

Lesson 53: TImageList

Lazarus Pascal Beginners Series — Common Controls tab

What is TImageList?

TImageList is a container that holds a collection of same-sized bitmaps. Instead of each control storing its own icon, they all point to a shared TImageList and reference images by index. This means one set of icons can serve an entire application — TToolBar, TTreeView, TListView, TPageControl, THeaderControl and any other control that accepts an Images property.

TImageList is found on the Common Controls tab and belongs to the ImgList unit.

The Golden Rules

There are three rules that must always be followed:

- Set Width and Height BEFORE adding any images. Once images are added the size cannot change.
- All images must be exactly Width × Height pixels. Images of a different size are scaled.
- Add returns the new zero-based index. Store this index on the control: `ImageIndex := Add(Bmp, nil)`.

Key Methods

Method	Description
<code>Add (Bmp, Mask)</code>	Add a TBitmap with optional mask. Returns the new zero-based index.
<code>Insert (Idx, Bmp, Mask)</code>	Insert a bitmap at a specific position. All higher indices shift up.
<code>Delete (Idx)</code>	Remove the image at index. All higher indices shift down.
<code>Replace (Idx, Bmp, Mask)</code>	Swap the image at index without changing the index of any other image.
<code>Move (From, To)</code>	Reorder images. Moves the image at From to the To position.

Method	Description
Clear	Remove all images.
Draw(Canvas,X,Y,Idx)	Paint the image at index onto any TCanvas at position X,Y.
GetBitmap(Idx, Bmp)	Copy the image at index into an existing TBitmap.
Count	Read-only number of images currently in the list.

Key Properties

Property	Description
Width / Height	Size of every image. Set before adding images.
BkColor	Background colour. clNone = transparent mask applied.
Masked	When True the mask bitmap is used for transparency.

Loading Icons from the Lazarus IDE Images Folder

This lesson uses the same FindImagesFolder pattern as Lessons 47 and 48. Icons are loaded from C:\lazarus\images\general_purpose\ using TPortableNetworkGraphic. A drawn fallback is used if the folder is not found:

```
PNG := TPortableNetworkGraphic.Create;
try
  PNG.LoadFromFile(FImagesPath + 'File_01_24.png');
  Idx := ilDrawn.Add(PNG, nil);
  { Idx is the index to assign to buttons/nodes/tabs }
finally
  PNG.Free;
end;
```

Icons loaded for ilDrawn (24×24px): File, Open, Save, Cut, Copy, Paste, Undo, Print.
Icons loaded for ilControls (16×16px): File, Open, Save, Cut, Paste, Search, Undo.

Demo 1 — Building an Image List

Demo 1 shows ilDrawn being built at startup and displayed as a preview strip. The preview is a TPaintBox whose OnPaint handler calls Draw for every image:

```
procedure TForm1.DrawListPreview(APB: TPaintBox; AIL: TImageList);
var I, X: Integer;
begin
```

```

X := 4;
for I := 0 to AIL.Count - 1 do
begin
    AIL.Draw(APB.Canvas, X, 4, I);
    APB.Canvas.TextOut(X, AIL.Height + 6, IntToStr(I));
    X := X + AIL.Width + 6;
end;
end;

```

The "Add circle" button adds a new coloured circle bitmap to `ilDrawn` at runtime. "Load PNG..." opens a file dialog to load any PNG. "Reset list" rebuilds the original 8 icons from the Lazarus images folder.

► **Try it: Click "Add circle" three times — three new circles appear in the preview strip with their indices. Click "Load PNG..." and pick any PNG from the Lazarus images folder — it is added at the next index. Click "Reset list" to restore the original 8.**

Demo 2 — Sharing One List Across Three Controls

`ilControls` is a single `TImageList` assigned simultaneously to a `TToolBar`, a `TTreeView` and a `TPageControl`. All three use the same icons without any duplication:

```

tbControls.Images := ilControls;
tvControls.Images := ilControls;
pcControls.Images := ilControls;

```

Each control picks its icon by index:

```

{ TToolButton }
Btn.ImageIndex := 0; { File_01_16.png }

{ TTreeNode }
N.ImageIndex := 1; { Open_01_16.png - normal state }
N.SelectedIndex := 2; { Save_01_16.png - selected state }

{ TTabSheet }
tsCtrl1.ImageIndex := 0;
tsCtrl2.ImageIndex := 1;
tsCtrl3.ImageIndex := 2;

```

Functional toolbar buttons

All seven toolbar buttons are wired to a shared `OnClick` handler that dispatches by `Caption` to real actions on the `TMemo` document below:

```

procedure TForm1.tbControlsButtonClick(Sender: TObject);

```

```

var Btn: TToolButton;
begin
  Btn := TToolButton(Sender);
  if      Btn.Caption = 'New'      then DoNew
  else if Btn.Caption = 'Open'    then DoOpen
  else if Btn.Caption = 'Save'    then DoSave
  else if Btn.Caption = 'Cut'     then DoCut
  else if Btn.Caption = 'Paste'   then DoPaste
  else if Btn.Caption = 'Search'  then DoSearch
  else if Btn.Caption = 'Undo'    then DoUndo;
end;

```

The Search action uses InputBox to ask for text and then selects the match in the memo:

```

Srch := InputBox('Search', 'Find text:', '');
Pos  := System.Pos(Srch, memoDoc.Text);
if Pos > 0 then
begin
  memoDoc.SelStart := Pos - 1;
  memoDoc.SelLength := Length(Srch);
end;

```

► **Try it:** Type some text in the memo. Select a word and click Cut. Click Paste to restore it. Click Search, type part of the text — it is highlighted in the memo. Click New — it warns about unsaved changes. Notice the TTreeView and TPageControl tabs share the same icons as the toolbar.

Demo 3 — Manual Drawing

ImageList.Draw paints one image from the list onto any TCanvas at any position. This demo uses three TTrackBars to control which image to draw and where:

```

ilDrawn.Draw(pbManual.Canvas,
             trkDrawX.Position,
             trkDrawY.Position,
             trkDrawIdx.Position);

```

The TPaintBox OnPaint handler is called whenever Invalidate is triggered. Each TTrackBar OnChange calls pbManual.Invalidate which queues a repaint:

```

procedure TForm1.trkDrawIdxChange(Sender: TObject);
begin
  pbManual.Invalidate;
end;

```

► **Try it:** Drag the Index slider to pick different icons. Drag X and Y to move the icon around the white canvas. The index label updates to show the current position and index.

Demo 4 — Operations

Four operations on `ilDrawn` demonstrated live with a preview strip that updates after each action.

Operation	Code
<code>GetBitmap</code>	<code>ilDrawn.GetBitmap(Idx, Bmp)</code> — copies image to an existing <code>TBitmap</code> . Useful for displaying a single icon elsewhere.
<code>Delete</code>	<code>ilDrawn.Delete(Idx)</code> — removes the image. All higher indices shift down by 1.
<code>Move left/right</code>	<code>ilDrawn.Move(Idx, Idx-1)</code> — swaps adjacent images. All controls using the list update automatically.
<code>Replace</code>	<code>ilDrawn.Replace(Idx, NewBmp, nil)</code> — swaps the bitmap at index without affecting any other index.

When you Delete or Move an image, every control that uses this `TImageList` automatically updates its display. This is the power of the shared list pattern — one change propagates everywhere.

► **Try it:** Type 2 in the index box and click Delete — index 2 disappears and all higher indices shift down. Type 0 and click Move right — index 0 and 1 swap. Type 1 and click Replace — a coloured rectangle appears at that index.

TScrollBox Portability

All panels are wrapped in a `TScrollBox` (`Align=alClient`, `AutoScroll=True`, `BorderStyle=bsNone`). `Application.Scaled := True` in the `.lpr` and `Scaled=True` in the `.lpi` ensure correct DPI scaling. No Windows-specific code is used. The icon paths include Linux fallbacks.

Project Files

File	Description
<code>Unit1.pas</code>	Main form source. <code>FindImagesFolder</code> , <code>BuildDrawnList</code> , <code>BuildControlsList</code> , all action handlers.

File	Description
Unit1.lfm	Form layout. Three TImageList objects are direct children of the form.
Lesson53_TImageList.lpr	Program entry point. Sets Application.Scaled := True.
Lesson53_TImageList.lpi	Project file. Sets Scaled=True for DPI awareness.

Notes and Cross-Platform Behaviour

- **ImgList unit:** TImageList is declared in ImgList. Always add ImgList to the uses clause.
- **Set Width and Height first:** This is the most common mistake. If you add images before setting the size, all images are stored at the default 16x16 regardless of their actual size.
- **Add returns the index:** Always capture the return value of Add to know which index was assigned.
- **Delete shifts indices:** After Delete(2), what was index 3 is now index 2. Update any stored ImageIndex values accordingly.
- **Replace does not shift:** Replace(2, Bmp, nil) swaps the image but leaves all indices unchanged. Prefer Replace over Delete+Insert when you just want to change an icon.
- **Shared list auto-updates:** When you Add, Delete, Replace or Move, every control using the list updates its display automatically.
- **PNG loading:** Use TPortableNetworkGraphic.Create, LoadFromFile, Add, then Free. Never assign the PNG directly to Width/Height of the image list.
- **Cross-platform:** This lesson uses only standard LCL units. Compiles and runs unchanged on Windows, Linux and macOS

