

TDevice_Independence Overview

The **TDevice_Independence** control was designed to easily facilitate the creation of forms that would fit on any screen. Controls are rearranged and resized to make the form look like the author intended it to. It is recommended that **TrueType** fonts are used, and there are a few considerations that should be observed to ensure a professional-looking, high-impact, device independent application. These concerns are summarized here, and are also listed under the property or properties to which they apply. Please read the [concerns](#) carefully; they will help you to avoid any problems early on.

See:

Properties

OrigFormWidth Property FrmWidth Property

OrigFormHeight Property FrmHeight Property

ScreenWidth Property ScreenHeight Property

MaxScreenSize Percent Property

MinScreenSize Percent Property

FormInUpperLeft Property

AutoStretch Property

Methods

Refresh method

OrigFormWidth Property

The **OrigFormWidth** property contains the width your form was when it was initially loaded. You can use this to reset the form back to its original dimensions. If you do this, make sure that **AutoStretch** is true.

See Also:

[OrigFormHeight Property](#)

OrigFormHeight Property

The **OrigFormHeight** property indicates the original form's design-time height setting. This allows you to make decisions based on the difference between sizes, and can even allow you to surmise things about the display mode!

See Also:

[OrigFormWidth Property](#)

FrmWidth Property

The **FrmWidth** property returns the same thing you would get if your form's name was Form1 and the following code was executed:

```
MyInteger := Form1.Width;
```

This is a read-only property.

See Also:

[FrmHeight Property](#)

FrmHeight Property

The **FrmHeight** property returns the same thing you would get if your form's name was Form1 and the following code was executed:

```
MyInteger := Form1.Height;
```

This is a read-only property.

See Also:

[FrmWidth Property](#)

ScreenWidth Property

The **ScreenWidth** property gives you the width of the user's screen.

This is a read-only property.

See Also:

[ErmWidth Property](#)

ScreenHeight Property

The ScreenHeight property gives you the height of the user's screen.

This is a read-only property.

See Also:

[FrmHeight Property](#)

MaxScreenSize_Percent Property

The **MaxScreenSize_Percent** property allows you to prevent the user (or your own program) from making the form larger than the specified percent of the screen. This is in reference to the form's Width. Therefore, if your form is tall and skinny, it is in reality taking up a small percentage of the screen. If it's a toolbar with a short height and a long width, its percentage of screen is higher. If your form is not shaped like the screen, it's helpful to think of this as the maximum form width in percent.

This value is ignored if **AutoStretch** is false.

If you have forms that you want device independent; but you don't want them taking up the whole screen, this property must be set. You can either use a calculator to figure out the correct setting, or just try a few values and see how the forms look at run-time. If this value is changed at run-time, you must call the **Refresh** event afterwards to ensure that the form is within the limits you specified.

See Also:

[MinScreenSize_Percent](#)

MinScreenSize_Percent

The **MinScreenSize_Percent** property allows you to prevent the user (or your own program) from making the form smaller than the specified percent of the screen. This is in reference to the form's Width. Therefore, if your form is tall and skinny, it is in reality taking up a small percentage of the screen. If it's a toolbar with a short height and a long width, its percentage of screen is higher. If your form is not shaped like the screen, it's helpful to think of this as the minimum form width in percent.

This value is ignored if **AutoStretch** is false.

You can either use a calculator to figure out the correct setting, or just try a few values and see how the forms look at run-time. If this value is changed at run-time, you must call the **Refresh** event afterwards to ensure that the form is within the limits you specified.

See Also:

MaxScreenSize_Percent

FormInUpperLeft Property

The **FormInUpperLeft** property tells TDevice_Independence to position the upper left corner of the form at coordinates 0,0 when it is first displayed. This is useful especially when you're using TDevice_Independence to create a full-screen multimedia or kiosk application. Since by default your form will take up the whole screen, it's helpful to know that it will not be offset. This property should probably be set to False if you're using either the MaxScreenSize_Percent or MinScreenSize_Percent properties, since you probably want the forms to be located elsewhere at runtime.

FormInUpperLeft is boolean. Although you can set it at runtime, there is absolutely no reason to do so.

AutoStretch Property

The **AutoStretch** property is similar to an **Enabled** property; it tells TDevice_Independence to automatically scale your form every time the size changes. It is important to note that if you change this to False at runtime, the form will not be scaled when the user changes its size. This may be fine, but after the size has changed without scaling the form's scaling may not work properly once AutoStretch is turned back on.

AutoStretch is Boolean.

```
Device_Independence1.AutoStretch := False;
```

See Also:

Controlling Form Size

Concerns that affect your application

OrigFormWidth

MaxScreenSize_Percent

OrigFormHeight

MinScreenSize_Percent

Refresh method

The **Refresh** method allows your code to explicitly tell your form to scale **now**. It is not expected that you'll need it, but is included in case there is a conflict in your program that's not allowing the form to update properly. No like behavior is indicated to date, so you can most likely ignore this.

Concerns that affect your application

Please read all of these concerns carefully. It is very important that you understand how this control relates to forms in Delphi applications. There are some behaviors that IDE objects demonstrate. If you are aware of them, you can avoid some headache the author couldn't avoid.

Form sizing

WindowState:

Controlling Form Size

Controlling Form Size:

By default, this control automatically resizes the form to fit the entire screen upon program start. Subsequent resizes cause everything on the form to be rescaled. If this behavior is not desired and you wish your form to be the size on the *user's* screen that it was on *yours*, set the **MaxScreenSize_Percent** property to a value between 1 and 100. If this is done, it will be scaled to the percentage of screen specified in the **MaxScreenSize_Percent** upon program start. This property will also limit the size of manual user resize action--i.e., if the **MaxScreenSize_Percent** property is at 30 and the form is currently covering 20 percent of the screen, the user cannot make the form any larger than 30% of the screen size. The inverse is true for the **MinScreenSize_Percent** property.

Following this, if you want the form to be a certain screen percentage upon program start but want the user to be able to resize it to any size they desire, you can turn this property on and off any time you want to.

How TDevice_Independence knows the size:

It is important to remember, that this control checks the **width** of the form when checking sizes, and makes the **height** of the form the correct size in relation to the width. Of course, if the height would be higher than the screensize, it will be reduced and the width would be brought into line. **Very important to remember also**, and to always keep in mind, is that **TDevice_Independence will always keep the form the same shape at runtime that it was in at design time**. If you want your form to fill the whole screen without gaps at the side or bottom, you must make sure it's **exactly the right shape**. Device independence is gained by keeping all items the same shape they were to begin with, and the form is no exception.

You can bypass this feature if you wish. After the form loads and has been resized, you can turn **AutoStretch** off (**AutoStretch := False**), resize your form, and turn it back on. That should be followed by calling the **Refresh method**. It defeats the 'stretch' properties of this control, however, during the time **AutoStretch** is false. You need to ensure that the size isn't changed too much during feature bypassing.

Your form's WindowState

Your form's WindowState:

TDevice_Independence doesn't care what WindowState your form is set to. This can cause some interesting side-effects. It is possible to have a Maximized form half the size of a Normalized one. If you notice strange effects, try setting the form's WindowState at design-time and see what effect this has on the display of your form. If you have used this suite of controls to create a toolbar and the toolbar doesn't display, look at the WindowState and Visible properties in the Object Browser for that form. If the WindowState is wsNormal and you are using the **MaxScreenSize_Percent** property, try changing the WindowState property for the form and see what happens.

