Subject: [solved] Register a callback on "when window resized" ?
Posted by xrysf03 on Sun, 24 Nov 2019 22:35:00 GMT
View Forum Message <> Reply to Message

Dear everyone,

another rookie question:
So I have this toy app - a main window, containing a handful of uninteresting controls, and a large ScatterCtrl to plot some data. That ScatterCtrl is "anchored" to all four edges of the main window = scales with the window.

And the question is:

Is there an overridable callback, or an event / WhenSomething that I could use, that gets called "on Window resize"? Actually on window resize or on ScatterCtrl resize.

The point is: I'd like to update the axis major units (grid spacing) in response to the control changing dimensions. I already know how to set the grid units/dimensions, but I don't know how to have myself woken up when the ScatterCtrl's pixel dimensions change.

Thanks in advance for any ideas...

Frank

Subject: Re: Register a callback on "when window resized" ?
Posted by Oblivion on Sun, 24 Nov 2019 22:53:11 GMT
View Forum Message <> Reply to Message

Hello Frank,

Quote:Is there an overridable callback, or an event / WhenSomething that I could use, that gets called "on Window resize"? Actually on window resize or on ScatterCtrl resize.

Ctrl::Layout() virtual method is used for that purpose. It is called whenever the layout of a given ctrl has changed.

Best regards,
Oblivion

Subject: Re: Register a callback on "when window resized" ?
Posted by xrysf03 on Mon, 25 Nov 2019 21:14:18 GMT
View Forum Message <> Reply to Message

Oh, excellent, it appears to work :)
Well kind of.
It works if I just resize the window in "normal size" mode.
For some reason, it doesn't work when I maximize the window. Actually it does seem to re-calculate the density of the MajorUnit grid for the maximized size, but the denser grid only gets applied after I "un-maximize" the window again - at which point the grid is already too dense :) So it kind of "lags one step behind". Even if I finish it off by Ctrl::ProcessEvents() .

I have noticed that the TopWindow indeed has the Layout() method, and of course my main window inherits from that class (or some template around it), so I just added the overridden method to my existing class. (I did not need to inherit from ScatterCtrl and swap my own class into the visual layout.) A snippet of my example code follows:

```
class my_main_window : public Withmy_main_windowLayout<TopWindow> {

 [...misc stuff...]

 //Event<> WhenLayout; // not even necessary
 void Layout() override
 {
  // first and foremost, do whatever you need to
  // with all the widgets (resize etc)
  Withmy_main_windowLayout<TopWindow>::Layout();

  // and then finally, mess with the ScatterCtrl
  set_chart_grid();

  Chart1.Refresh();  // doesn't make a difference
  Ctrl::ProcessEvents(); // doesn't make a difference
  //PromptOK("hello there");  // DOES make a difference?
 }
}
```

The "misbehavior" upon maximize is fairly consistent. I've tried adding a Chart1.Refresh() and/or Ctrl::ProcessEvents(), tried swapping the order of set_chart_grid() and the call to the upstream version of the Layout() method, I've also tried without calling the upstream version, to no avail. The only moment when the ScatterCtrl did reflect the desired changes "upon main window maximize", was when I also inserted some PromptOK("hello there") at the end of my overridden version of Layout().

It's curious, but I don't mind if that's just the way it is - as soon as I run my method that fetches some data from the back-end and displays them (which gets started by clicking a GUI button), the grid with appropriate spacing gets displayed in the ScatterCtrl Chart1.

Subject: Re: Register a callback on "when window resized" ?
Posted by slashupp on Tue, 03 Dec 2019 05:32:24 GMT
View Forum Message <> Reply to Message

don't know what you do in set_chart_grid(), but it should use the current dimensions of your ScatterCtrl for it's calculations
just do:

```
void YOURWINDOWNCLASS::Layout()
{
   set_chart_grid();
}
```

see how that goes

Subject: Re: Register a callback on "when window resized" ?
Posted by xrysf03 on Sun, 08 Dec 2019 13:12:24 GMT
View Forum Message <> Reply to Message

@slashupp: thanks for your response and for your care :)
Just for the record, the set_chart_grid() method goes like this:

```
void rtlsdr_skyline::set_chart_grid()
{
 if (display_in_calib_mode)
  return; // no modification needed

 if ((from_freq_rounded == 0) || (to_freq_rounded == 0))
  return;  // it's too early

 // grid spacing along axis X
 double x_tick = chart1_optimal_major_unit_x();

 // avoid unnecessary fractional digits in grid tick label,
 // by shifting the grid conveniently into alignment
 // with integer multiples of grid tick
 double x_ofs = (from_freq_rounded/1000000)
  - (ffloor(from_freq_rounded / 1000000 / x_tick) * x_tick);

 Chart1.SetXYMin(from_freq_rounded / 1000000, -100);
 Chart1.SetRange((to_freq_rounded - from_freq_rounded)/ 1000000, 100);
 Chart1.SetMinUnits(x_ofs,0);
 Chart1.SetMajorUnits(x_tick, 10);

 Chart1.SetLabels("MHz","dBFS");
```

```
 Chart1.ShowLegend(false);
}
```

...where chart1_optimal_major_unit_x() indeed polls the ScatterCtrl for its horizontal dimension in pixels, via GetPlotWidth() and calculates its output based on some "double" numbers in my own code.
It takes the frequency range, walks the exponential scale of a few decimal orders and looks for a nice spacing with a ratio of 1, 2 or 5 across the given range.

```
// the returned value is in MHz
double rtlsdr_skyline::chart1_optimal_major_unit_x()
{
 double retval = 1; // default response (1 MHz)
 double decimal_order = 0;
 int chart_horiz_pixels = Chart1.GetPlotWidth();

 if (chart_horiz_pixels < DIV_NO_SMALLER_THAN_PX)
  return retval; // prevent dv_by_zero. Return the default response.

 // we could also use "bandwidth" and "num_freqs" and whatnot...
 // but from_freq_rounded and to_freq_rounded are perhaps
 // the closest to the visual chart.
 double chart_bandwidth = (to_freq_rounded - from_freq_rounded) / 1000000;
 double mhz_per_optimal_div = chart_bandwidth
  / (chart_horiz_pixels / DIV_NO_SMALLER_THAN_PX);

 for (decimal_order = 0.001; decimal_order <= 1000; decimal_order *= 10)
 {
  if (decimal_order > mhz_per_optimal_div)
  {
   retval = decimal_order;
   break;
  }
  else if (decimal_order * 2 > mhz_per_optimal_div)
  {
   retval = decimal_order * 2;
   break;
  }
  else if (decimal_order * 5 > mhz_per_optimal_div)
  {
   retval = decimal_order * 5;
   break;
  }
  // else try another decimal order
 }
```

```
 return retval;
}
```

So it's really along the lines of what you suggest.

Nonetheless, I think I've caught wind of some other peculiarities in the inner workings of the GUI that I should pay attention to, with respect to flow of control, reentrancy of event handlers, calling Ctrl::ProcessEvents() etc. Vs. using threads etc. I'll start another topic on the subject.

Frank
