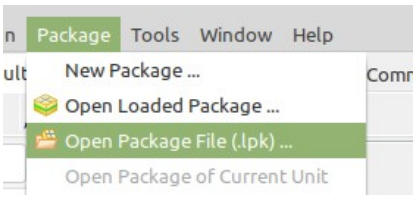
The Package Multis

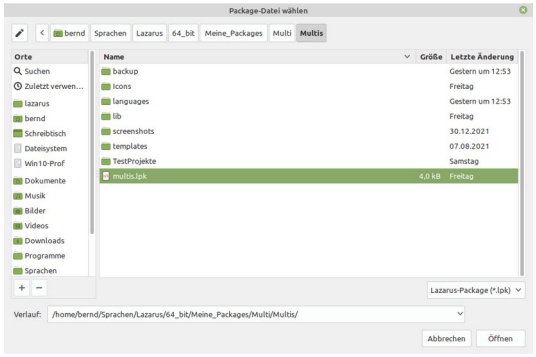
This text was translated with Deepl and my poor school English.

Installation

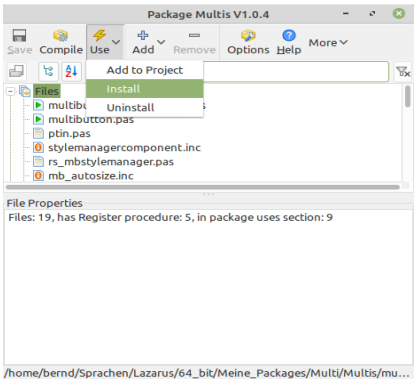
The package is located in the following Github account: https://github.com/wennerer/Multis
After the package has been cloned or downloaded, it can be installed in Lazarus. To do this, open Lazarus and click on Open Package File (.lpk)... under Package.



Now navigate to the Multis folder and select the file multis.lpk.

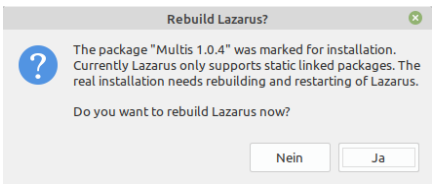


The following window will open:

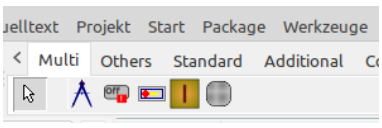


Click on Use and then Install.

Confirm this dialogue with Yes:



Now there is a new Multi tab in the palette selection.



TMultiPanel

Properties

Action	: The Action associated with the control
Align	: Specifies the placement of the control inside its Parent.
Anchors	: The set of anchor definitions for this control
Autosize	: Allows automatic adjustment of the size for the control, according to its content
BidiMode	: Customization (of text controls) in bidirectional reading environments
BorderSettings	: The properties of the border
BorderSettings.Between	: The space between inner- and outerborder
BorderSettings.InnerColor	: The color of the innerborder
BorderSettings.InnerWidth	: The width of the innerborder
BorderSettings.OuterColor	: The color of the outerborder
BorderSettings.OuterWidth	: The width of the outerborder
BorderSpacing	: Determines the inner and outer border spacing for this control
Caption	: The text that the user writes in the panel
CaptionAlignment	: Alignment of the text in the caption (left, center, right)
CaptionHorMargin	: The horizontal distance of the text in the text rectangle (only effective with taLeftJustify)
CaptionLayout	: Alignment of the text in the caption (top, center, bottom)
CaptionVerMargin	: The vertical distance of the text in the text rectangle (only effective with tlTop)
CaptionWordbreak	: Allows a line break in the caption
ColorEnd	: The end color of the panel (for color gradient)
ColorGradient	: The direction of the gradient
ColorStart	: The start color of the panel (for color gradient)
Constraints	: The minimum and maximum Width and Height for the control
Cursor	: The shape of the mouse pointer, when the mouse is over this control
DoubleBuffered	: Allows to reduce flicker in the painting of the control
DragCursor	: The cursor shape shown while the control is dragged
DragKind	: The operation when the control is dragged - Drag or Dock
DragMode	: Allows the user to drag the control
DrawACustomPanel	: Opens an editor where you can draw a panel
DropDownMenu	: The properties of the dropdownmenu
DropDownMenu.Active	: Activates the dropdown function
DropDownMenu.Compressed	: Properties of the compressed panel
DropDownMenu.Compressed.Active	: Makes the selection the starting value
DropDownMenu.Compressed.Height	: The height of the compressed panel
DropDownMenu.Compressed.Width	: The width of the compressed panel
DropDownMenu.Direction	: The fold-out direction
DropDownMenu.Hotspot	: Defines the area in which a click is effective, only active with DropDownMenu.Active and trPinned (only at runtime!)
DropDownMenu.Speed	: The drawing speed (timer intervall)
DropDownMenu.Step	: The drawing steps (pixels)
DropDownMenu.Stretched	: Properties of the streched Panel

DropDownMenu.Stretched.Active	: Makes the selection the starting value
DropDownMenu.Stretched.Height	: The height of the stretched panel
DropDownMenu.Stretched.Width.	: The width of the stretched panel
DropDownMenu.Trigger	: Trigger
Font	: The font to be used for text display in this panel
Height	: The vertical size of the control
HelpContext	: The ID for context-sensitive Help on this control
HelpKeyword	: The keyword for context-sensitive Help on this control
HelpType	: Determines whether context-sensitive Help is selected by numeric ID or keyword
Hint	: The text to show in the Hint window for the control
ImageIndex	: The Index of a Image in a ImageList
ImageLeft	: The coordinate of the left edge of a Image
Images	: A list for including images
ImageTop	: The coordinate of the top edge of a Image
ImageWidth	: The unique width of all images in the list
Left	: The client coordinate of the left edge of the control
RndRctRadius	: Corner diameter if the geometric shape is RoundRect
Style	: The geometric shape of the panel
Top	: The client coordinate of the top edge of the control
Visible	: Allows to show or hide the control, and all of its children
Width	: The horizontal extent of the control
Appear	: makes the panel appear (only at runtime!)
Disappear	: makes the panel disappear (only at runtime!)
AnimationSpeed	: Speed for Appear bzw. Disappear (default 0,05) (only at runtime!)
ParentAsBkgrd	: Background of the panel takes on the colour of the parent (only at runtime!)

Events

OnChangeBounds	: Event handler for a change of the Bounds of the control
OnClick	: Notification handler for mouse clicks
OnCompressed	: Handler when the panel is compressed, only active when DropDownMenu.Active
OnDragDrop	: This handler determines the action on an drop onto this control, in a drag-drop operation
OnDragOver	: Event handler for a control being dragged over this control
OnEndDrag	: Notification handler for the end of a dragging operation
OnEnter	: Handler for control receiving the focus
OnExit	: Handler for control loosing the focus; This is a good place for checking the finished user input
OnKeyDown	: Handler for keyboard key pressed
OnKeyPress	: Handler for a character entered by the user
OnKeyUp	: Handler for keyboard key released
OnMouseDown	: Event handler for mouse button going down
OnMouseEnter	: Event handler for mouse entering the area of the control
OnMouseLeave	: Event handler for mouse leaving the area of the control
OnMouseMove	: Event handler for mouse movement within the control
OnMouseUp	: Event handler for mouse button going up
OnStartDrag	: Event handler for the start of a dragging operation
OnStreched	: Handler wenn das Panel ausgeklappt ist, nur aktive wenn DropDownMenu.Active

Public procedures

procedure MouseMove({%H-}Shift: TShiftState; X, Y: Integer);override;

procedure MouseDown({%H-}Button: TMouseButton;{%H-}Shift: TShiftState; X, Y: Integer);override;

procedure MouseUp({%H-}Button: TMouseButton; {%H-}Shift: TShiftState; {%H-}X, {%H-}Y: Integer);override;

procedure LoadFromFile(aFileName: string);

procedure InvalidateBackground;

procedure ParentInputHandler({%H-}Sender: TObject; Msg: Cardinal);

procedure Notification(AComponent: TComponent;Operation: TOperation); override;

constructor Create(AOwner: TComponent); override;

destructor Destroy; override;

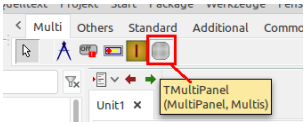
procedure MouseEnter; override;

procedure MouseLeave; override;

procedure Paint; override;

Description

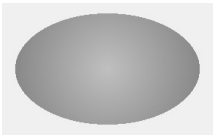
You will find the MultiPanel in the Multis tab.



The shape of the MultiPanel can be influenced with the Style property.

mpsRect:

mpsEllipse:



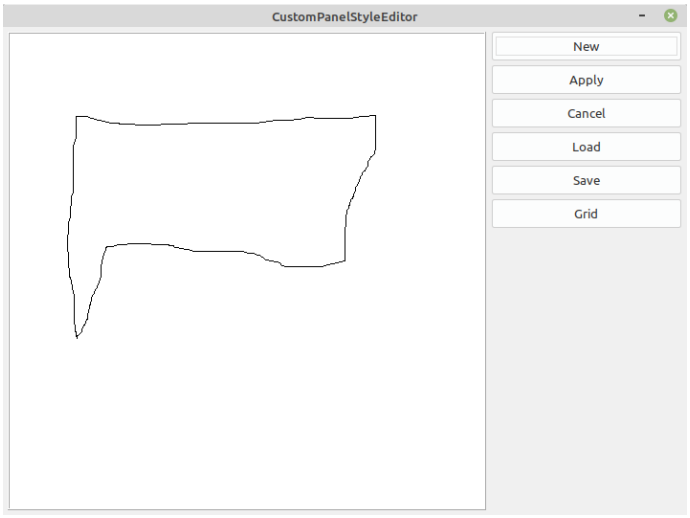
mpsRoundRect:

The corner radius can be set with RndRctRadius . Default setting is 40

mpsCustom:

By default, mpsCustom has a triangle behind it. To draw a custom panel, click on the 3 dots behind DrawACustomPanel .

A property editor will open:



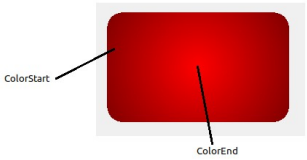
If you now click on New, you can simply draw the shape of the MultiPanel with the mouse.If you click on Use, the MultiPanel shape is adopted. With Discard the MultiPanel shape is not accepted and the editor is closed. With Save you can save a drawn shape and with Load you can get it again. Grid displays an auxiliary grid that may help you when drawing.

Remember it must be set to mpsCustom!

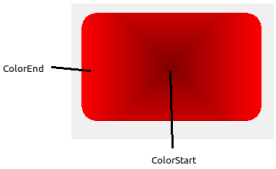
At runtime, MultiPanels saved in advance can also be loaded with LoadFromFile.

To change the colour of the MultiPanel you need the properties ColorStart, ColorEnd and ColorGradient. To get a single-coloured MultiPanel, ColorStart and ColorEnd must be the same. Otherwise, the composition of the three properties determines the appearance.

gcSpread:



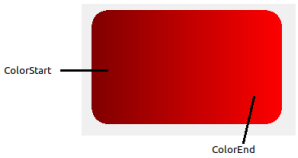
gcRadiant:



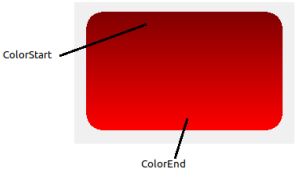
gcAlternate:

Alternately sets one pixel each to Start- and EndColor.

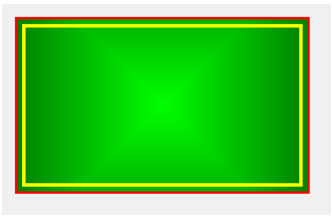
gcHorizontal:

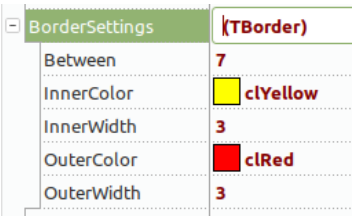


gcVertical:



If you want to highlight the border, you can use the properties of the BorderSettings.





To create a border, simply select a colour. If you do not want a border, use clNone.

BorderSettings.Between : The space between inner- and outerborder
BorderSettings.InnerColor : The color of the innerborder
BorderSettings.InnerWidth : The width of the innerborder
BorderSettings.OuterColor : The color of the outerborder
BorderSettings.OuterWidth : The width of the outerborder

The Appear, Disappear and AnimationSpeed properties can only be set at runtime!
To make an invisible MultiPanel appear, use the property Appear.
Example code:

```
procedure TForm1.MultiButton1Click(Sender: TObject);
begin
 MultiPanel1.Appear:= true;
end;
```

To make a visible MultiPanel disappear, use the property Disappear.
Example code:

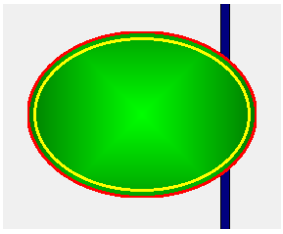
```
procedure TForm1.MultiButton2Click(Sender: TObject);
begin
 MultiPanel1.Disappear:= true;
end;
```

With the property AnimationSpeed the speed of appearance or disappearance can be influenced.
The default value is 0.05. The smaller the value, the slower the animation. With a value of 0.001 it is already very slow.
Example code:

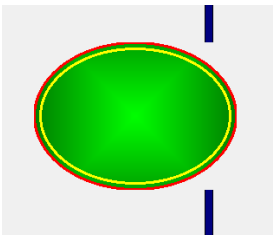
```
MultiPanel1.AnimationSpeed:= 0.001;
```

If you select something other than mpsRect as the geometric shape (Style property), a part of the background of the MultiPanel becomes visible. These visible corners take on the colour set in the parent. If there are self-drawn lines in the parent, for example, these are also shown. This happens because the property ParentAsBkgrd is set to true by default.

ParentAsBkgrd := true



ParentAsBkgrd := false

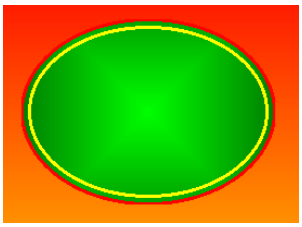


This setting makes sense especially when the parent changes its size. Because then, for example, the drawn line is not scaled correctly here.

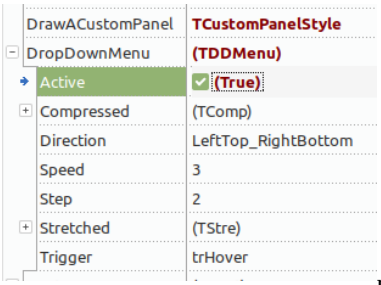
If the parent has a colour gradient, it is possible to compensate for the scaling problem by calling the procedure InvalidateBackground.

Example code:

```
procedure TForm1.FormChangeBounds(Sender: TObject);
begin
 MultiPanel1.InvalidateBackground;
end;
```



To create a DropDown menu (hamburger menu), first set the property DropDownMenu.Active to true.



The MultiPanel now shows the compressed state. The positioning can be done with the mouse or the properties Left or Top (of course anchors can also be set). The size can simply be dragged with the mouse or assigned with the properties DropDownMenu.Compressed.Height or DropDownMenu.Compressed.Width. If the size fits, switch to the expanded state with the properties DropDownMenu.Stretched.Active or DropDownMenu.Compressed.Active. Now the desired size can also be set by dragging with the mouse or with the properties DropDownMenu.Stretched.Height or DropDownMenu.Stretched.Width. It is recommended to place the desired child controls (buttons etc.) in this state. The direction in which the MultiPanel unfolds is determined by the property DropDownMenu.Direction.
The following options are available:
TDirection = (LeftTop_RightBottom,RightTop_LeftBottom,LeftBottom_RightTop,RightBottom_LeftTop)

The speed of the unfolding can be influenced by the properties DropDownMenu.Speed and DropDownMenu.Step . The timer interval with which the unfolding is called is hidden behind Speed. To slow down, increase this value to the desired speed. With Step you can set the number of additional pixels that are drawn per interval. If you want to unfold faster, increase the value for Step.

With DropDownMenu.Trigger you determine the trigger for the unfolding.
The following possibilities are available:
TTrigger = (trClick,trHover,trPinned)
With trClick you have to click in the panel, with trHover it is enough to move the mouse over it. With trPinned, you must click in the MultiPanel, but it only collapses if you click in a definable hotspot (DropDownMenu.Hotspot).

TMultiButton

Properties

Action	: The Action associated with the control
Align	: Specifies the placement of the control inside its Parent
AllowsUp	: Allows a pressed button to be set to not pressed
Anchors	: The set of anchor definitions for this control
AutoSize	: Allows automatic adjustment of the size for the control, according to its content
BidiMode	: Customization (of text controls) in bidirectional reading environments
BorderColor	: The color of the border
BorderSpacing	: Determines the inner and outer border spacing for this control
BorderWidth	: The whidth of the border
Caption	: The text that the user writes in the button
CaptionAlignment	: Alignment of the text in the caption (left, center, right)
CaptionHorMargin	: The horizontal distance of the text in the text rectangle (only effective with taLeftJustify)
CaptionLayout	: Alignment of the text in the caption (top, center, bottom)
CaptionVerMargin	: The vertical distance of the text in the text rectangle (only effective with tlTop)
CaptionWordbreak	: Allows a line break in the caption
ColorEnd	: The end color of the button (for color gradient)
ColorGradient	: The direction of the gradient
ColorStart	: The start color of the button (for color gradient)
Constraints	: The minimum and maximum Width and Height for the control
DisabledAlphaBValue	: The blendvalue at Enable:=false, only at runtime!
DisabledColor	: The colour at Enable:=false, only at runtime!
Down	: The Button has been set in the Down state
DragCursor	: The cursor shape shown while the control is dragged
DragKind	: The operation when the control is dragged - Drag or Dock
DragMode	: Allows the user to drag the control
Enable	: Determines whether the control reacts on mouse or keyboard input
FocusAlphaBValue	: How translucent the focusframe is (0=transparent, 255=opaque)
FocusColor	: The color of the Fokusframe/Foregroundfocus when the Control has the focus
FocusFrameOn	: Indicates when the button has focus
FocusFrameWidth	: The whidth of the focus-frame
Font	: The font to be used for text display in this button
ForegroundFocusOn	: Indicates when the button has focus
GroupIndex	: The Index within the group of MultiButtons
Height	: The vertical size of the control.The height of the MultiButton is minus HoverFrameWidth

HelpContext	: The ID for context-sensitive Help on this control
HelpKeyword	: The keyword for context-sensitive Help on this control
HelpType	: Determines whether context-sensitive Help is selected by numeric ID or keyword
Hint	: The text to show in the Hint window for the control
HoverEndColor	: The endcolor of a hoverevent
HoverFontColor	: The color of the Caption during one hoverevent
HoverImageIndex	: The Index of a Image in a ImageList when during one hoverevent
HoverOn	: Allows to show or hide a hoverevent
HoverStartColor	: The startcolor of a hoverevent
ImageIndex	: The Index of a Image in a ImageList
ImageLeft	: The coordinate of the left edge of a Image
Images	: A list for including images
ImageTop	: The coordinate of the top edge of a Image
ImageWidth	: The unique width of all images in the list
Left	: The client coordinate of the left edge of the control
MessageButton	: A message button to display information or to provide a second integrated button
MessageButton.Alignment	: The position of the messagebutton
MessageButton.BorderColor	: The color of the border
MessageButton.BorderWith	: The whidth of the border
MessageButton.CalculateAlthoughInvisible	: Is required if the MessagButton is only visible at runtime
MessageButton.Caption	: The text that the user writes in the messagebutton
MessageButton.CaptionAlignment	: Alignment of the text in the caption (left, center, right)
MessageButton.CaptionHorMargin	: The horizontal distance of the text in the text rectangle (only effective with taLeftJustify)
MessageButton.CaptionLayout	: Alignment of the text in the caption (top, center, bottom)
MessageButton.CaptionVerMargin	: The vertical distance of the text in the text rectangle (only effective with tlTop)
MessageButton.ColorEnd	: The end color of the messagebutton (for color gradient)
MessageButton.ColorGradient	: The direction of the gradient
MessageButton.ColorStart	: The start color of the messagebutton (for color gradient)
MessageButton.Font	: The font to be used for text display in this button
MessageButton.Height	: The vertical size of the control
MessageButton.HoverOn	: The color of a hoverevent
MessageButton.HoverOn	: Allows to show or hide a hoverevent
MessageButton.ImageIndex	: The Index of a Image in a ImageList
MessageButton.ImageLeft	: The coordinate of the left edge of a Image
MessageButton.Images	: A list for including images
MessageButton.ImageTop	: The coordinate of the top edge of a Image

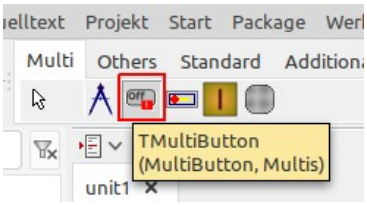
MessageButton.ImageWidth	: The unique width of all images in the list
MessageButton.PositionFactor	: Position factor, only active if alSE,alSW,alNW,alNE,alW,alE,alN, alS,alRightIn,alLeftIn,alTopIn,alBottomIn
MessageButton.PresdColBlendVal	: How translucent the pressedcolor is (0=transparent, 255=opaque)
MessageButton.PressedColor	: The color of the messagebutton when it is pressed
MessageButton.ShowBorder	: Allows to show or hide a border
MessageButton.ShowPressed	: Allows to show or hide the pressedoption
MessageButton.Style	: The geometric shape of the messagebutton
MessageButton.Visible	: Allows to show or hide the control, and all of its children
MessageButton.Width	: The horizontal extent of the control
MultiButton_StyleManager	: Simplifies the design of the MultiButton
PopupMenu	: A context-sensitive menu that pops up when the right mouse button is clicked over this control
PressedEndColor	: The end color of the button when it is pressed (for color gradient)
PressedFontColor	: The color of the text of the caption when the button is pressed
PressedImageIndex	: The Index of a Image in a ImageList when the Button is pressed
PressedStartColor	: The starting color of the button when it is pressed (for color gradient)
RndRctRadius	: Corner diameter if the geometric shape is RoundRect
ShowBorder	: Allows to show or hide a border
ShowHint	: When True, the Hint text is shown when the mouse hovers over the control
ShowMsgButtonInGroup	: Shows the message button on the MultiButton in a group
Style	: The geometric shape of the button
TabOrder	: Determines the sequence of control navigation when the user presses the Tab key
TabStop	: Allows the user to navigate to this control, by pressing the Tab key
Top	: The client coordinate of the top edge of the control
Visible	: Allows to show or hide the control, and all of its children
Width	: The horizontal size of the control.The width of the MultiButton is minus HoverFrameWidth

Events

OnClick	: Notification handler for mouse clicks
OnDragDrop	: This handler determines the action on an drop onto this control, in a drag-drop operation
OnDragOver	: Event handler for a control being dragged over this control
OnEndDrag	: Notification handler for the end of a dragging operation
OnEnter	: Handler for control receiving the focus
OnExit	: Handler for control loosing the focus; This is a good place for checking the finished user input
OnKeyDown	: Handler for keyboard key pressed
OnKeyPress	: Handler for a character entered by the user
OnKeyUp	: Handler for keyboard key released
OnMouseDown	: Event handler for mouse button going down
OnMouseEnter	: Event handler for mouse entering the area of the control
OnMouseLeave	: Event handler for mouse leaving the area of the control
OnMouseMove	: Event handler for mouse movement within the control
OnMouseUp	: Event handler for mouse button going up
OnStartDrag	: Event handler for the start of a dragging operation
MessageButton.OnClick	: Notification handler for mouse clicks
MessageButton.OnMouseMove	: Event handler for mouse movement within the control

Description

You can find the MultiButton in the Multi tab:



It is important to know that the MultiButton is surrounded by a focus frame. As you can see here, the focused MultiButton has an olive green frame. This means that the actual button is smaller around the frame.

The FocusColor property can be used to set the colour of the focus frame. With FocusAlphaBValue the transparency of the focus frame can be controlled. The value 0 means transparent and 255 opaque. FocusFrameWidth determines the thickness of the frame.

Wert 50:

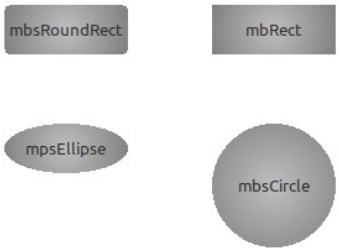
Wert 200:

If FocusFrameOn is set to false, the border is retained but the focus is not shown in colour.

With ForegroundFocusOn, the focused MultiButton has a dotted rectangle. The colour of the rectangle can be influenced with FocusColor. It can be useful here to set FocusFrameWidth to 0 and FocusFrameOn to false so that the corners are not visible!

The Style property is used to set the desired geometric shape of the MultiButton.

If mbsRoundRect is set, the RndRctRadius property can be used to set the diameter of the corner rounding.



If you want to add a coloured border to the MultiButton, set ShowBorder to true. The colour of the border is set with BorderColor and the width with BorderWidth.

To change the colour of the MultiButton you need the properties ColorStart, ColorEnd and ColorGradient. To get a single-coloured MultiButton, ColorStart and ColorEnd must be the same. Otherwise, the composition of the three properties determines the appearance.

Alternate alternately sets one pixel each to Start- and EndColor.

Here is StartColor clGreen and EndColor clYellow.

By default, HoverOn is set to true. This means that when a hover event occurs (the mouse moves over the MultiButton) the appearance can be changed as desired with HoverStartColor, HoverEndColor, HoverFontColor and HoverImageIndex. If you do not want this, set HoverOn to false.

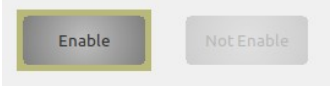
HoverEndColor	: The endcolor of a hoverevent
HoverFontColor	: The color of the Caption during one hoverevent
HoverImageIndex	: The Index of a Image in a ImageList when during one hoverevent
HoverOn	: Allows to show or hide a hoverevent
HoverStartColor	: The startcolor of a hoverevent

When the MultiButton is pressed, the properties PressedStartColor, PressedEndColor, PressedFontColor and PressedImageIndex influence the appearance. If you do not want any changes when the button is pressed, the only thing left to do is to set the same settings such as ColorStart etc..

PressedEndColor	: The end color of the button when it is pressed (for color gradient)
PressedFontColor	: The color of the text of the caption when the button is pressed
PressedImageIndex	: The Index of a Image in a ImageList when the Button is pressed
PressedStartColor	: The starting color of the button when it is pressed (for color gradient)

The Enable property determines whether the control reacts to mouse or keyboard input. The appearance when not enabled can be influenced at runtime with DisabledAlphaBValue and DisabledColor.

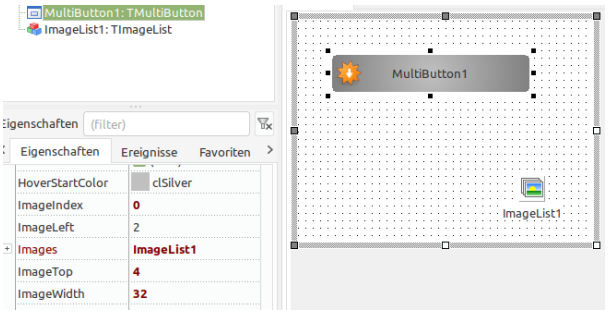
If you do not want to see a frame, set DisabledColor to the same colour as the parent.



If necessary, use MultiButton2.DisabledColor:=GetColorResolvingParent instead of clDefault.

If you want to insert an image, you must first drag an ImageList onto the form. You then assign the desired images in the desired sizes to this ImageList. The operation of the ImageListEditor is described very well here: https://www.lazarusforum.de/viewtopic.php?f=18&t=13170

With Images you enter the image list on the form. With ImageIndex you can select the desired image from the ImageList, where -1 means no image. With ImageLeft and ImageTop you determine the position of the image. With ImageWidth you can scale the size of the image. It is recommended to scale only smaller.



Tip:

If you use HighDPI under Windows, the images and the MultiButton are scaled. In order for it to work at runtime, I had to select Vista-8:an,8.1+:pro Monitor(True/PM) in the project settings for DPI adjustment.

The AllowsUp property turns the button into a kind of switch. This means that when the button is pressed, it remains pressed until it is pressed again. If you want the MultiButton to appear pressed at the beginning of the programme, you do this with the property Down.

If the button is pressed, it is displayed with the properties Pressed....!

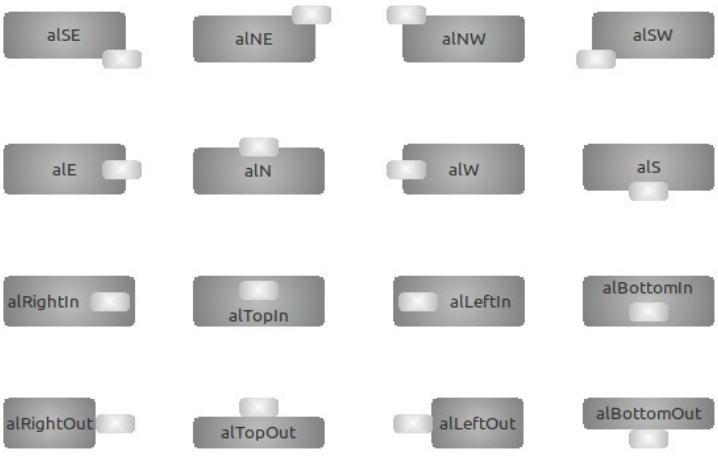
If needed, the MultiButton can belong to a group. This is achieved with the property GroupIndex. If a MultiButton has a value other than 0, it belongs to the group with the same value. Only one MultiButton can be pressed in a group. If a button in the group should already be pressed when the programme is started, this can be achieved with the property Down.


```
MultiButton1.GroupIndex:=1;
MultiButton2.GroupIndex:=1;
MultiButton3.GroupIndex:=1;  MultiButton3.Down:=true;
```

If you want to make the pressed button visually clearer, you can still use the property ShowMsgButtonInGroup. The last pressed button gets a MessageButton.

The MessageButton

To use the integrated MessageButton, you must first set MessageButton.Visible to true. Then you can set the position of the MessageButton with MessageButton.Alignment.



The MessageButton.PositionFactor property can be used to influence the position of the MessageButton somewhat. However, only with alSE,alSW,alNW,alNE,alW,alE,alN,alS,alRightIn,alLeftIn,alTopIn,alBottomIn

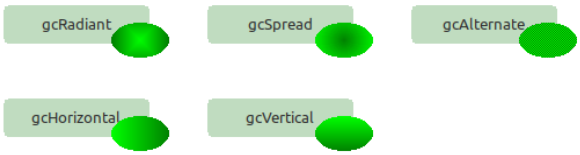
If you want to change the shape of the MessageButton, you can do this with the property MessageButton.Style.

If you place several MultiButtons in a row and the MessageButton is not visible for all of them, the MultiButtons have different sizes. Here no. 2 appears larger:

To get around this there is the property MessageButton.CalculateAlthoughInvisible. If you set this property to true for No. 2, it looks like this:

If you want to provide the MessageButton with a coloured border, set MessageButton.ShowBorder to true. The colour of the border is set with MessageButton.BorderColor and the width with MessageButton.BorderWith.

To change the colour of the MessageButton you need the properties MessageButton.ColorStart, MessageButton.ColorEnd and MessageButton.ColorGradient. To get a single-coloured MessageButton, ColorStart and ColorEnd must be the same. Otherwise, the composition of the three properties determines the appearance.



Alternate alternately sets one pixel each to Start- and EndColor.

Here StartColor is clLime and EndColor is clGreen.

By default, MessageButton.HoverOn is set to true. This means that when a hover event occurs (the mouse moves over the MessageButton) a border is drawn around the MessageButton. The colour of the border can be set with MessageButton.HoverColor. If you do not want this, set HoverOn to false.

MessageButton. BorderWidth influences the thickness of the hover border!

If the MessageButton is pressed and MessageButton.ShowPressed is true, then the colour set in MessageButton.PressedColor with the value stored in MessageButton.PresdColBlendVal is faded over the MessageButton. Where 0 means transparent and 255 means opaque.