DlgArea Class

The DlgArea class defines an area of a UserDialog and provides co-ordinates of and within it.
It is a helper for the DefineDialog Method of an OODialog UserDialog Object and has no effect on any Windows Object.

To use objects of the DlgArea or DlgAreaU class include this line in your code:

```
::requires 'dlgarea.cls'
```

Init

```
>>- aDlgArea~Init(--X--,--Y--,--Width--,--Height--+-------------+--)--><
                                                  +--,--Margin--+
```

Arguments:

The arguments you pass to the new method when creating a DlgArea Object are:

X	The *X* Co-ordinate in dialog units relative to the UserDialog of the top left corner of the area you wish to define.
Y	The *Y* Co-ordinate in dialog units relative to the UserDialog of the top left corner of the area you wish to define.
Width	The width in dialog units of the area of the UserDialog you wish to define
Height	The height in dialog units of the area of the UserDialog you wish to define
Margin	An inner margin within the DlgArea in Dialog Units to be left blank. The default is 5.

Example:

```
u=.dlgArea~new( 0, 0, self~SizeX,Self~SizeY) /* u is whole of UserDialog */
b=.dlgArea~new(u~x('70%'),u~y ,u~w('R'),u~h('R')) /* sub area for buttons */
```

Methods:

X

```
>>--ADlgArea~x(--+------+--)-->< 
                 +--n---+
                 +--n%--+
```

Returns an X Coordinate in dialog units relative to the dialog area.

If no n is passed returns the x coordinate of the left margin.
If n is passed and is positive, then returns the x coordinate n dialog units to the right of the left margin.
If n is passed and is negative, then returns the x coordinate n dialog units to the left of the right margin.
If n is passed and is a percentage, then returns the x coordinate n% of the way between the left and right margins

Y

```
>>--ADlgArea~y(--+------+--)-->< 
                 +--n---+
                 +--n%--+
```

Returns a Y Coordinate in dialog units relative to the dialog area.

If no n is passed returns the Y coordinate of the top margin.
If n is passed and is positive, then returns the y coordinate n dialog units below the top margin.
If n is passed and is negative, then returns the y coordinate n dialog units above the bottom margin.
If n is passed and is a percentage, then returns the y coordinate n% of the way between the top and bottom margins

W (a synonym of CX may be used for W)

```
>>--ADlgArea~w(--+-------+--)-->< 
                 +--n%---+
                 +--'R'--+
```

Returns a width in dialog units relative to the dialog area.

If no n is passed returns the inter-margin width
If n is a %age returns n% of inter-margin width
If 'R' is passed returns Remaining width between last ~x and right margin

WR

```
>>--ADlgArea~wr
```

Returns the remaining width between last ~X and right margin in dialog units. This is equivalent to ~W('R').

H (A synonym of CY may be used for H)

```
>>--ADlgArea~h(--+-------+--)--><
                 +--n%---+
                 +--'R'--+
```

Returns a height in dialog units relative to the dialog area.

If no n is passed returns the inter-margin height
If n is a %age returns n% of inter-margin height
If 'R' is passed returns Remaining height between last ~y and bottom margin

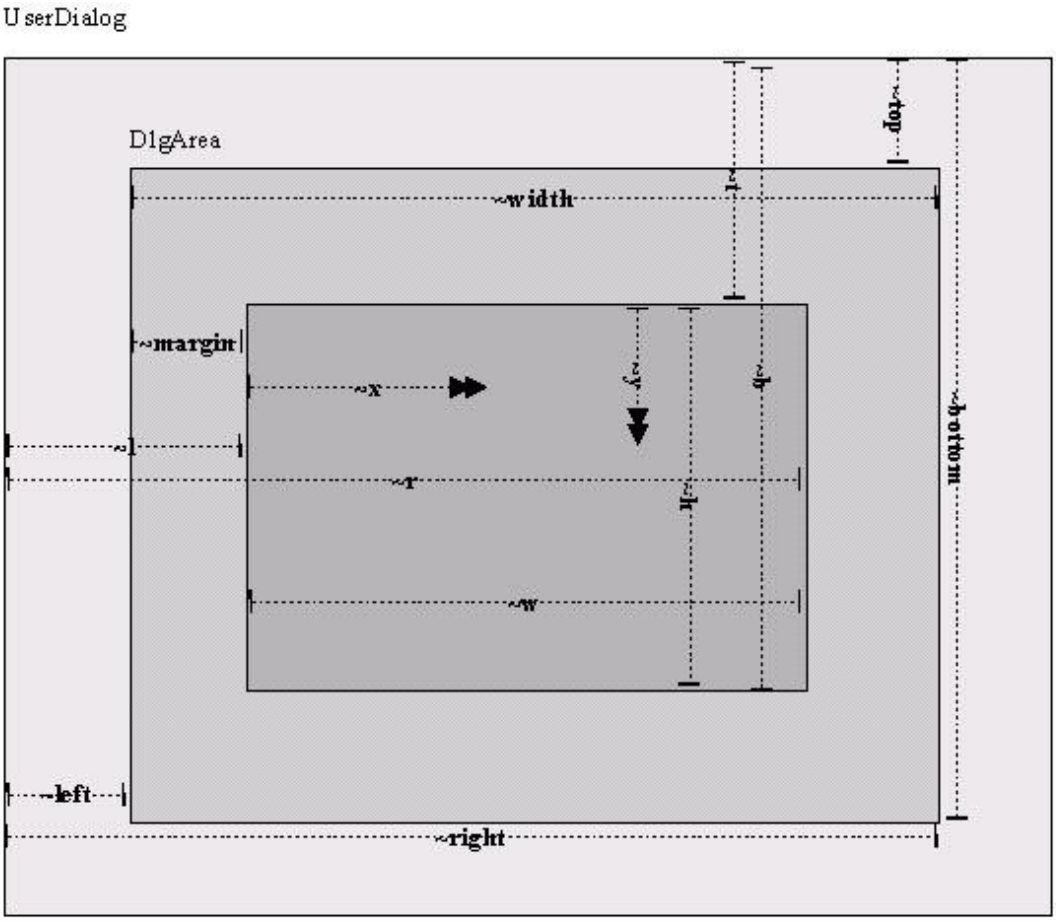
HR

```
>>--ADlgArea~hr--><
```

Returns the remaining height between last ~Y and bottom margin in dialog units. This is equivalent to ~H('R').

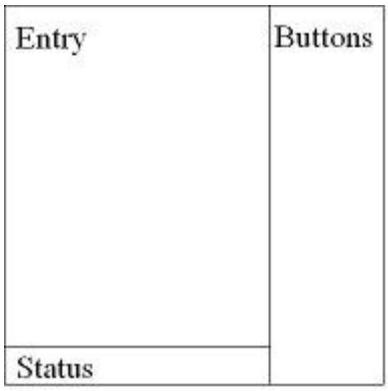
Attributes

L	x coordinate of Left Margin in Dialog units relative to UserDialog
R	x coordinate of Right Margin in Dialog units relative to UserDialog
T	y coordinate of Top Margin in Dialog units relative to UserDialog
B	x coordinate of Bottom Margin in Dialog units relative to UserDialog
Left	x coordinate of Left Edge in Dialog units relative to UserDialog
Right	x coordinate of Right Edge in Dialog units relative to UserDialog
Top	y coordinate of Top Edge in Dialog units relative to UserDialog
Bottom	y coordinate of Bottom Edge in Dialog units relative to UserDialog
Margin	Size in dialog units of DlgArea Margin



Example
On a UserDialog we want the top left to be an entry area, with an area for buttons on the right and a status area below.

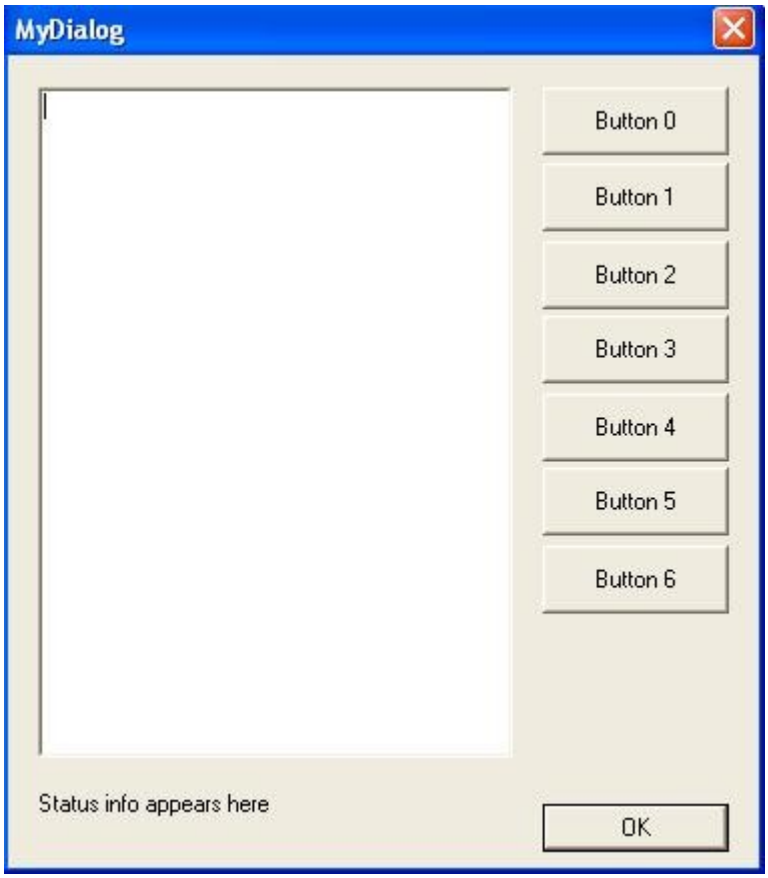
We decide to give 70% of the width and 90% of the height to the entry area.
We leave the margin as the default.
We want all the areas to resize themselves if we change the UserDialog size except the OK button should remain 15 units high.



Our Dialog DlgArea plan

Here is what our DefineDialog method might look like

```
/* ---------------------------------------------------------------------- */
::method DefineDialog
/* ---------------------------------------------------------------------- */
/* define DlgArea named u as whole of user dialog                         */
u=.dlgArea~new(  0       ,         0,self~SizeX,Self~SizeY)   /* whole dlg   */

/* define DlgArea named e within DlgArea u for entry line                 */
e=.dlgArea~new(u~x       ,u~y       ,u~w('70%'),u~h('90%'))

/* define DlgArea named s within DlgArea u for Status Area in remaining height*/
s=.dlgArea~new(u~x       ,u~y('90%'),u~w('70%'),u~hr      )

/* define DlgArea named b within DlgArea u for Button area                */
b=.dlgArea~new(u~x('70%'),u~y       ,u~wr      ,u~hr      )

/* entry line coterminous with area e margins                             */
self~AddEntryLine(12,'text',e~x,e~y,e~w,e~h,'multiline')

/* Status Area Coterminous with area s margins                            */
self~AddText(s~x,s~y,s~w,s~h,'Status info appears here',11)

/* Seven buttons evenly spaced at 10% intervals, 9% high                  */
do i = 0 to 6
   self~addButton(12+i,b~x,b~y((i * 10)||'%'),b~w,b~h('9%'),'Button' i,'Button'||i)
end /* DO */

/* ok button 15 dialog units high at bottom of area b                     */
self~AddButton(1,b~x,b~y(-15),b~w,15,'OK','OK','Default')
```



The resultant dialog

DlgAreaU Class

The DlgAreaU class is a subclass of the DlgArea class that assists the creation of dynamically resizable Dialogs.

To use objects of the DlgArea or DlgAreaU classes include this line in your code:
```
::requires 'dlgarea.cls'
```

Init
```
>>- aDlgAreaU~Init(Dialog--+-------------------------+--------+-)-->
                          +--,--Margin-+            |        |
                          +--,-------- +--,-NoResize-+        |
                          +--,---------+--,----------+-NoMove-+
```

The DlgAreaU object creates a dialog area coterminous with the calling dialog.
It inherits all the methods and attributes of the DlgArea Class (listed above).

Arguments:

Dialog:	An Object that is SubClassed from A UserDialog
Margin	See DlgArea Margin
NoResize	A set of Dialog Ids of widgets not to be resized during a resize
NoMove	A set of Dialog Ids of widgets not to be moved during a resize

Attributes:

NoResize	A set of Dialog Ids of widgets not to be resized during a resize
NoMove	A set of Dialog Ids of widgets not to be moved during a resize
LastError	.nil or details of error parsing calling dialog's DefineDialog method
CorrectionFactor	Adjusts the ratio of top & left margins to bottom & right (default=1.05)

Creating resizable Dialogs:

You can use the DlgAreaU Object to facilitate in creating Dynamically resizable Dialog objects, but to do so you must carefully satisfy all the requirements below.

The Dialog must be created with the '`ThickFrame`' option.
It may be created with the UserDialog Create method or CreateCenter method.

Your Dialog init method must include the line
```
self~connectResize('OnResize')
```

Your DefineDialog method must start with these lines:
```
expose u
u=.dlgAreaU~new(self)          /* whole dlg   */
```

Your DefineDialog method must not reference variables within the Add Method parameters, although you can use references to DlgArea attributes, so
```
Self~AddButton(23,1,5,6,'MyButton','Pressed')
self~addButton(14,b~x,b~y('10%'),b~w,b~h('9%'),'Button' 1,'Button'||1)
```
Are both OK, whereas
```
/* Seven buttons evenly spaced at 10% intervals, 9% high                    */
do i = 0 to 6
```

```
   self~addButton(12+i,b~x,b~y((i * 10)||'%'),b~w,b~h('9%'),'Button' i,'Button'||i)
end /* DO */
```

would now fail as the variable i is referenced within the parameters.

To debug your DlgAreaU call insert the following after the instantiation

```
if u~lastError \= .nil then call errormessage u~lastError
```

Your Dialog must contain the following method:

```
/* ---------------------------------------------------------------------- */
::method OnResize
/* ---------------------------------------------------------------------- */
expose u
use arg dummy,sizeinfo
                                  /* wait for last size event msg then resize */
if self~PeekDialogMessage~left(8) \= "OnResize" then u~resize(self,sizeinfo)
```

The DlgAreaU method will then automatically resize & place the following widgets in your dialog when the dialog frame is dragged or the minimize, maximise or restore buttons are pressed:

BLACKFRAME	**BLACKRECT**	**BITMAPBUTTON**	**BUTTON**
CHECKBOX	**COMBOBOX**	**ENTRYLINE**	**GRAYFRAME**
GRAYRECT	**GROUPBOX**	**LISTBOX**	**LISTCONTROL**
PASSWORDLINE	**PROGRESSBAR**	**RADIOBUTTON**	**SCROLLBAR**
SLIDERCONTROL	**TABCONTROL**	**TEXT**	**TREECONTROL**
WHITEFRAME	**WHITERECT**		

The following widgets can be defined in a UserDialog, but cannot be handled by the DlgAreaU resize method, so should not be used.

BUTTONGROUP	**CHECKBOXSTEM**	**CHECKGROUP**	**COMBOINPUT**
FULLSEQ	**INPUT**	**INPUTGROUP**	**INPUTSTEM**
OKCANCEL…	**RADIOGROUP**	**RADIOSTEM**	

All of these widgets can be achieved using combinations of the permitted methods.

Resizing tokenized dialogs

DlgAreaU resizing works by parsing the code of your defineDialog method. If you tokenise your code then that source is not available. You can still use DlgAreaU to resize your dialog, but you have to do some of the work for it. So when you declare dlgAreaU include the following code:

```
U = .dlgAreaU~new(self)
U~dlgObjList = .list~new
```

And every time you add an item to your dialog you must also register it for resizing by inserting a line in the list composed of the items id, x, y, width & height all delimited by '@' character.

```
c = .dlgArea~new(u~x,u~y,u~w,10)
self~addText(c~x,c~y,c~w,c~h,'Header line','CENTER',10)
u~dlgObjList~insert('10'||'@'||c~x||'@'||c~y||'@'||c~w||'@'c~h)
```

Your dialog should now resize even if tokenized

Known possible bug:

The DlgAreaU resize method can create slightly over-size margins on the left & bottom of the Dialog. To correct for this the DlgAreaU class has a CorrectionFactor attribute set by default to 1.05 which in tests appears to neutralise this effect. If your dialogs have over (or under) sized margins, you may be able to correct this in your code by overiding the Correction Factor like this (for example):

```
U=.DlgAreaU~new(self)
U~CorrectionFactor=1.05
```

You will have to experiment to find the apropriate setting for this attribute.

Sample Code

```
/* DlgAreaDemo.Rex  --  Demonstrate DlgArea & DlgAreaU Classes  --  Feb 2006 */

MyDlg=.MyDialog~new
MyDlg~execute('ShowTop')
MyDlg~DeInstall

exit
::requires 'dlgArea.cls'
::requires 'OODWIN32.CLS'
/* ============================================================================ */
::class MyDialog Subclass UserDialog
/* ============================================================================ */
::method Init
/* ---------------------------------------------------------------------------- */
  self~Init:super
  rc=self~CreateCenter(250,250,'MyDialog','ThickFrame',,"MS Sans Serif",8)
  self~InitCode=(rc=0)
  self~connectResize('OnResize')

/* ---------------------------------------------------------------------------- */
::method DefineDialog
/* ---------------------------------------------------------------------------- */
expose u

u=.dlgAreaU~new(self)                                        /* whole dlg   */
if u~lastError \= .nil then call errormessage u~lastError
e=.dlgArea~new(u~x       ,u~y       ,u~w('70%'),u~h('90%'))   /* edit   area */
s=.dlgArea~new(u~x       ,u~y('90%'),u~w('70%'),u~hr      )   /* status area */
b=.dlgArea~new(u~x('70%'),u~y       ,u~wr      ,u~hr      )   /* button area */

self~AddEntryLine(12,'text',e~x,e~y,e~w,e~h,'multiline')
self~AddText(s~x,s~y,s~w,s~h,'Status info appears here',,11)

self~addButton(13,b~x,b~y('00%'),b~w,b~h('9%'),'Button' 0,'Button'||0)
self~addButton(14,b~x,b~y('10%'),b~w,b~h('9%'),'Button' 1,'Button'||1)
self~addButton(15,b~x,b~y('20%'),b~w,b~h('9%'),'Button' 2,'Button'||2)
self~addButton(16,b~x,b~y('30%'),b~w,b~h('9%'),'Button' 3,'Button'||3)
self~addButton(17,b~x,b~y('40%'),b~w,b~h('9%'),'Button' 4,'Button'||4)
self~addButton(18,b~x,b~y('50%'),b~w,b~h('9%'),'Button' 5,'Button'||5)
self~addButton(19,b~x,b~y('60%'),b~w,b~h('9%'),'Button' 6,'Button'||6)
self~AddButton( 1,b~x,b~y('90%'),b~w,b~h('9%'),'Ok','Ok','DEFAULT')

/* ---------------------------------------------------------------------------- */
::method Unknown
/* ---------------------------------------------------------------------------- */
use arg msgname, args
StatusLine = Self~GetStaticControl(11)
StatusLine~SetTitle('You Pressed' msgname)

/* ---------------------------------------------------------------------------- */
::method OnResize
/* ---------------------------------------------------------------------------- */
expose u
use arg dummy,sizeinfo
                                /* wait for last size event msg then resize */
if self~PeekDialogMessage~left(8) \= "OnResize" then u~resize(self,sizeinfo)
```

This achieves the same Dialog as the previous sample, but now it is resizable by dragging the frame.

Jon Wolfers (Sahananda) February 2006