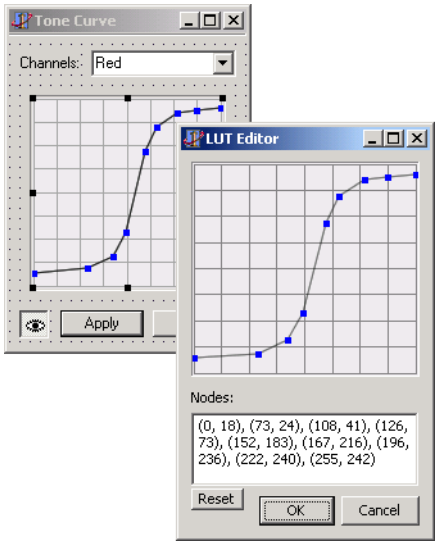
LUTControl

Delphi VCL Component
Version 1.3
September 9, 2000

Alex Denissov

Overview

LUTControl is a VCL component which acts similar to the tone curve found in many image processing programs:



The current version does not support any interpolation of values between nodes except the linear interpolation.

Note: LUTControl requires the *Graphics32* library v0.97 or later to be installed.

The latest version of the *LUTControl* component as well as the *Graphics32* library may be found on the web site:

http://www.geocities.com/den_alex

License

Installation

Delphi 5 is required in order to install the *LUTControl* component.

- Unzip the files.
- Select **File** | **Open**... on the menu bar. Set **Files of type** to **Delphi package source**, locate and select the **LUT_Install.dpk** file, and click **Open**.
- Check the necessary file paths in **Tools** | **Environment Options** | **Library** | **Library Path**. They should include LUTControl's directory as well as **$(DELPHI)\Source\Toolsapi** and **$(DELPHI)\Source\Vcl**.
- A package editor window will appear. Click **Compile**, then click **Install**.

Overview

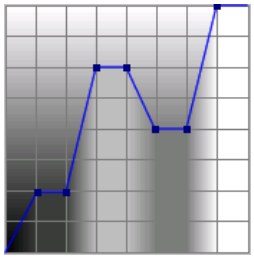
The most common use of *LUTControl* is the generation of lookup tables in image processing applications, for example for brightness and contrast regulation.

It provides generation of 256-element lookup tables where the value of each element varies in 0...255 range.

Two modes of operation are currently supported: *lmNodes* and *lmCustom*.

- In *lmNodes* mode *LUTControl* has several nodes which are draggable in run time. User may add new nodes by clicking the left mouse button in the empty area. Nodes may be removed with right mouse button;
- *lmCustom* mode allows changing the curve just by dragging the mouse.

In addition to a normal behavior, *LUTControl* can optionally display a background image which may be changed in real time to make the user interaction more intuitive.



TBitmap32 and *TColor32* types mentioned here are the parts of *Graphics32* library, which may be downloaded from:

http://www.geocities.com/den_alex/graph32.html

Properties

BackgndImage

property BackgndImage: TBitmap32;

This property specifies a bitmap used as background image. The bitmap is automatically stretched to fit the component's boundary minus edge value. When the bitmap is empty, the control paints its background using the standard *Color* property.

SEE ALSO: *Edge*.

DefaultLeftEdge

property DefaultLeftEdge: Byte;

When there is no node lying on the left edge, this property specifies the value the curve will be interpolated to.

SEE ALSO: *DefaultRightEdge*

DefaultRightEdge

property DefaultRightEdge: Byte;

Specifies the default right edge's default value, which is used for interpolation as long as there is no node with $x = 255$ coordinate.

SEE ALSO: *DefaultLeftEdge*

DoubleBuffered

property DoubleBuffered: Boolean; *// inherited from TCustomComponent*

Set the *DoubleBuffered* property if your application resizes *LUTControl*. It will help to avoid flicker when resizing.

Edge **property** Edge: Integer;

Set *Edge* greater or equal to *HandleSize* if you do not want the node hanles on the edges to be cropped by the control boundary.

SEE ALSO: *HandleSize*.

GridColor **property** GridColor: TColor32;

Specifies the color of the grid. The grid will be semi-transparent if *GridColor*'s alpha is less than $FF.

SEE ALSO: *GridCols*, *GridRows*.

GridCols **property** GridCols: Integer;

A number of columns in the grid. Set *GridCols* to 0 to hide vertical grid lines.

SEE ALSO: *GridRows*.

GridRows **property** GridRows: Integer;

A number of rows in the grid. Set *GridRows* to 0 to hide horizontal grid lines.

Mode **property** Mode: TLUTMode;
type TLUTMode = (lmNodes, lmCustom);

Mode specifies if the control has draggable nodes, or some arbitrary curve. When switchin to custom mode, the curve shape is preserved. Switching back from *lmCustom* to *lmNodes* will recalculate the curve according to *Nodes* property.

SEE ALSO: *Nodes*.

NodeCount **property** NodeCount: Integer;

Returns the number of nodes. Although it is possible to set *NodeCount*, it is recommended to use a much safer *Nodes* property to set the number of nodes and their coordinates. Access *NodeCount* for writing only when it is unavoidable, enclose the writing into *BeginUpdate*...*EndUpdate* block and set the consistent points coordinates immediately after changing *NodeCount*, inside the same *BeginUpdate*...*EndUpdate* block.

SEE ALSO: *BeginUpdate*, *EndUpdate*, *Nodes*.

LineColor **property** LineColor: TColor32;

Specifies a color of the curve. May contain transparency.

NodeColor **property** NodeColor: TColor32;

Specifies a color of nodes. May contain transparency.

NodePoints **property** NodePoints: TArrayOfPoints;
type TArrayOfPoints = **array of** TPoint;

Use *NodePoints* to access coordinates of nodes.

Nodes **property** Nodes: TNodeString;
type TNodeString = **type string**;

Use *Nodes* to access coordinates of nodes as a string. The string is constructed as a set of comma separated integer numbers with optional spaces. Each pair of coordinates is enclosed in brackets:

```
LUTControl.Nodes := '(0, 0), (60, 80), (140,160)';
```

this call is analogous to the following code:

```
LUTControl.BeginUpdate;
LUTControl.NodeCount := 3;
LUTControl.NodePoints[0] := Point(0, 0);
LUTControl.NodePoints[1] := Point(60, 80);
```

```
LUTControl.NodePoints[2] := Point(140, 160);
LUTControl.EndUpdate;
```

Usage of the *Nodes* property should be considered as a much safer way to rearrange the nodes. It will sort the nodes automatically, reject invalid ones (for example there can be no two points with the same *x* coordinate) and verify that they lye in [0..255] range. If the control is in lmCustom mode, writing nodes will have no effect on the curve.

SEE ALSO: *Mode*.

HandleSize **property** NodeSize: Integer;

Specifies the displayed size of the node. The actual displayed size is *NodeSize* * 2 + 1 pixels.

Methods

BeginUpdate **procedure** BeginUpdate;

Temporarily prevents the generation of *OnChange* events. The method must be paired with *EndUpdate* calls and may safely be nested. This method should be used, for example, when changing the *NodeCount* property.

SEE ALSO: *EndUpdate*, *OnChange*.

CopyTo
```
procedure CopyTo(out Destination: TLUT256);
type TLUT256 = array [0..255] of Integer;
```

CopyTo reads the lookup table into 256-element array. Data between nodes is linearly interpolated.

EndUpdate **procedure** EndUpdate;

EndUpdate call re-enables generation of *OnChange* events, but only if it is a last *EndUpdate* call in the nested *BeginUpdate...EndUpdate* structure.

SEE ALSO: *BeginUpdate*, *OnChange*.

GetFrom
```
procedure GetFrom(const Source: TLUT256);
type TLUT256 = array [0..255] of Integer;
```

Call the *GetFrom* procedure to set the curve from external data. The control is automatically switched to *lmCustom* mode.

SEE ALSO: *Mode*.

Events

OnCustomPaint
```
property OnCustomPaint: TCustomPaintEvent;
type TCustomPaintEvent = procedure(
  Sender: TObject;
  Bitmap: TBitmap32;
  var DrawDefault: Boolean
) of object;
```

This event is called every time the control is painted. It allows you to customize its appearance. Change *DrawDefault* to *False*, if you do not want the control to proceed with drawing of grid, curve and node points.

OnChange **property** OnChange: TNotifyEvent;

OnChange is called every time the node arrangement changes.