

Lesson 42: TTrackBar

Lazarus Pascal Beginners Series — Common Controls tab

What is TTrackBar?

TTrackBar is a slider control that lets the user select a numeric value within a defined range by dragging a thumb along a track. It can be oriented horizontally or vertically and supports tick marks, keyboard navigation and a visual selection range highlight.

TTrackBar is found on the Common Controls tab of the Component Palette. It belongs to the ComCtrls unit — add ComCtrls to the uses clause whenever you use TTrackBar.

The most important thing to know: TTrackBar stores a single integer value in Position. You read this value in the OnChange event, which fires every time the user moves the thumb — whether by dragging, clicking the track, or pressing a key. That single event handler is all you need for most use cases.

Key Properties

Property	Description
Min	Minimum value of the range. Default 0.
Max	Maximum value of the range. Default 10.
Position	Current integer value. Read this in OnChange. Can be set in code to move the thumb.
Orientation	trHorizontal (default) — left/right slider. trVertical — up/down slider.
TickStyle	tsAuto — ticks drawn at every Frequency interval (default). tsManual — you place ticks with SetTick. tsNone — no ticks.
TickMarks	tmBottomRight — ticks below (H) or right (V). tmTopLeft — ticks above or left. tmBoth — ticks on both sides.
Frequency	Spacing between tick marks in value units. Default 1. Set to 10 for a 0..100 range to get 11 ticks.
LineSize	How many units Position moves per arrow key press. Default 1.
PageSize	How many units Position moves per Page Up/Down key.

Property	Description
	Default 2.
Reversed	When True, the track runs right-to-left (H) or bottom-to-top (V). Default False.
ScalePos	Where scale labels appear: trTop, trBottom, trLeft, trRight or trNone.
SelStart	Start value of the highlighted selection range.
SelEnd	End value of the highlighted selection range.
ShowSelRange	When True, the range between SelStart and SelEnd is highlighted on the track.

Key Events

Event	Description
OnChange	Fires whenever Position changes — drag, click, or keyboard. This is the only event needed for most uses.

Key Methods

Method	Description
SetParams (Position, Min, Max)	Sets Position, Min and Max in one call. Use this instead of setting them separately to avoid FixParams being called three times.
SetTick (Value)	Places a tick mark at a specific value. Only used when TickStyle = tsManual.

Demo 1 — Basic: Sliders Controlling a Shape

This demo uses four TTrackBar sliders to control a TShape in real time. One slider sets the size of the shape (20 to 200 pixels). The other three set the Red, Green and Blue colour channels (0 to 255 each). As you drag any slider the shape updates instantly.

The demo shows the most fundamental TTrackBar pattern: reading Position in OnChange and using the value to drive something else on screen.

Setup

In FormCreate the four sliders are configured with their ranges and starting positions:

```
trkSize.Min      := 20;
trkSize.Max      := 200;
trkSize.Position := 80;
trkSize.Frequency := 20; { tick every 20 units }

trkRed.Min  := 0; trkRed.Max  := 255; trkRed.Position := 41;
trkGreen.Min := 0; trkGreen.Max := 255; trkGreen.Position := 128;
trkBlue.Min  := 0; trkBlue.Max := 255; trkBlue.Position := 185;
```

All four OnChange handlers call the same private procedure UpdateShape, which reads all four positions and applies them at once. This avoids duplicating the same logic in four separate procedures:

```
procedure TForm1.trkSizeChange(Sender: TObject);
begin
    UpdateShape;
end;

procedure TForm1.UpdateShape;
var
    R, G, B : Byte;
    Size    : Integer;
begin
    R := trkRed.Position;
    G := trkGreen.Position;
    B := trkBlue.Position;
    Size := trkSize.Position;

    shpDemo.Width      := Size;
    shpDemo.Height     := Size;
    shpDemo.Brush.Color := RGBToColor(R, G, B);

    lblSizeVal.Caption := 'Size: ' + IntToStr(Size) + 'px';
    lblRedVal.Caption  := 'R: '   + IntToStr(R);
    lblGreenVal.Caption := 'G: '   + IntToStr(G);
    lblBlueVal.Caption  := 'B: '   + IntToStr(B);
end;
```

RGBToColor(R, G, B) is the standard LCL function for building a TColor from three byte values. It is in the Graphics unit.

i The colour label captions are coloured red, green and blue via Font.Color in the lfm, giving an instant visual cue for each channel.

► Try it: Drag the Size slider left to make the shape smaller and right to make it larger. Then drag all three colour sliders to mix any colour. Drag Red to 255 and Green and Blue to 0 for pure red. Try R=0, G=255, B=0 for pure green.

Demo 2 — Orientation: Horizontal and Vertical

This demo shows both orientations side by side. `trkHoriz` uses the default `trHorizontal` orientation. `trkVert` uses `trVertical`, set in `FormCreate`:

```
trkVert.Orientation := trVertical;
```

The Orientation can also be set at design time in the Object Inspector. Both sliders run from 0 to 100 and display their current value in a label via `OnChange`:

```
procedure TForm1.trkHorizChange(Sender: TObject);
begin
    lblHorizVal.Caption := IntToStr(trkHoriz.Position);
    Log('trkHoriz.Position = ' + IntToStr(trkHoriz.Position));
end;
```

For a vertical slider, Position 0 is at the top and Position Max is at the bottom by default. Setting `Reversed := True` flips this so 0 is at the bottom — more natural for sliders representing volume or height.

► Try it: Drag `trkHoriz` left and right. Drag `trkVert` up and down. Notice the labels update live. Check the event log to see every position change recorded.

⚠ When switching Orientation at runtime the control repaints but its Width and Height stay the same. Swap Width and Height in code if needed to keep the visual proportions correct.

Demo 3 — Tick Styles and Marks

This demo shows three `TTrackBar` controls side by side, each with different tick settings. The tick settings are applied in the `lfm` at design time so you can see them immediately when the application starts.

trkTickAuto — tsAuto + tmBottomRight

This is the default combination. `TickStyle=tsAuto` means the LCL draws tick marks automatically at every Frequency interval. `TickMarks=tmBottomRight` means the ticks appear below the track (for a horizontal slider). Frequency is set to 10 on a 0..100 range, giving 11 ticks:

```
trkTickAuto.TickStyle := tsAuto;
trkTickAuto.TickMarks := tmBottomRight;
trkTickAuto.Frequency := 10;
```

trkTickNone — tsNone

TickStyle=tsNone removes all tick marks entirely, giving a clean minimal slider. Use this when tick marks would clutter the interface or when the slider is very small:

```
trkTickNone.TickStyle := tsNone;
```

trkTickBoth — tsAuto + tmBoth

TickMarks=tmBoth draws ticks on both sides of the track — both above and below for a horizontal slider. This is useful when you want a more prominent visual scale:

```
trkTickBoth.TickStyle := tsAuto;
trkTickBoth.TickMarks := tmBoth;
trkTickBoth.Frequency := 10;
```

► **Try it: Drag all three sliders. Notice how tsAuto shows ticks below, tsNone is clean with no ticks, and tmBoth shows ticks on both sides making the scale more prominent.**

! TickStyle=tsManual lets you place ticks at specific values using SetTick(Value). For example, you could mark only the values 0, 33, 66 and 100 on a 0..100 range regardless of Frequency.

Demo 4 — Range, Frequency and Page Size

This demo lets you change the Min, Max, Frequency and PageSize of trkRange at runtime using edit boxes and an Apply button. This shows how these properties affect both the visual appearance and the keyboard behaviour of the slider.

Applying new settings

The Apply button handler reads the four edit boxes, validates them and applies them to trkRange:

```
Mn := StrToIntDef(edtMin.Text, trkRange.Min);
Mx := StrToIntDef(edtMax.Text, trkRange.Max);
Fr := StrToIntDef(edtFreq.Text, trkRange.Frequency);
Pg := StrToIntDef(edtPage.Text, trkRange.PageSize);

if Mx <= Mn then
```

```

begin
    ShowMessage('Max must be greater than Min.');
```

```

    Exit;
end;

trkRange.SetParams(trkRange.Position, Mn, Mx);
trkRange.Frequency := Fr;
trkRange.PageSize  := Pg;
```

SetParams(Position, Min, Max) is used instead of setting Min and Max separately. This is important because each individual assignment triggers an internal FixParams call that clamps Position to the new range. Using SetParams applies all three values together in one operation.

► **Try it: Set Min=0, Max=10, Frequency=1 and press Apply. Notice the ticks are very dense. Then try Min=0, Max=1000, Frequency=100. Now try pressing the right arrow key to step by LineSize (1), and Page Down to jump by PageSize (10). Change PageSize to 100 and try again.**

⚠ **Setting Min and Max as separate assignments can cause unexpected behaviour if Position is outside the new range during the transition. Always use SetParams when changing both Min and Max together.**

Demo 5 — Selection Range

TTrackBar can highlight a range on the track between two values using SelStart, SelEnd and ShowSelRange. This is useful for showing a valid range, a target zone, or the output range of a mapped value.

In this demo trkSel starts with SelStart=25 and SelEnd=75, marking the middle half of the 0..100 range. The highlight is the coloured band drawn between those two positions on the track.

Setting the selection range

```

trkSel.SelStart    := 25;
trkSel.SelEnd      := 75;
trkSel.ShowSelRange := True;
```

The Apply button lets you change SelStart and SelEnd at runtime. Values are clamped to the track range:

```

S := StrToIntDef(edtSelStart.Text, trkSel.SelStart);
E := StrToIntDef(edtSelEnd.Text,   trkSel.SelEnd);
if S < trkSel.Min then S := trkSel.Min;
if E > trkSel.Max then E := trkSel.Max;
```

```
trkSel.SelStart := S;  
trkSel.SelEnd   := E;
```

The ShowSelRange checkbox toggles the highlight on and off:

```
procedure TForm1.chkShowSelChange(Sender: TObject);  
begin  
    trkSel.ShowSelRange := chkShowSel.Checked;  
end;
```

► **Try it: Drag trkSel and watch the thumb move over the highlighted band. Uncheck ShowSelRange to hide the band. Then set SelStart=0, SelEnd=50 and press Apply to shift the highlighted zone to the left half. Try SelStart=80, SelEnd=100 for a narrow zone at the right end.**

i The selection range is a visual indicator only. TTrackBar does not restrict Position to the selected range. The thumb moves freely across the full Min..Max range regardless of SelStart and SelEnd.

Panel 7 — Event Log

The event log at the bottom records all OnChange events from Demo 2 (orientation) and all Apply button presses from Demo 4 and Demo 5. Each entry is timestamped.

Demo 1 does not log to avoid flooding the log with every pixel of a drag. In a real application you would typically only log on significant events or when the user releases the thumb, not on every intermediate position during a drag.

The log uses SelStart and SelLength to auto-scroll to the bottom after each new entry, which works on all platforms without requiring SendMessage:

```
memoLog.Lines.Add(FormatDateTime('[hh:nn:ss] ', Now) + AMsg);  
memoLog.SelStart := Length(memoLog.Text);  
memoLog.SelLength := 0;
```

Companion: TTrackBar Visual Demo and Installer

This lesson comes with two companion executables that show TTrackBar in a real working application:

TTrackBar_Demo.exe

A polished standalone demo with four tabs showing TTrackBar in real-world scenarios:

- RGB Colour Mixer — three horizontal sliders (0–255) mix any colour live, updating a swatch and a gradient preview.
- Audio Equaliser — five vertical sliders (-12 to +12 dB) simulate a 5-band EQ with a live curve visualization. Flat, Bass boost and Treble boost presets.
- Image Adjustments — four centred sliders (Brightness, Contrast, Saturation, Warmth, -100 to +100) with a live colour grid preview.
- Zoom and Rotation — Zoom (10–400%) and Rotation (0–360°) sliders drive a live rotating rectangle drawn on a canvas.

TTrackBar_Installer.exe

A self-contained installer that copies TTrackBar_Demo.exe to the user's AppData folder and launches it. The installer uses both TProgressBar and TTrackBar together to show real copy progress:

```
procedure TForm1.tmrCopyTimer(Sender: TObject);
begin
    Written := FInStream.Read(Buf, ToWrite);
    FOutStream.WriteBuffer(Buf, Written);
    FWritten := FWritten + Written;
    Pct := Round(FWritten * 100 / FTotalSize);
    pbProgress.Position := Pct;    { TProgressBar }
    trkProgress.Position := Pct;   { TTrackBar    }
end;
```

The file is read in 32KB chunks via a TTimer firing every 30ms. This keeps the UI responsive during the copy — the user sees smooth real-time progress rather than the application freezing. The TTrackBar and TProgressBar both track the same percentage value, demonstrating both controls side by side in a practical context.

Distribution: ship both files in a zip together. The installer finds TTrackBar_Demo.exe in the same folder using ExtractFilePath(ParamStr(0)).

i The installer is Windows-only (uses %APPDATA% and OpenDocument to launch). The demo itself is cross-platform.

TScrollBox Portability

All panels are wrapped in a TScrollBox (Align=alClient, AutoScroll=True, BorderStyle=bsNone). This allows the form to work correctly on screens where the content is taller than the window, without needing to resize the form. The project sets Application.Scaled := True in the .lpr and Scaled=True in the .lpi for correct DPI scaling on all platforms.

Project Files

File	Description
Unit1.pas	Main form source. All demo logic, event handlers and the UpdateShape helper.
Unit1.lfm	Form layout. Contains all TTrackBar design-time property settings.
Lesson42_TTrackBar.lpr	Program entry point. Sets Application.Scaled := True.
Lesson42_TTrackBar.lpi	Project file. Sets Scaled=True for DPI awareness.

Notes and Cross-Platform Behaviour

- ComCtrls unit: TTrackBar is declared in the ComCtrls unit. Always add ComCtrls to the uses clause.
- SetParams: Use SetParams(Position, Min, Max) when changing both Min and Max together to avoid intermediate clamping of Position.
- Reversed property: Useful for vertical sliders where 0 should be at the bottom (e.g. a volume knob). Set Reversed := True.
- Selection range visual only: SelStart/SelEnd highlight a zone but do not restrict user input. You must check Position yourself if you want to enforce a range.
- TickStyle tsManual: Call SetTick(Value) after setting TickStyle := tsManual to place individual tick marks. Ticks do not snap the thumb — they are visual only.
- Keyboard navigation: Arrow keys move by LineSize. Page Up/Down moves by PageSize. Home/End jump to Min/Max. This works on all platforms.
- Cross-platform: This lesson uses only standard LCL units. No Windows-specific code. Compiles and runs on Windows, Linux and macOS without changes.



