Dave's Development Blog

Software Development using Borland / Codegear / Embarcadero RAD Studio

Remembering an Applications Size and Position on Multiple Screens

By David | December 9, 2018 0 Comment

Overview

This article is about how I solved the problem of getting my applications to remember their size and position on a multiple monitor systems.

Requirements

I'm one of these people, when using software, I expect them to remember the size and position of dialogues, grids, etc so my requirements were that I needed to remember the size and position of the applications main form (include whether its maximised) on a multi-monitor system. Further, I've seen issues on a number of my older applications that have been used on multiple monitors by other users, especially when they change their configurations. This is because I just used to save the **Top**, **Left**, **Width**, **Height** and **WindowState** of the forms. If someone has moved a form to a secondary monitor and then runs the application just on the laptop, they cannot see the window and many of the younger folk of today do not know about **Alt+Space**, **M** to move forms with the keyboard. So I also needed to have different persistence of the window on different monitor configurations.

Monitor Profiles

So the first thing I started with was understanding different monitor profiles with the following code:

```
Function TfrmMMAMainForm.MonitorProfile: String;

Const
  strMask = '%d=%dDPI(%s,%d,%d,%d,%d)';

Var
  iMonitor: Integer;
  M: TMonitor;

Begin
```

```
  Result := '';
  For iMonitor := 0 To Screen.MonitorCount - 1 Do
    Begin
      If Result <> '' Then
        Result := Result + ':';
      M := Screen.Monitors[iMonitor];
      Result := Result + Format(strMask, [
        M.MonitorNum,
        M.PixelsPerInch,
        BoolToStr(M.Primary, True),
        M.Left,
        M.Top,
        M.Width,
        M.Height
      ]);
    End;
End;
```

Its not really rocket science but the above code builds a string based on the Monitor Number, DPI, Primary Status, Left, Top, Width and Height and concatenates the multiple monitors with a colon.

Saving the Size and Position

One of the early issues I found when trying to save the size and position of windows was that if the window has been maximised then the Top, Left, Width and Height were for the maximised window. So a long time ago I found the windows API function **GetWindowPlacement()**. This function will provide you the **Top**, **Left**, **Height** and **Width** of the restored window even when the form is maximised, i.e. this is the size and position of the form when you double click the title or select the restore icon from the title bar. So I continue to use this but also now use the equivalent **WindowState** from the information returned by the methods as follows:

```
Procedure TfrmMMAMainForm.SaveSettings;

Var
  strMonitorProfile : String;
  recWndPlmt : TWindowPlacement;

Begin
  strMonitorProfile := MonitorProfile;
  recWndPlmt.Length := SizeOf(TWindowPlacement);
  GetWindowPlacement(Handle, @recWndPlmt);
  FINIFile.WriteInteger(strMonitorProfile, 'Top', recWndPlmt.rcNormalPosition.Top);
  FINIFile.WriteInteger(strMonitorProfile, 'Left', recWndPlmt.rcNormalPosition.Left);
  FINIFile.WriteInteger(strMonitorProfile, 'Height',
recWndPlmt.rcNormalPosition.Height);
  FINIFile.WriteInteger(strMonitorProfile, 'Width', recWndPlmt.rcNormalPosition.Width);
  FINIFile.WriteInteger(strMonitorProfile, 'WindowState', recWndPlmt.showCmd);
  FINIFile.Save;
End;
```

Loading the Size and Position

The biggest change to my original code was that I now use the **SetWindowPlacement()** function when restoring the window size and position on loading the application as follows:

```
Procedure TfrmMMAMainForm.LoadSettings;

Var
  strMonitorProfile : String;
  recWndPlmt : TWindowPlacement;

Begin
  strMonitorProfile := MonitorProfile;
  recWndPlmt.Length := SizeOf(TWindowPlacement);
  recWndPlmt.rcNormalPosition.Top := FINIFile.ReadInteger(strMonitorProfile, 'Top',
100);
  recWndPlmt.rcNormalPosition.Left := FINIFile.ReadInteger(strMonitorProfile, 'Left',
100);
  recWndPlmt.rcNormalPosition.Height := FINIFile.ReadInteger(strMonitorProfile,
'Height', 480);
  recWndPlmt.rcNormalPosition.Width := FINIFile.ReadInteger(strMonitorProfile, 'Width',
640);
  recWndPlmt.showCMD := FINIFile.ReadInteger(strMonitorProfile, 'WindowState',
SW_NORMAL);
  SetWindowPlacement(Handle, @recWndPlmt);
End;
```

Again, not rocket science.

Known Issues

There are a number of known issues with this code which I might change in the future as follows:

- The code does not store any information about the relative position of the monitors to each other although you will not change this very often;
- If you were to undock your laptop with the applications running then the current position of the application will be stored on closing to a different monitor profile. You probably want to call the method
- **Vcl.Forms.TCustomForm.MakeFullyVisible()** on the form's **OnActivate** event handler to ensure that the form is bought into view.

I hope this provide some food for thought, it has certainly made my application better citizens of multi-monitor environments.

regards
Dave

Category: Applications Delphi RAD Studio