'C'⎕WS('ValidIfEmpty' 1)('Size' 24 72)('Posn' 56 88)
'C2F'⎕WC'Button' 'c->f'(56 184)(24 48)
'LTB'⎕WC'Label' 'Fahrenheit'(88 96)(24 72)
'LLO'⎕WC'Label' ''(112 16)('Decimals' 0)num
'LLO'⎕WS('FCol'(0 0 255))('Value' 0)('Size' 24 16)
'TB'⎕WC'TrackBar'('Limits'(0 212))('Posn'(112 40))
'TB'⎕WS('Size'(32 168))('TickSpacing' 10)
'LHI'⎕WC'Label' ''(24 24)('Decimals' 0)num
'LHI'⎕WS('FCol'(255 0 0))('Value' 0)('Posn' 112 208)
'Q'⎕WC'Button' 'Quit'(152 80)(24 80)('Default' 1)
```

1.2.3.1 Verify that a `Menu` is only visible if it has a non-empty `Caption` set. A `MenuBar` is never visible.

| | |
|---|---|
| `)NS` | Reports the name of the current space |

## § 1.3 Property Variables

It is possible to create ordinary APL variables in vanilla namespaces.

1.3.1 Create a namespace called MySpace and inside it create a variable called MyVar. Verify that the variable is only visible inside MySpace and is invisible from the Root space.

### §§ 1.3.1 Exposing Object Properties with ⎕WX

| ⎕WX | ⍝ Whether GUI names are exposed |
|---|---|

⎕WX is a localisable system variable that determines whether or not the names of properties, methods and events provided by Dyalog APL GUI objects are exposed. The value of ⎕WX in a clear workspace is defined by the **default_wx** parameter (see the definitive Dyalog APL *User Guide*). Check ⊢⎕WX←0. For every object there is a property called PropList.

1.3.1.1 Create a Printer object. Check that the value of ⎕WX has been inherited from the Root. Type PropList and get a VALUE ERROR. Now assign ⎕WX to 1 and again type PropList. The property name has now been *exposed*. The listed keywords are case dependent . Show they cannot be erased.

### §§ 1.3.2 Assigning Properties with ←

Notice that most properties, having been exposed, act just like variables in the object space. Go into a Form's object space and type

```
      Size
50 50
```

1.3.2.1 By manipulating object properties, the "state" of an object may be changed and influenced. Try assigning Size and notice the Form change size. Show that there is a minimum Size of a Form. Investigate the effect on Size and Posn when setting the Coord property to 'Pixel'. Every object has a Data property to which may be assigned any array. Any other variable may be assigned within the object space and treated like a new property. Verify these statements.

1.3.2.2 Create a Bitmap object in the Root with the Bits property set to a 150 by 150 matrix of random numbers between 0 and 15. In a Form, assign the Picture property to the name of the Bitmap that you have just created. This should display the Bitmap in the centre of your Form in colours chosen from the first 16 rows of the default colour map. Now assign the CMap property of the Bitmap to a 16 by 3 matrix of random numbers between 0 and 255. The picture changes each time this property is set.

1.3.2.3 You can get more control over the Bitmap. Create an Image object as a child of a Form and then assign the Picture property of the Image to the name of the Bitmap. Assign the Dragable property of the Image to 2 and use the mouse to drag the Image around the Form. (Remember that the Points property must be set at create time.)

1.3.2.4 Using a Poly object, draw a solid triangle in the middle of a Form. Rotate it.

### §§ 1.3.3 Rebuilding complex Objects

Essentially, properties are variables and can be treated as such. All use of ⎕WG and ⎕WS can be eliminated as of Dyalog version 9.0.

1.3.3.1 Rewrite the lines of code in §§1.2.3 to eliminate the use of ⎕WS and to minimise the Rarg of ⎕WC.

Now you have to learn lots about each primitive object available to you in Dyalog APL. Typing any GUI keyword into the APL Session and hitting **F1** brings up the 'well navigable' *GUI Help* file at the page of the selected keyword topic. This is backed up by the extensive and thorough (could one ever again say comprehensive?) Dyalog APL *Object Reference* manual.

1.3.3.2 Ask for the next module on **methods and events**. How did you get on with Module 1? ☺.