How to set boundaries for newly created controls

If you are designing a new Delphi control and you need to catch or limit changes to *Left*, *Top*, *Width*, or *Height*, there is one simple way to do it. The key to catching changes to *Left*, *Top*, *Width*, *Height*, and even *BoundsRect* in Delphi is the *SetBounds()* method (found in *TControl* and all descendants). *SetBounds()* is a virtual function used to set the position and size of a control all in one easy step. However, what they do not tell you in the documentation is that *TControl.SetBounds()* is called by *TControl*'s *SetLeft()*, *SetTop()*, *SetWidth()*, and *SetHeight()* property access methods every time a value is assigned to the *Left*, *Top*, *Width*, *Height*, and *BoundsRect* properties.

Thus, to catch changes to these properties, simply override the *SetBounds()* method, do whatever you need with the new values, and then pass them on to the inherited *SetBounds()* method.

In the following example, I have a control that automatically updates its *X* and *Y* custom properties to refer to the center of the control whenever *Left*, *Top*, *Width*, *Height* or *BoundsRect* are changed. Conversely, *Left* and *Top* will be updated whenever *X* and *Y* are changed:

```
type
  TMyControl = class(TControl)
  private
    FX, FY: integer;
    {property access methods}
    procedure SetX(value: integer);
    procedure SetY(value: integer);
    { ... }
  public
    procedure SetBounds(aLeft, aTop, aWidth, aHeight: integer); override;
    { ... }
    property X: integer read FX write SetX;
    property Y: integer read FY write SetY;
  end;
{ ... }

procedure TMyControl.SetX(value: integer);
begin
  if FX <> value then
    SetBounds(value - Width div 2, Top, Width, Height);
end;

procedure TMyControl.SetY(value: integer);
begin
  if FY <> value then
    SetBounds(Left, value - Height div 2, Width, Height);
end;

procedure TMyControl.SetBounds(aLeft, aTop, aWidth, aHeight: integer);
begin
  {Go ahead and let SetBounds() do its thing...}
  inherited SetBounds(aLeft, aTop, aWidth, aHeight);
  {Now adjust FX and FY according to our new bounds.}
  FX := Width div 2;
  FY := Height div 2;
end;
```

Also not mentioned in the documentation is the fact that the *FLeft*, *FTop*, *FWidth*, and *FHeight* private fields that *TControl* uses to keep track of its bounding rectangle are not updated in the corresponding *SetLeft()*, *SetTop()*, etc. property access methods. These variables, in fact, do not get updated anywhere except in *TControl*'s *SetBounds()* method (as with *FX* and *FY* in the above example).

So, to limit changes to the bounds of your control, you can override *SetBounds()* to check and modify any of the properties before passing the values on to the inherited *SetBounds()* method.

In the following example, I have a control that limits its width and height to no more than 100 pixels:

```
type
  TMyControl = class(TControl)
    { ... }
  public
```

```
    procedure SetBounds(aLeft, aTop, aWidth, aHeight: integer); override;
    { ... }
  end;
{ ... }

procedure TMyControl.SetBounds(aLeft, aTop, aWidth, aHeight: integer);
begin
  if aWidth > 100 then
    aWidth := 100;
  if aHeight > 100 then
    aHeight := 100;
  inherited SetBounds(aLeft, aTop, aWidth, aHeight);
end;
```

Original resource:	The Delphi Pool
Author:	Unknown
Added:	2009-08-12
Last updated:	2009-08-12