From: **Eric Wachsmann** eric@flex-radio.com
Subject: Re: Inquiry From: douglas.adams@me.com
Date: March 8, 2016 at 7:06 PM
To: Doug Adams douglas.adams@me.com
Cc: Devhelp@flexradio.com devhelp@flexradio.com

Doug,

The gradients are currently stored in the closed source for the SmartSDR for Windows client. However, I'm happy to share the code snippets for the relevant pieces provided that the code's intent is for use on FlexRadio hardware. Let me know if you have any questions. Here are the snippets:

```
private void GenerateAllWaterfallGradients()
        {

            // the default gradient
            waterfallGradientBasic.colorList.Clear();
            waterfallGradientBasic.colorList.Add(new GradientColorStop(Colors.Black, 0.0));
            waterfallGradientBasic.colorList.Add(new GradientColorStop(Colors.Blue, 0.15)); // 0.15
            waterfallGradientBasic.colorList.Add(new GradientColorStop(Colors.Cyan, 0.25)); // 0.10
            waterfallGradientBasic.colorList.Add(new GradientColorStop(Color.FromRgb(0, 255, 0), 0.35));
// 0.10
            waterfallGradientBasic.colorList.Add(new GradientColorStop(Colors.Yellow, 0.55)); // 0.20
            waterfallGradientBasic.colorList.Add(new GradientColorStop(Colors.Red, 0.90)); // 0.35
            waterfallGradientBasic.colorList.Add(new GradientColorStop(Colors.White, 1.0)); // 0.10

            // Abed's choice of colors
            waterfallGradientPurple.colorList.Clear();
            waterfallGradientPurple.colorList.Add(new GradientColorStop(Colors.Black, 0.0));
            waterfallGradientPurple.colorList.Add(new GradientColorStop(Colors.Blue, 0.15));
            waterfallGradientPurple.colorList.Add(new GradientColorStop(Color.FromRgb(0, 255, 0), 0.30));
            waterfallGradientPurple.colorList.Add(new GradientColorStop(Colors.Yellow, 0.45));
            waterfallGradientPurple.colorList.Add(new GradientColorStop(Colors.Red, 0.60));
            waterfallGradientPurple.colorList.Add(new GradientColorStop(Colors.Purple, 0.75));
            waterfallGradientPurple.colorList.Add(new GradientColorStop(Colors.White, 1.0));

            // Dark
            waterfallGradientDark.colorList.Clear();
            waterfallGradientDark.colorList.Add(new GradientColorStop(Colors.Black, 0.0));
            waterfallGradientDark.colorList.Add(new GradientColorStop(Colors.Blue, 0.65));
            waterfallGradientDark.colorList.Add(new GradientColorStop(Color.FromRgb(0, 255, 0), 0.90));
            waterfallGradientDark.colorList.Add(new GradientColorStop(Colors.Red, 0.95));
            waterfallGradientDark.colorList.Add(new GradientColorStop(Colors.LightPink, 1.0));
            waterfallGradientDark.colorList.Add(new GradientColorStop(Colors.LightPink, 1.0));
            waterfallGradientDark.colorList.Add(new GradientColorStop(Colors.LightPink, 1.0));


            // No colors allowed
            waterfallGradientGrayscale.colorList.Clear();
            waterfallGradientGrayscale.colorList.Add(new GradientColorStop(Colors.Black, 0.0));
            waterfallGradientGrayscale.colorList.Add(new GradientColorStop(Colors.White, 1.00));
            waterfallGradientGrayscale.colorList.Add(new GradientColorStop(Colors.White, 1.00));
            waterfallGradientGrayscale.colorList.Add(new GradientColorStop(Colors.White, 1.00));
            waterfallGradientGrayscale.colorList.Add(new GradientColorStop(Colors.White, 1.00));
            waterfallGradientGrayscale.colorList.Add(new GradientColorStop(Colors.White, 1.00));
            waterfallGradientGrayscale.colorList.Add(new GradientColorStop(Colors.White, 1.00));
            waterfallGradientGrayscale.colorList.Add(new GradientColorStop(Colors.White, 1.00));

            // Deuteranopia
            waterfallGradientDeuteranopia.colorList.Clear();
            waterfallGradientDeuteranopia.colorList.Add(new GradientColorStop(Colors.Black, 0.0));
       // Black
            waterfallGradientDeuteranopia.colorList.Add(new GradientColorStop(Color.FromRgb(8, 60,
107), 0.15));     // Dark Blue
            waterfallGradientDeuteranopia.colorList.Add(new GradientColorStop(Color.FromRgb(132, 162,
214), 0.50)); // Light Blue
            waterfallGradientDeuteranopia.colorList.Add(new GradientColorStop(Color.FromRgb(165, 150,
115), 0.65));  // Dark Yellow
            //waterfallGradientDeuteranopia.colorList.Add(new GradientColorStop(Color.FromRgb(255, 219,
49), 0.80));  // Less Dark Yellow
            waterfallGradientDeuteranopia.colorList.Add(new GradientColorStop(Colors.Yellow, 0.75)); //
Light Yellow
            waterfallGradientDeuteranopia.colorList.Add(new GradientColorStop(Colors.Yellow, 0.95));
        // White
            waterfallGradientDeuteranopia.colorList.Add(new GradientColorStop(Colors.White, 1.00));
        // White

            // Tritanopia
            waterfallGradientTritanopia.colorList.Clear();
            waterfallGradientTritanopia.colorList.Add(new GradientColorStop(Colors.Black, 0.0));
            waterfallGradientTritanopia.colorList.Add(new GradientColorStop(Color.FromRgb(0, 69, 82),
0.15)); // dark teal
            waterfallGradientTritanopia.colorList.Add(new GradientColorStop(Color.FromRgb(107, 186,
214), 0.45));   // light blue  rgb(107, 186, 214)
            waterfallGradientTritanopia.colorList.Add(new GradientColorStop(Color.FromRgb(74, 8, 24),
0.46));  // dark red rgb(74, 8, 24)
            waterfallGradientTritanopia.colorList.Add(new GradientColorStop(Colors.Red, 0.90));   // light
red
            waterfallGradientTritanopia.colorList.Add(new GradientColorStop(Color.FromRgb(214, 121,
132), 0.99));   // light red rgb(214, 121, 132)
            waterfallGradientTritanopia.colorList.Add(new GradientColorStop(Colors.White, 1.00));   //white

==============================

private void GetWaterfallColor(ushort input, out byte red, out byte green, out byte blue)
        {
            ushort low, high;
            if (_autoBlackLevelEnable)
            {
                low = _fallAutoLowThreshold;
                high = _fallAutoHighThreshold;
            }
            else
            {
                low = _fallLowThreshold;
                high = _fallHighThreshold;
            }
```

```
         if (_updateWaterfallColors && _waterfallGradientSelected != null)
         {
            // choose a specific color gradient to use
            // this needs to be called whenever the selected gradient has changed
            GenerateColorGradientArray(_waterfallGradientSelected);
            _updateWaterfallColors = false;
         }

         // figure out where input is relative to color dynamic range
         // is the input below the low threshold?
         if (input <= low)
         {
            // yes -- just use the low color value
            red = _fallLowColor.R;
            green = _fallLowColor.G;
            blue = _fallLowColor.B;
         }
         // no -- is the input above the high threshold?
         else if (input >= high) // input is above range
         {
            // yes -- just use the high color value
            red = _fallHighColor.R;
            green = _fallHighColor.G;
            blue = _fallHighColor.B;
         }
         else // no --  input is in the middle of the range
         {
            // We need to figure out where the input is as a percentage of the color space.
            // This the low and high thresholds into account
            float percent = (input - low) / (float)(high - low);

            // percentage to 0-65535
            ushort colorIndex = (ushort)((float)ushort.MaxValue * percent);

            red = waterfallColorArray[colorIndex].R;
            green = waterfallColorArray[colorIndex].G;
            blue = waterfallColorArray[colorIndex].B;
         }
      }

====================================

private void UpdateAutoColorDynamicRange()
      {
         _fallAutoHighThreshold = CalculateHighThreshold(_fallAutoLowThreshold);
         // force a redraw since we have changed color parameters
         KickRenderWaterfall();
      }

==================================

private ushort CalculateHighThreshold(ushort low_threshold)
      {
         ushort ret_val = 0;

         // adjust high boundary from low + margin to max in X^3 pattern
         // move from 0-100 space into [1,cuberoot(2^16) space]
         double temp = (100 - _fallColorGain) / 100.0 * Math.Pow(ushort.MaxValue - low_threshold, 1 /
3.0); // need a buffer between low and high??

         // now scale the value using the new value
         ret_val = (ushort)(low_threshold + Math.Pow(temp, 3.0));

         // Make sure that the _fallHighThreshold is not lower than _fallLowThreshold
         // Lets give a minimum allowed separation of 100
         if (ret_val < low_threshold + 100)
            ret_val = (ushort)(low_threshold + 100);

         return ret_val;
      }

===================================
      private ushort CalculateLowThresholdFromBlackLevelSlider()
      {
         double val = 1.0 - (double)_fallBlackLevel / 100.0; // map the 0-100 black level slider to an
inverted 1.0-0.0
         double val2 = Math.Pow(val, 8); // remap the value to give extra dynamic range on the low end of
the slider -- note that this leaves the values from 75-100 with no change
         return (ushort)(val2 * (ushort.MaxValue - 10000)); // bring back into the 2^16 space (less the
typical noise floor value)
      }
```

On Mon, Mar 7, 2016 at 10:40 AM, Doug Adams <douglas.adams@me.com> wrote:

> Dear devhelp,
>
> As you can see from the email trail below, I have asked about beta access and additional information regarding SmartSDR. Matt Youngblood suggested I pose my questions to you. There are only two (I think simple) questions at this time. I've highlighted them below in red. If you are able to help, I'm sure I will have other similar questions from time to time.
>
> I very much appreciate any help you can give me.
>
> 73's
> Doug
>
>> Doug,
>>
>> Thanks for the feedback. I am glad to hear that you have been working with the API! You can communicate directly with our engineers using the devhelp@flexradio.com email. They can point you in the right direction on your questions.
>>
>> Thanks again for the note. Let me know if you need anything else!
>>
>> Best,
>>
>> Matt Youngblood

FlexRadio Systems
4616 W Howard Lane Ste 1-150
Austin, TX 78728
Phone: 512-535-4713

“Rediscover Radio!” ™

On Mon, Mar 7, 2016 at 10:11 AM, Doug Adams <douglas.adams@me.com> wrote:

Matt,

Thank you for that information. Until such time as an opening occurs is there a way that I can obtain information regarding some of the inner workings of SmartSDR? I’m working on a Mac-native (OS X) version of the API and a Mac-native client program similar to SmartSDR. I’ll give you an example of the kind of information I’m talking about.

Right now I’m working on Waterfall:

1. Is there somewhere I can find the color gradients actually used by Flex? I’ve made my own but it would be nice to mimic yours.

2. What is the algorithm Flex uses to compute Auto Black Level and how is Color Gain used in the gradient calculation. Again, I’ve made my own approximation but, if it’s available, I’d like to use the Flex version.

I don’t see these types of things being discussed on the Community site. Most people don’t need this level of detail.

I purchased my 6500 just before Dayton last year. I attended Dayton and spoke with a number of people at your booth but at that point I hadn’t started developing my code so my questions were very basic. Now I am close to a working version of the software and questions like the above come up from time to time.

To be clear, I have no commercial interest at all, I’m retired and don’t need a job. The software I develop will be placed on GitHub just prior to Dayton and made available to anyone at no charge. It is built upon knowledge that I have gained from reviewing your C# API’s public source code, the work done by Stu Phillips on his Objective-C version of your API and a lot of time spent watching SmartSDR with Wireshark. My code gives credit to all of its influences in whatever manner required by the codes’ authors.

I will attend Dayton again this year and hope to demonstrate my software to your team.

Thank for your help.

Doug

On Mar 7, 2016, at 9:26 AM, Matt Youngblood <matt@flexradio.com> wrote:

Doug,

Thank you for your email and your great support of FlexRadio! We have a beta team that tests our

Thank you for your email and your great support of FlexRadio! We have a beta team that tests our products and software before they are released. This is a fixed set of people that is managed by the engineering department. Occasionally they will have openings that they will need to fill so I will pass on your information to them in case they need someone else. Thanks again!

Best regards,

Matt Youngblood
FlexRadio Systems
4616 W Howard Lane Ste 1-150
Austin, TX 78728
Phone: 512-535-4713

"Rediscover Radio!" ™

On Fri, Mar 4, 2016 at 8:29 PM, FlexRadio Website <no-reply@flexradio.com> wrote:

> To:
> Amateur Sales
>
> Name:
> Douglas Adams
>
> Call Sign:
> K3TZR
>
> Email:
> douglas.adams@me.com
>
> Message:
> I'm a proud 6500 owner who also happens to be a (retired) software developer. I would like to know if there is a Flex Radio program that would allow me to have access to beta versions of SmartSDR and/or to technical information concerning SmartSDR that might not be available on the Community site.
>
> 73's - Doug
> K3TZR