Table of Contents

1. INTRODUCTION

Last Christmas I used an Effects Generator plug-in (FxGen) to dynamically generate channel values for some RGB props (for example, a Snow Globe prop[1]), rather than adding 1000s more channels directly to my Vixen 2.x Sequences and editing all those channels by hand. The result was much smaller Sequence files and vastly easier cell editing within Vixen.

FxGen is an *incremental* approach to adding effects to a Vixen Sequence – it can be used in conjunction with other tools so you are not "locked into" using only FxGen. Although this is a work in progress and has some rough edges, FxGen has enough functionality in it that it might be helpful to other DIYC members so I'll document what I have so far. I'll begin by giving a brief overview of FxGen operation, and then describe how to use it in an actual Sequence. I will not cover background information such as Vixen 2.x Sequencing techniques or terminology; the reader is assumed to already be familiar with that info.

There are a couple of demo Sequences included with FxGen, so you can quickly try it out to see if it might be useful to you (just skip ahead to the Demo section if you want). One of the demo Sequences uses over 12,000 channels so it will be a little laggy/jumpy/choppy when running on older PCs, but this is mostly due to the Adjustable Preview and Grid Editor[2] plug-ins rather than FxGen itself (Adjustable Preview and Grid Editor display channel updates on the screen).

The source code is included with FxGen, so you can add your own effects - if you do, please share them, in keeping with the spirit of DIYC openness. After the Christmas 2012 season I did some initial clean-up work, but FxGen is still somewhat messy and there are still bugs in it. Although it seems to work fairly well in the Sequences I've used it with, please use caution - make backups of any Sequences or Profiles you use it with, and have a fall-back plan in case it stops working.

If you have questions or run into problems, let me know. Due to the large variety of display environments, FxGen may not work with every combination, but I can at least take a look at it.

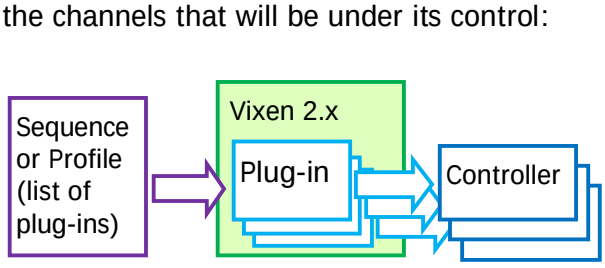
BTW, FxGen is not trying to compete with Vixen 3.x, LSP, HLS or any other DIYC tools. Vixen 3.x seems to be addressing some of the issues related to large RGB channel counts, but some of my goals go beyond that and I had already started down the FxGen path before Vixen 3.0 was announced, so for now I think I will continue along this path unless it turns into a dead end. This is DIY, after all. ☺

[1] http://doityourselfchristmas.com/forums/showthread.php?19450-How-I-Did-GECE-Snow-Globe-Renard-RGB-Retrofit
[2] The Grid Editor plug-in is described in the Dumb RGB Pixel Grid write-up at http://downloads.eShepherdsOfLight.com

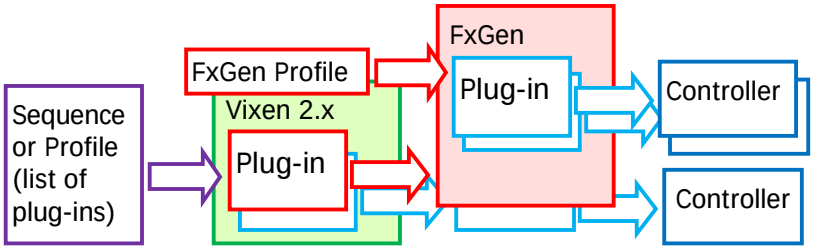
2. Overview and Concepts

FxGen is a Vixen 2.x Output Plug-in that uses a "virtual" Vixen Profile to control other Output Plug-ins.

FxGen is similar in nature to desktop VMWare[3], with the Vixen Sequence acting as the "host" and virtualized plug-ins and channels behaving as "guests" running within that host. As far as Vixen is concerned, FxGen is just another Output Plug-in, so it can be added into any Sequence or Profile just like other plug-ins. However, once FxGen has been added, it uses its own private Vixen Profile to load additional "virtual" plug-ins to control the channels that will be under its control:



Vixen instantiates plug-ins normally



Vixen loads FxGen which then loads more plug-ins

I only used FxGen for RGB-related plug-ins last season because those were the ones that had larger numbers of channels. In theory, FxGen can be used to virtualize any Vixen 2.x Output Plug-in; the Renard-RGB, Adjustable Preview, and Grid Editor plug-ins seem to work okay within FxGen. I only tried the plug-ins that I needed so there might be problems with other Output Plug-ins – if you find some let me know and I'll try to fix them (if FxGen is the problem). Note that most Output Plug-ins do not allow port sharing, so that may impose some limitations here (ie, you can't use the same port between multiple (virtual and real) copies of an Output Plug-in).

Within the host Vixen Sequence or Profile, other plug-ins can be used alongside FxGen. I ran my non-RGB props directly from within the host Vixen Profile rather than FxGen because my sequences already had channel values hand-sequenced for those from previous years. If I had wanted to apply new effects using FxGen, I could have moved those plug-ins into my FxGen virtual Profile also – that would have left the host Vixen Profile nearly empty. (It can't ever be completely empty because FxGen itself requires a few channels for its operation).

With virtualized Output Plug-ins under the control of FxGen rather than Vixen itself, channel values can be dynamically generated at run-time. This leads to a number of useful advantages:

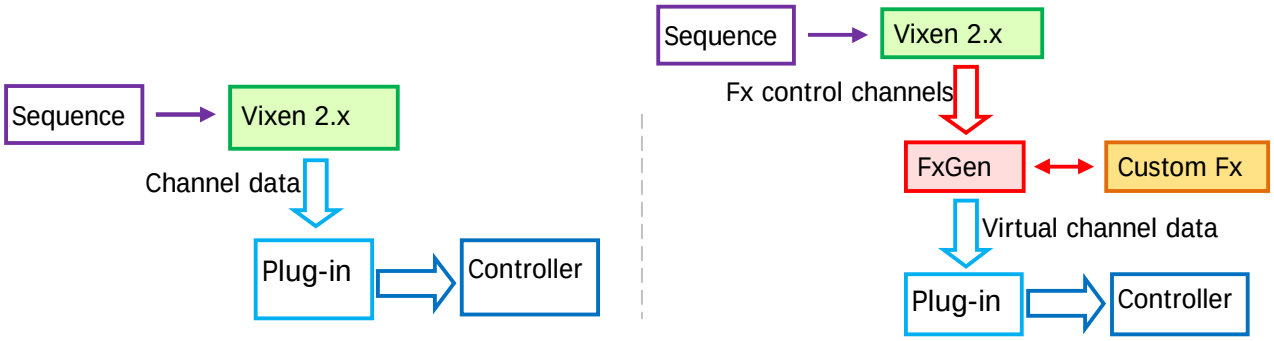
- Sequence files shrink; that big <EventValues> matrix of Base 64 encoded values nearly goes away.
- Sequence effects can be generated using functions in C#, VB.NET, or any other language; these can be arbitrarily complex, or can utilize external run-time data (clock, temperature sensors, cameras, video, etc)
- Vixen channels can be grouped and mapped into more convenient structures; for example, I mapped them to a Windows GDI Bitmap, which then allowed me to use GDI functions to generate the effects. (This was also an initial exploratory step to verify that Madrix-like functionality could be added to Vixen fairly easily).

Since FxGen uses pre-compiled custom effects functions, I see it as more of a framework for adding custom effects than as a regular pre-packaged Output Plug-in. To enforce the separation of custom effects from the base behavior, I placed the underlying "machinery" into an abstract base class so the FxGen plug-in itself cannot actually be added to a Vixen Sequence or Profile. Instead, the FxGen object class must be used to derive a custom child class, which can then be used as the actual Vixen plug-in. The steps to do this are described in later sections. By giving each derived class a different name, effects "libraries" can be built and used.

[3] See http://www.vmware.com

2.1. Operation

Conceptually, the operation of FxGen is quite simple. During the StartUp event Vixen will create an instance of FxGen, which will in turn create an instance of each Output Plugin listed in its virtual Profile. Then for each event period, Vixen will invoke FxGen just like a regular Output Plug-in, causing FxGen to invoke custom effects functions to generate channel values. Those channel values are then passed on to the virtualized Output Plug-ins by FxGen using the regular Vixen Output Plug-in interface. As such, most Output Plug-ins do not really know that they are being invoked by FxGen instead of Vixen (Grid Editor is an exception, because it tries to get run-time info such as overall Sequence length, but FxGen only has Profile info available). I suppose FxGen might also be an easy way to run Vixen 2.x plug-ins within Vixen 3.x in future – just modify FxGen itself to work with Vixen 3.x, and then any Vixen 2.x plug-in would be able to run via FxGen within Vixen 3.x without change.

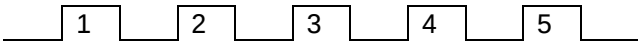


Normal channel data flow *Channel data flow thru FxGen*

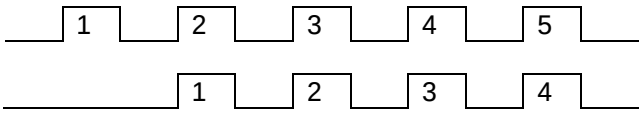
For my sample/custom effects, I designated a "Macro function" channel to control which effects functions to invoke. Some effects functions require more info than that, so I also used some additional "parameter" channels to specify the behavior of the individual effects functions. These channels can all be edited using the regular Vixen cell editing functions. See the Sample Effects section for more details about the macro control channels.

2.2. Timing Pipeline

I used fairly simple effects functions in my Sequences and found that FxGen was able to keep up okay during Sequence playback. However, more complex CPU-intensive effects could potentially bog down FxGen and affect Sequence playback timing. To help protect against this, FxGen will call the effects functions *one frame ahead* of when they are actually needed and then cache the results so they are available immediately when Vixen processes the next frame:



Vixen passes channel values to normal Plug-ins for each event period



FxGen calls effects functions one frame ahead so generated channel values are ready "on time"

The net effect is that the virtualized Output Plug-ins receive the generated channel values at the same time that they would have received them from Vixen. I suppose the only down side would be if any of the effects functions used external data from a clock or sensor – those would end up being delayed by one event period before they were displayed. I don't think this is significant, however, since an event period is typically 50 msec or less anyway.

Since Vixen can sometimes skip frames (due to timing issues), and because it also supports looping over a range of cells within the Sequence, FxGen uses a few channels for a "timing track". These tell FxGen which point it is at within the Sequence. The timing track will be generated automatically at start of playback if needed.

2.3. Sample Effects Functions

The effects functions can do anything you want. For this first round, I only implemented some rudimentary effects related to filling a rectangular grid with colored pixels - just enough for my style of sequencing plus a few "experimental" effects. I am releasing these as-is in case they are useful to other DIYC members, and because they serve as examples of how to use FxGen. I found the Text and Bitmap effects to be the most useful, since they both support animation and scrolling. Eventually I plan to implement more elaborate effects.

The effects functions that I implemented are listed below.

Function	Code	Parameters[4]	Purpose
Noop	0	n/a	No change (continue previous effect)
FillBkg	200	Color	Solid fill and set background color
FillFg	201	Color	Solid fill and set foreground color
FillRGBTest	202	None	R/G/B sparkle[5] test pattern; cycles pixels R-G-B
BTWipe	203	Color, speed	Bottom-to-top fill
TBWipe	204	Color, speed	Top-to-bottom fill
LRWipe	205	Color, speed	Left-to-right fill
MidWipe	206	Color, speed	Fill from center outward
EdgeWipe	207	Color, speed	Fill from edge inward to middle
Spiral	208	Color, speed	Diagonal fill; intended to give "spiral" effect
GECETest_zzud	209	Color, speed	Up-down zig-zag test pattern for GECE Snow Globe
GECETest_zzlr	210	Color, speed	Left-right zig-zag test pattern for GECE Snow Globe
DrawBorder	211	Color	Draw border
BTLine	212	Color, speed	Bottom-to-top line
LRLine	213	Color, speed	Left-to-right line
SpiralLine	214	Color, speed	Diagonal line; intended to give "spiral" effect
DrawColumn#	215 – 218	Color, columns[6]	Draw column (draw vertical line)
DrawRow	219	Color, rows[7]	Draw row (draw horizontal line)
DrawCorners	220	None	Draw corners; mainly for test/alignment checking
Burst	226	Color, speed	Burst/fireworks
Snow	227	Color, speed/drip[8]	Snow/drip
EqBar#	228 – 232	Color, height/speed[9]	Eq bars; used to simulate frequency spectrum bars
ShowBitmap	233	Text[10]	Show bitmap (can be animated/scrolling)
ShowText	234	Color, text[11]	Show text (can be animated/scrolling)
Countdown	235	Color, text[12], value	Count down/up
Timer	236	Color, text[13]	Timer/countdown clock

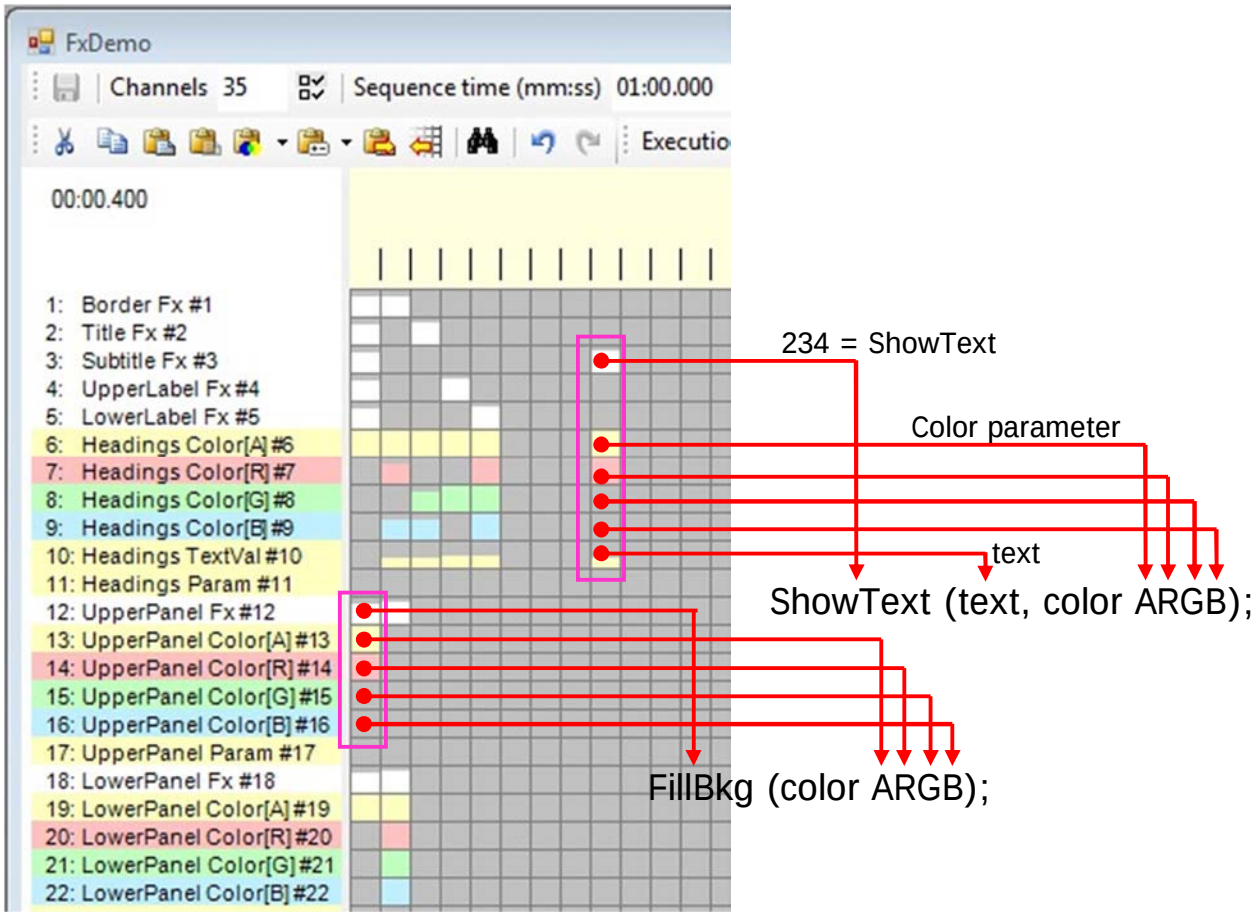
There are various arbitrary conventions I used on the parameter values, as noted below.

[4] Color parameter is ARGB and uses 4 channels: first = opacity, second = red, third = green, fourth = blue
[5] See Wizards in Winter video http://vimeo.com/34712313 at 2:50 - 3:00
[6] Columns: upper nibble = Bkg column#, lower nibble = Fg column#, 2 lower bits of func# = bit 0x10 of column#
[7] Rows: upper nibble = row# to draw using Bkg color, lower nibble = row# to draw using Fg color
[8] Speed/drip: top bit of speed is masked and used to select "drip" vs. "snow"
[9] Height/speed: upper nibble = height, lower nibble = speed; eq bar# determines horizontal position
[10] Text string specifies path name of bitmap file, with additional attributes for scrolling, offset and loop duration
[11] Text string specifies text to be displayed, with additional attributes for font, scrolling, offset and loop duration
[12] Text attribute selects font, horizontal scroll selects counter speed; value 0 for count up, non-0 for count-down
[13] Text attribute selects font, horizontal scroll speed selects up/down count direction

I used several channels within the Sequence to control the effects functions. In some cases these are a little convoluted, and I probably should have used more tags rather than channels. Oh well.

The "Macro function" channel controls which effects function to be called. The Code column in the table above shows the actual channel value to use in order to invoke the effect. I started the values at 200 so it would be easier to see when the cells have a non-0 value in Vixen.

Most of the effects functions also use some additional parameters (shown in the Parameters column). Some functions are one-shot, while others continue running until another effect replaces them. Whether or not a function repeats is determined by the function itself, and the duration is controlled by the parameters.



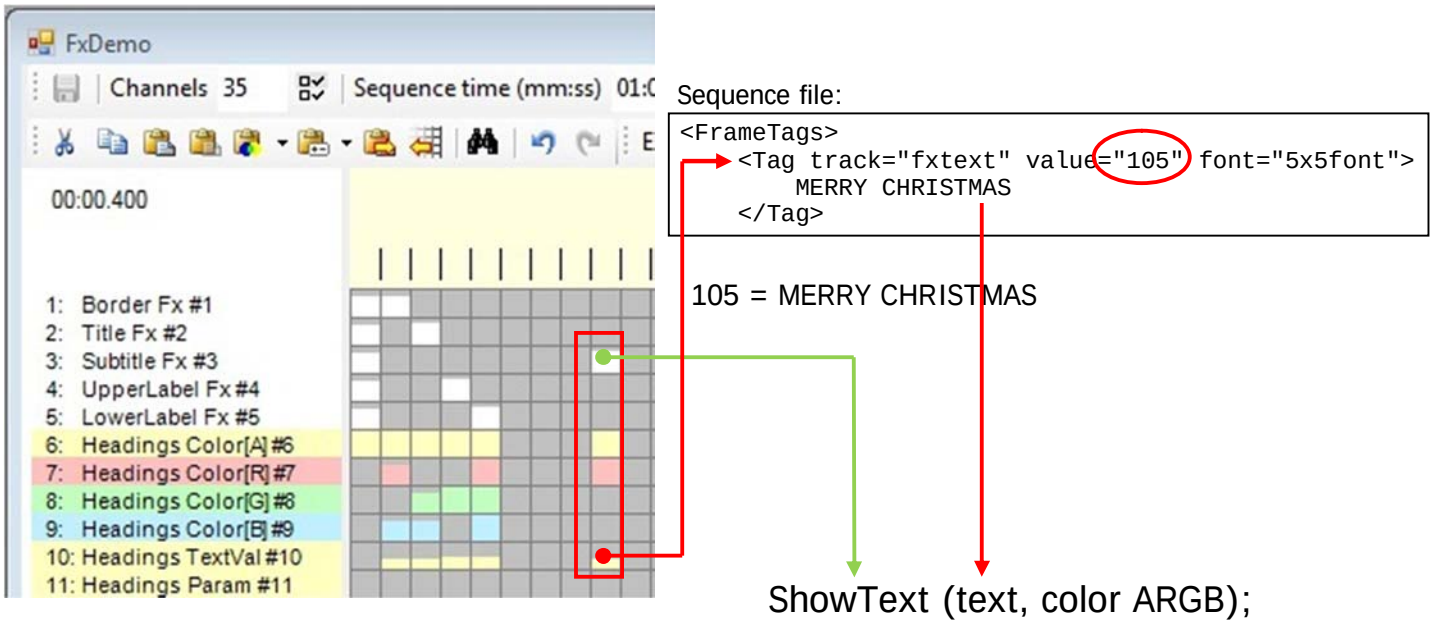
Channel cells within Sequence select the Fx function and parameters

Many of the effects functions use 2 colors: a "background" color for the "off" pixels, and a "foreground" color for the affected pixels. These color parameters are specified using 4 channels in the Sequence (representing the A, R, G, and B components of the color). I could have used 2 different color parameters (8 channels total), but since the background color typically stays the same across effects, I just used a separate function code to set the background color before other effects functions are used.

The FxGen demo Sequences run through most of the effects to show what they do. I will likely repackage many of these functions for next season, so I didn't spend much time cleaning them up at this point. Rather than going into detailed descriptions of each effect, I'll leave it as an exercise for the reader to try them out to see what they do if you are interested, or send me an email to ask about the specific ones you are interested in (this way I will avoid documenting functions that are not of interest to anyone).

2.4. FrameTags

Some effects functions require text string parameters, which are not easily represented within the Vixen cell editing window. For example, the ShowText function needs to know the text string to be displayed as well as the font, and ShowBitmap needs the path name for the bitmap file. For these, I just used a text editor to create a bunch of <FrameTags> directly within the Sequence file, and then used simple channel values to reference these within the Sequence. For example, in FxDemo a cell value of 105 refers to the "MERRY CHRISTMAS" text message. Although it's a little awkward, it's actually fairly quick and easy to add (or change) the text messages within a Sequence this way.



Text and bitmap parameters: cell value maps to a <FrameTag> to select the text string

FrameTags can also have additional attributes. For example, FrameTags used with the ShowText function can specify the font, horizontal/vertical offsets, and scrolling rates.

FrameTags are grouped according to a "track" name, which is analogous to Voices in Papagayo[14] or Tracks in a media file. The effects functions use different Track names for different types of parameters. For example, the ShowBitmap function uses the "fximage" track to select the bitmap path name, while the ShowText function uses the "fxtext" track to select the text string to be displayed, as well as the font to use.

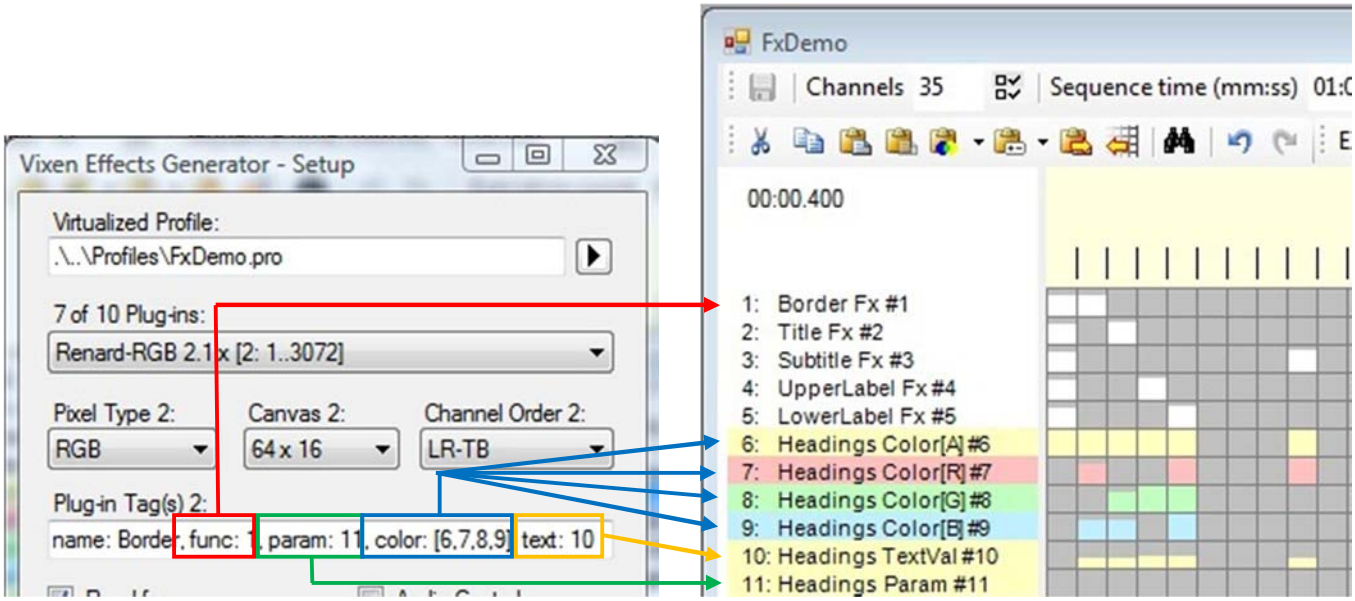
FrameTags apply to the entire Sequence unless they have start and end times specified. This allows several FrameTags to be associated with the same value over the life of the Sequence, if desired. The same value can then be used within the Sequence (for easier copy/paste), even though the actual text or bitmap itself may change over the course of the Sequence.

2.5. Parameter Channel Aliasing

Multiple props (ranges of channels) can be running different effects simultaneously, so each may need a different set of Vixen channels to hold the values (or they can be shared). For example, channel #1 in FxDemo is used to hold the Function Code for the Border panel (an RGB grid), channel #2 is used for the Function Code of the Title panel, etc, but both these panels share channel #10 as their Text parameter channel.

[14] Papagayo is a great voice and phoneme animating tool; see http://www.lostmarble.com/papagayo/index.shtml

The sample Fx functions access their parameter channels using channel "aliases", in order to allow the actual channel numbers to vary from one prop to another. The aliases are resolved using Plug-in Tags.



FxGen Plug-in Tag defines parameter channel aliasing

FxGen provides a generic Plug-in Tag mechanism to allow a text tag to be attached to each prop or virtual Output Plug-In, but it does not know anything about its meaning. The interpretation of the Plug-in Tag is left entirely up to the custom Fx functions.

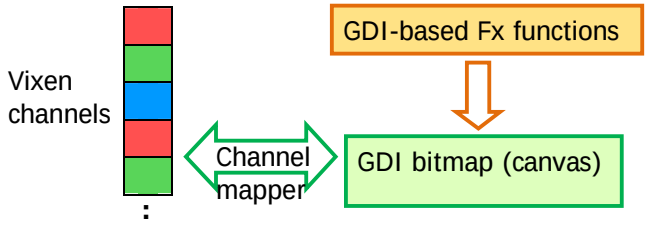
The parameter aliases are just (another) convention that I used within the sample Fx functions. I used the following aliases in the sample Fx functions:

- Name = prop name (for easier recognition within debug messages)
- Func = function code channel#
- Color = 4-channel ARGB color parameter
- Text = value of FrameTag containing text string and font
- Param = misc parameter (speed, row#, column#, counter value in the table of sample Fx functions)

By specifying unique values for the channel#s, each prop/plug-in will use its own separate Fx parameter channels. OTOH, the same Fx parameter channel can be shared by multiple props/plug-ins if desired.

2.6. Canvas Channel Mapping

I also implemented sample mapping functions to allow a rectangular grid of RGB pixels (represented as R/G/B Vixen channel triplets) to be mapped to a Windows GDI Bitmap. This provides the GDI Graphics Context or "canvas" for the effects functions, so they can use regular GDI functions to generate the effects.



Vixen channels mapped to GDI canvas

The mapping is bi-directional. FxGen reads channel values from the Sequence and maps them to the GDI Bitmap which the custom effects can then update, and the results of the effects are also mapped back to the channels within the Sequence again.

The example Channel Mapper included with the sample FxGen effects supports a number of different pixel orders, so the physical layout of the RGB pixel strings can vary. For example, left-to-right vs. right-to-left, top-to-bottom vs. bottom-to-top, and zig-zag variations are all supported.

3. Demo Sequences

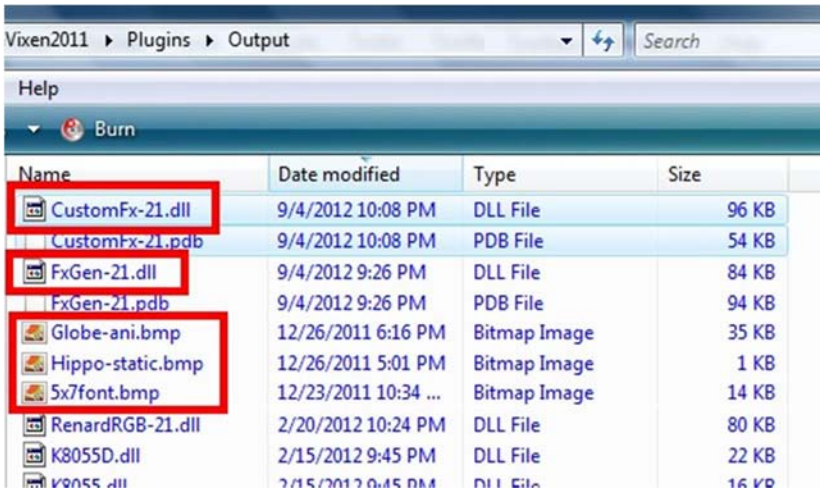
There are 2 demo Sequences included with FxGen, illustrating the 2 main ways to use it. These can be run with any installed copy of Vixen 2.x using the steps below.

Step 1 – Install Plug-in files

Copy the FxGen and sample effects DLLs and additional BMP data files to Vixen's Plugins\Output folder. These are included within the Plugins folder in the release package.

The files to be copied are as follows:

- FxGen-21.dll or FxGen-25.dll
- CustomFx-21.dll or CustomFx-25.dll
- GridEditor-21.dll or GridEditor-25.dll
- Globe-ani.bmp
- Hippo-static.bmp
- VixenFx.bmp
- 5x5font.bmp
- 5x7font.bmp



Choose the "-21" or "-25" version of the DLLs according to which version of Vixen you are using. The PDB files can be omitted unless you want stack trace info after a crash.

The BMP files provide graphics for various sample effects, with the *font files used to display low-res text.

The Grid Editor is also included, since there were a few tweaks in order to get it to be happy with FxGen.

Step 2 – Copy Sequences and Profiles

Copy the demo Sequences and Profiles files to the appropriate Vixen subfolders. The VIX files go in Vixen's Sequences folder, and the PRO files go into Vixen's Profiles folder, just like usual.

Step 3 – Run the Sequence

The demo Sequences can be run using the normal Vixen menu or toolbar buttons. The demos themselves are described in more detail below.

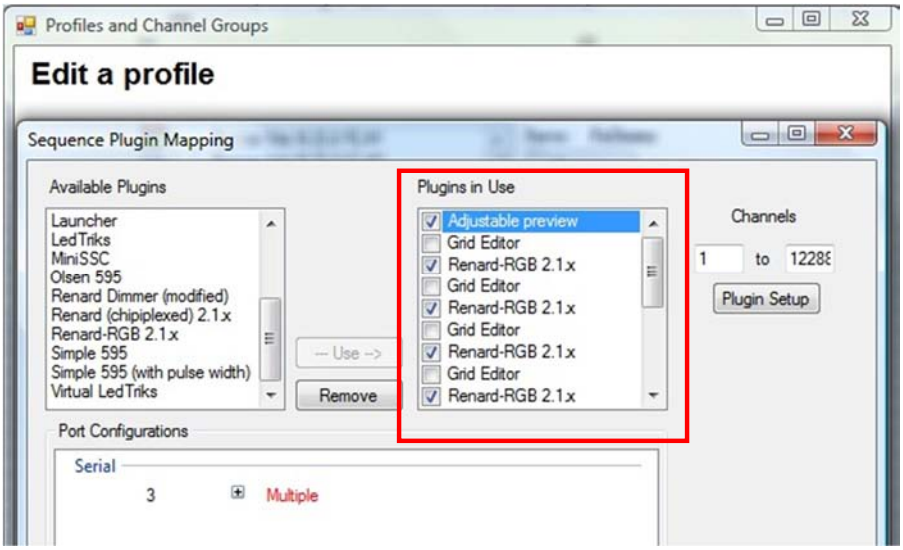
3.1. Pure Virtual Demo

The first demo, FxDemo, is intended to demonstrate most of the sample Fx functions. It uses a virtual Profile to control about 12K channels, leaving the host Sequence nearly empty. This represents the most aggressive way to use FxGen, and is appropriate when no manual editing of cells within Vixen is desired - the effects are generated entirely dynamically using virtual channels, so they aren't stored anywhere. Normally a 3½ minute Sequence file with 12K channels would be about 70 MB of data, but the FxDemo Sequence file is only 70K because none of the regular channels values are stored there.

Virtual Profile

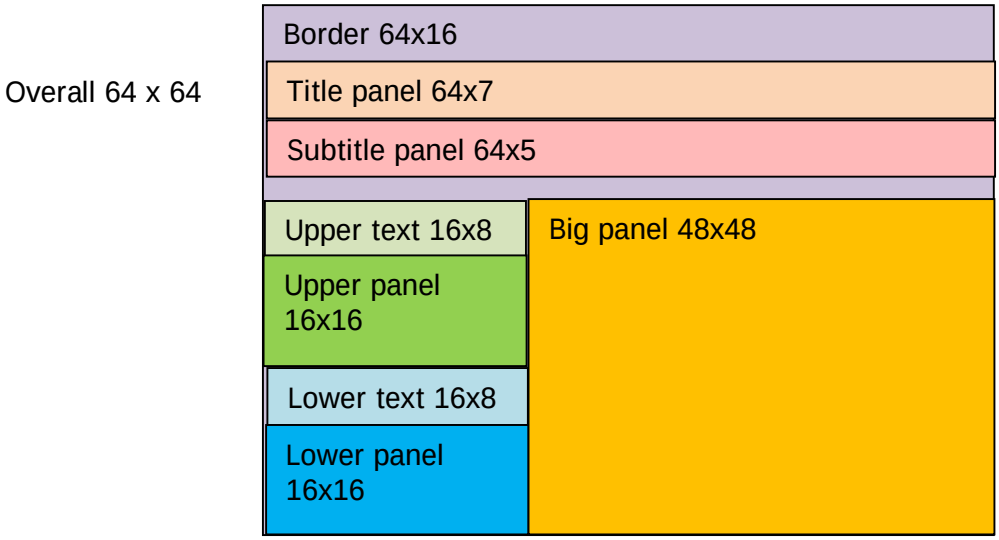
The virtual Profile contains an Adjustable Preview plug-in covering the entire set of 12K channels, so it will be laggy or jumpy during playback on slower machines ☹. Its purpose is to display the output of the various Fx functions on the screen as they are invoked within the Sequence. The Preview bitmap is arranged as a 64x64 grid, with 3 channels per pixel (R, G, B), for a total of 3 x 64 x 64 = 12,288 channels.

Several Grid Editor and Renard-RGB plug-ins are also defined within the Profile. These are overlayed onto various channel ranges, in essence subdividing the overall grid into "panels". The Grid Editors allow individual panels to be watched separately (RGB color fidelity of the Grid Editor is more accurate than Adjustable Preview). These contribute further to the laggy/jumpy playback behavior ☹, however, so only 2 have been enabled within the demo Profile (you can turn these on/off as desired). The Renard-RGB plug-ins are examples of hardware plug-ins that could also be overlayed onto various channel ranges to send the output to real (physical) controllers and props. These are disabled within the demo Profile, but could be enabled or replaced if you wanted to see some actual blinky-flashy output during the FxGen demo.



Multiple Grid Editor and Renard-RGB plug-ins, one for each "panel"

The organization of the channel ranges into panels is shown below, with the first channel being in the upper left corner:

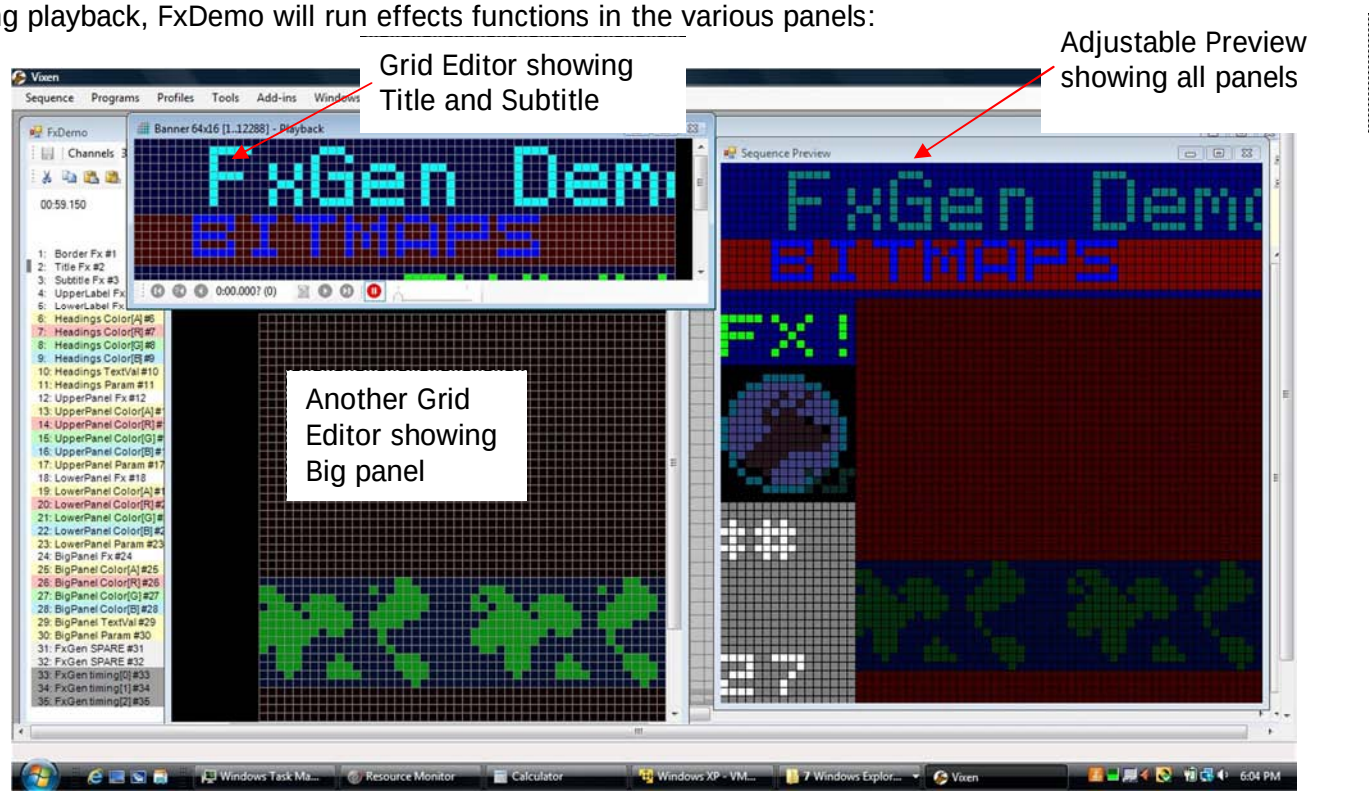


FxDemo panel layout

There is nothing special about this Profile – I built it using just the regular Vixen Profile Management functions to add the channels and plug-ins, and then used the Grid Editor to define the Preview bitmaps for itself and for the Adjustable Preview[15]. I suppose this resembles a miniature Million Dollar Home Page[16]. ☺

FxGen will work with monochrome as well as RGB props. For example, the Lower panel is a monochrome grid, which would be suitable for controlling LED or incandescent strings, rather than smart RGB nodes.

During playback, FxDemo will run effects functions in the various panels:



FxDemo playback

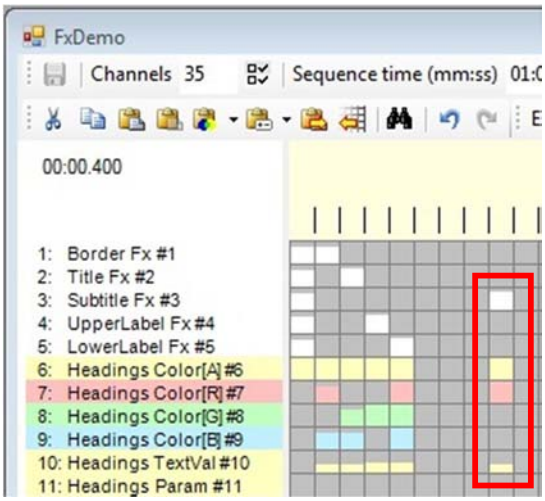
The Subtitle panel is used to describe the category of effects being displayed at various times - for example, Text, Bitmap, etc, and the Big panel is used as the main display area for the currently active effect. The Lower panel runs a Countdown effect to display a counter (elapsed time within the Sequence), so if you see an effect that you would like to use, you can see where it occurs within the Sequence timeline and then go back and copy or modify it after playback stops.

Sequence

The Sequence itself contains a small number of control channels, just enough to trigger the sample effects. Several channels are used for each panel, and some of the channels are shared across panels. For example, each panel has its own "Macro function" channel to select the effect, but the text panels all share parameter channels for color, text selection, etc. These are reflected in the channel names.

For any given effect, there is really not much to see in the Sequence - just a few cells with pre-determined values, placed at various times within the Sequence. An example is highlighted in red in the screen shot below.

[15] Technique is described on pages 48 - 49 of http://downloads.eshepherdsoflight.com/Howidid-DumbRGBPixelGrid.pdf
[16] http://www.milliondollarhomepage.com



Typical effect function invoked within Vixen Sequence

The interpretation of the control channel values was described earlier in the Sample Effects section. Fx control channels are associated with panels using the Channel Aliasing mechanism as described earlier.

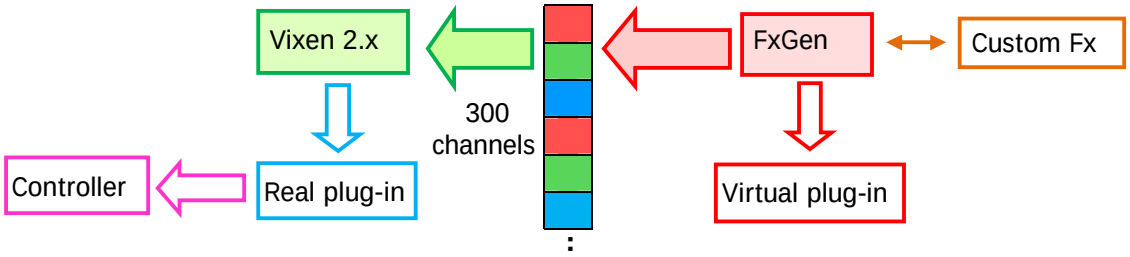
3.2. Mixed Mode Demo

The second demo is FxMixedDemo, and shows how to combine FxGen output with regular channel output to produce a "mixed mode" Sequence. This technique can also be used to record FxGen output back into the Sequence file, so it can then be edited by hand or played back with regular Output Plug-ins in the absence of FxGen - analogous to "compiling" the effects into "executable" channel data, whereas FxDemo was analogous to a straight interpreter.

Layout

The FxMixedMode demo defines a 10 x 10 RGB grid prop (300 channels). This is a "real" grid within the Sequence itself (there are actually channels defined for it), and there is a Grid Editor plug-in defined on those channels so you can see the effects on the screen. You could also connect a real controller and lights to it if you wanted some real blinky flashy.

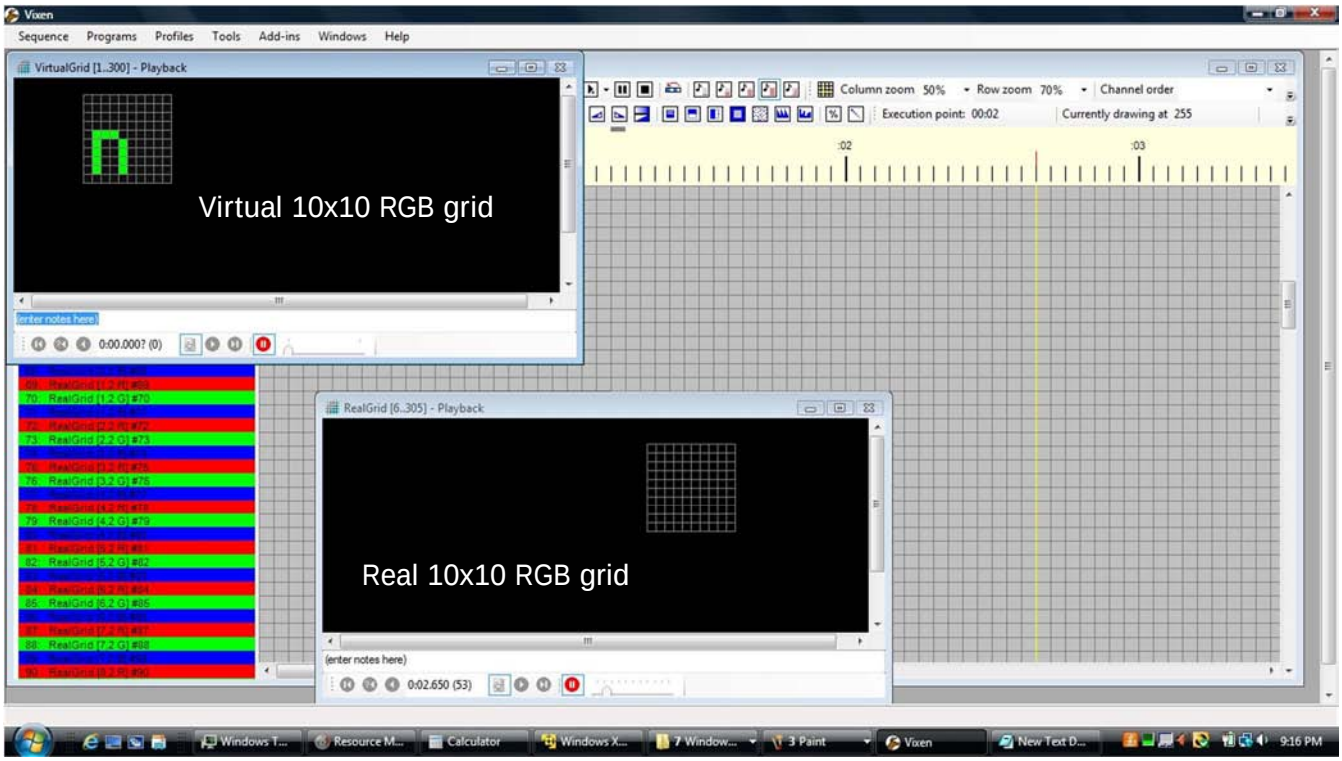
Within the FxGen virtual Profile, there is another Grid Editor plug-in overlayed onto the same channels. This, in essence, creates a "virtual copy" of the same RGB grid. The virtual grid is updated by the Fx functions, and since it maps to the same channels, the effects will also be visible on the real grid. This allows the effects to be "recorded", with the "virtual" grid as the source, and the "real' grid as the destination.



Recording from virtual to real plug-ins using overlayed channels

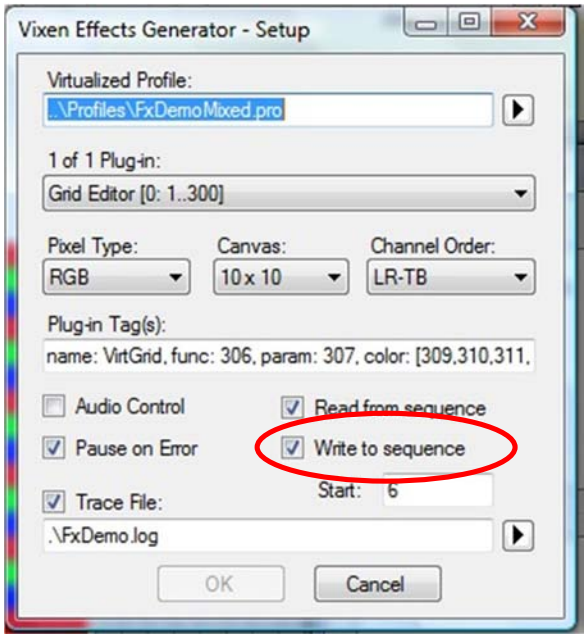
During FxMixedMode playback, you will see 2 Grid Editors. The virtual grid channels are displayed in the upper left Grid Editor, and the real grid channels are displayed in the lower middle Grid Editor.

The virtual grid will display some scrolling text because its content is being generated dynamically by FxGen. The real grid will be blank because its channel data is blank. The grids are fairly small, so playback will not be as laggy as FxDemo. ☺



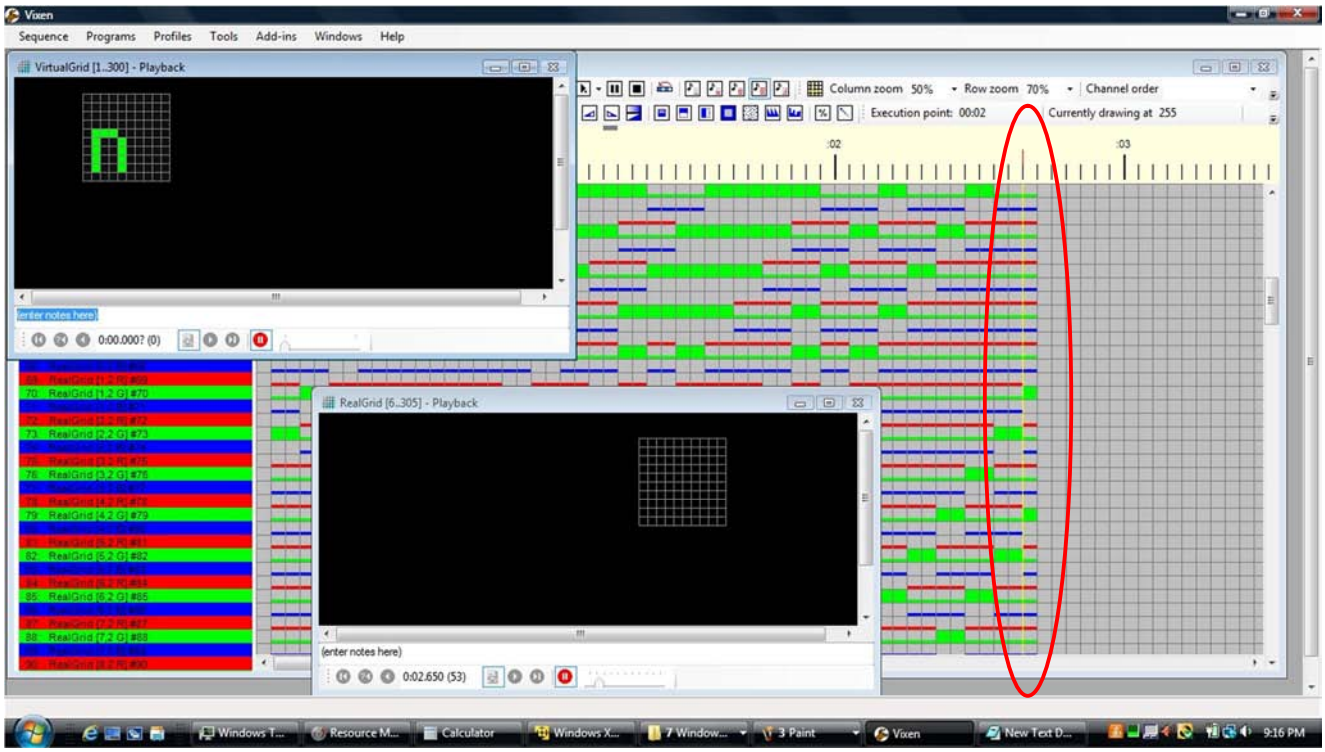
FxMixedMode playback – real grid channels are blank

Now suppose you wanted to "record" the effect back into the Sequence file, using the real grid's channels. To do this, you would use the Attached Plugins menu to go the the FxGen Setup window, then turn on the Write to Sequence option:



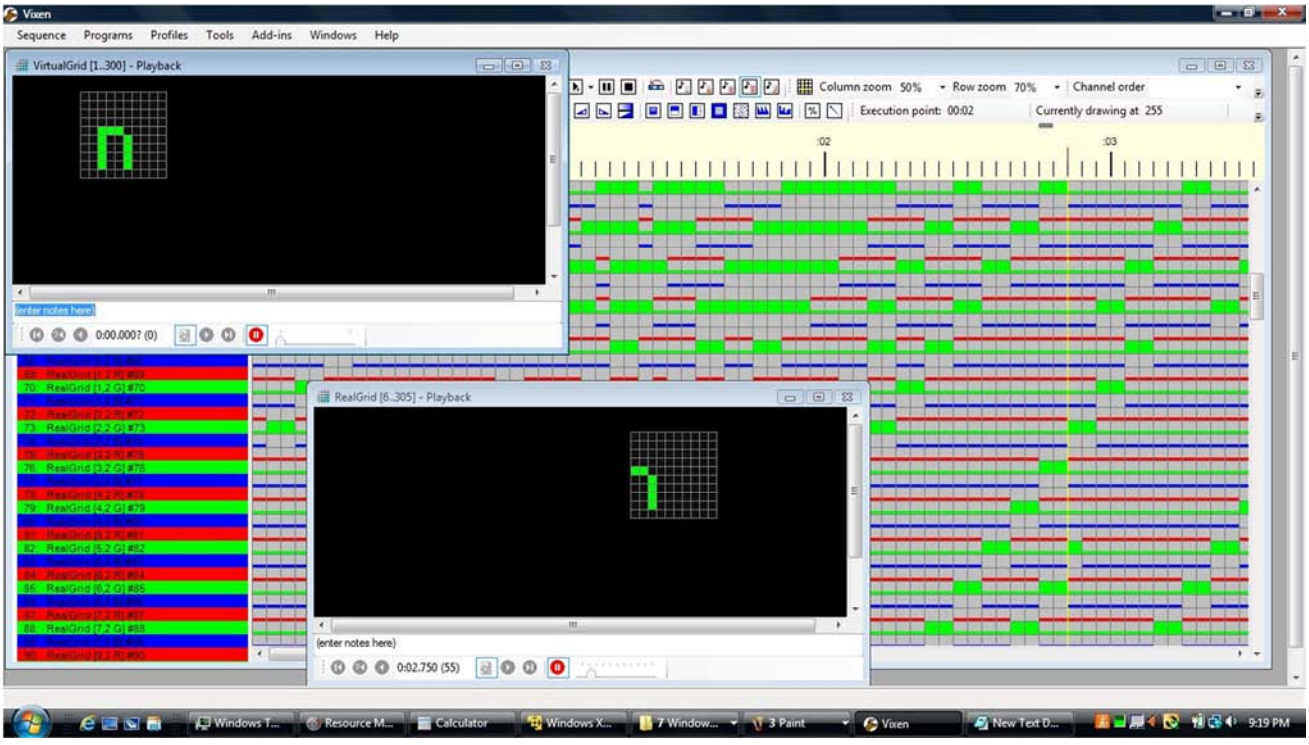
FxGen Setup – recording

Now if you replay FxMixedDemo, you will see the effects being recorded into the Sequence:



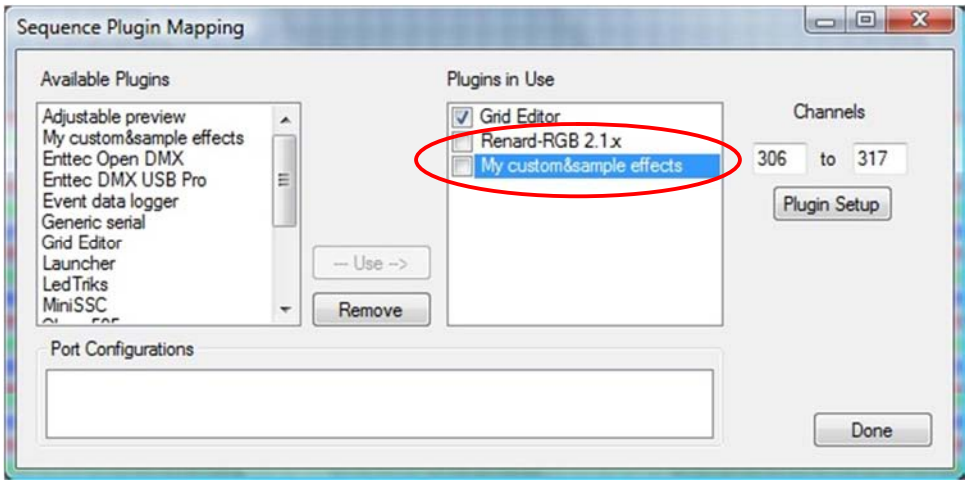
Effects channels being recorded into Sequence during playback

After the effects have been recorded, if you play back the demo a third time, you will see the effects in both copies of the grid:



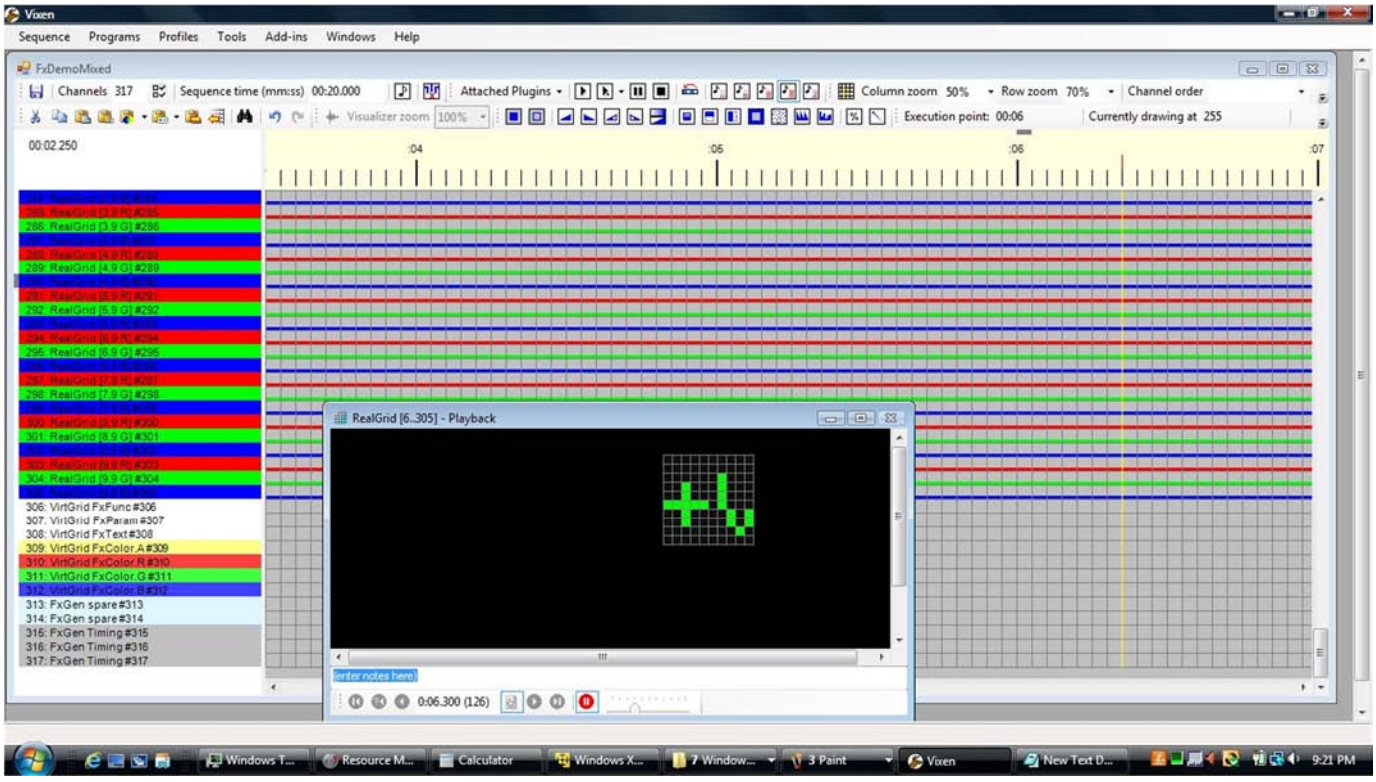
Effects displayed in virtual grid, recorded effects displayed in real grid

After recording, if you wish to "freeze" the effects (for the purposes of editing them further by hand, or perhaps exporting to another Sequencing tool), then you would turn off the Write to Sequence option again. If you were finished with the virtual profile altogether, then you could also just turn off FxGen:



Turn off FxGen when finished recording or to disable virtual effects and props

With FxGen disabled, the Sequence behaves like a regular static Sequence. For FxMixedDemo, only the real Grid Editor would remain, and the Sequence could be played without the use of FxGen:



Normal playback with FxGen effects and virtual props disabled/removed; real plug-ins remain

You could then use the real Grid Editor or any of the regular Vixen cell editing tools for further editing. This allows FxGen to be used for a quick "first draft" of effects, and then other tools can be used for "fine tuning".

There are various other ways to use FxGen and virtual vs. real channels, but I'll leave it as an exercise for the reader to explore. ☺

4. CUSTOM USAGE

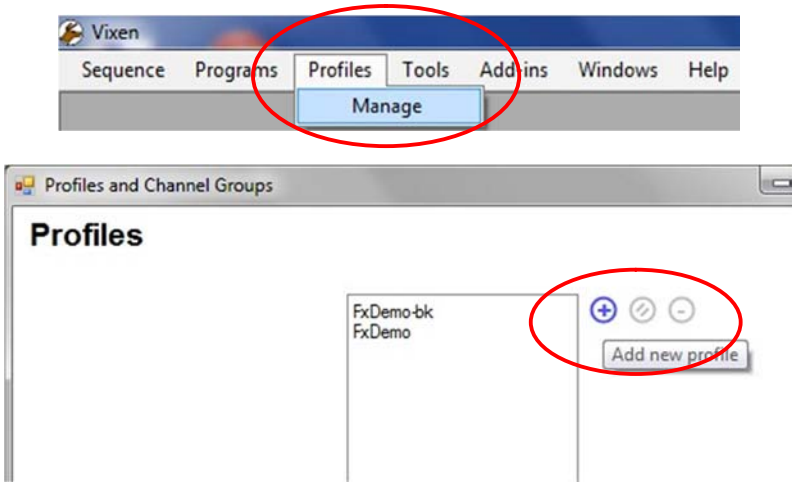
To use FxGen in your own Sequences, you could copy/paste from the FxGen demos or you could create a new Sequence manually. This section will describe how to manually add FxGen to a new Sequence, which is how I created the demo Sequences. This explanation will (hopefully) make more sense if you've read the Overview and Concepts section first, and played the demos.

Adding FxGen to a Sequence is similar to adding other Output Plug-ins, except for an extra virtual Profile. I suppose the config options are probably also a little more complicated.

Step 1 – Create Virtual Profile

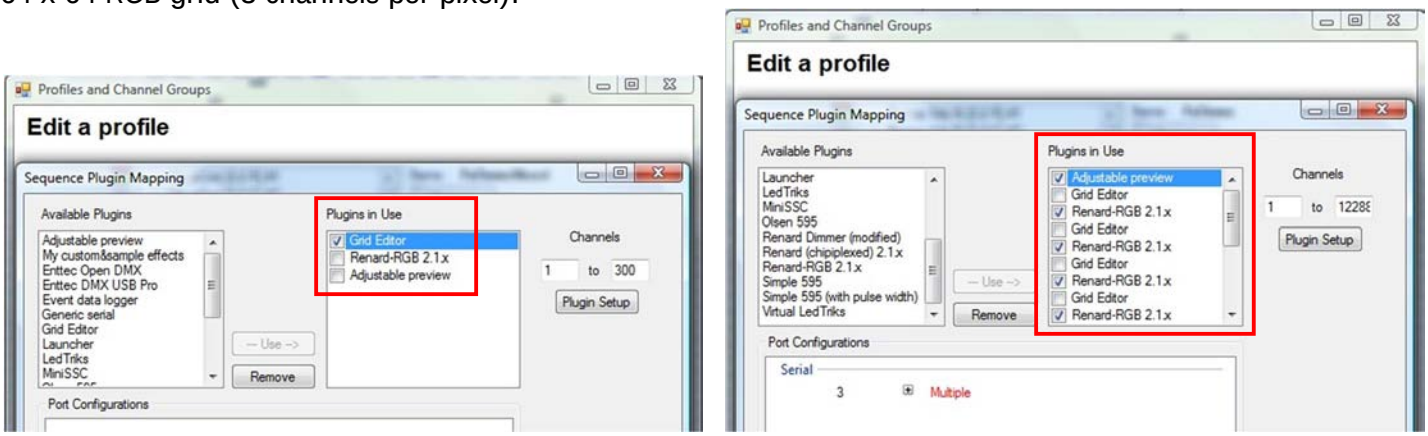
The first step is to create a virtual Profile for FxGen. This Profile is not the real Profile for your Sequence; it should only contain channels and plug-ins that will be controlled virtually by FxGen.

Use the regular Vixen Profile Management tools to create the Profile:



Vixen Profile Management menu

Next, add channels and plug-ins to the virtual Profile. Ranges of channels will be mapped to a GDI canvas for the Fx functions to draw on, so there should be enough channels for your desired canvas size, taking into account any overlaps. For example, I added 12,288 channels to the FxDemo Profile, which was enough for a 64 x 64 RGB grid (3 channels per pixel).



Adding plug-ins to the virtual Profile

Then add plug-ins to the virtual Profile, either to show the effects on the screen or to actually control the lights. I usually add an Adjustable Preview to display the entire set of channels in one window, and then some additional Grid Editors to visually represent individual props or controllers. For FxDemo, I added 8 copies of the Grid Editor to represent the 8 individual panels in the demo. I also added some Renard-RGB plug-ins as placeholders, in case I wanted to connect lights to some of those panels.

The Grid Editor is also a quick way to set up Preview bitmaps for RGB grids[17]; you can add it temporarily to draw the grid shape and set channel names and colors, and then disable or remove it if no longer needed. I usually set a unique name for the first channel of each RGB grid prop before running Grid Editor Setup, because it will copy that name to all the other channels for that grid. This avoids a lot of channel setup work.

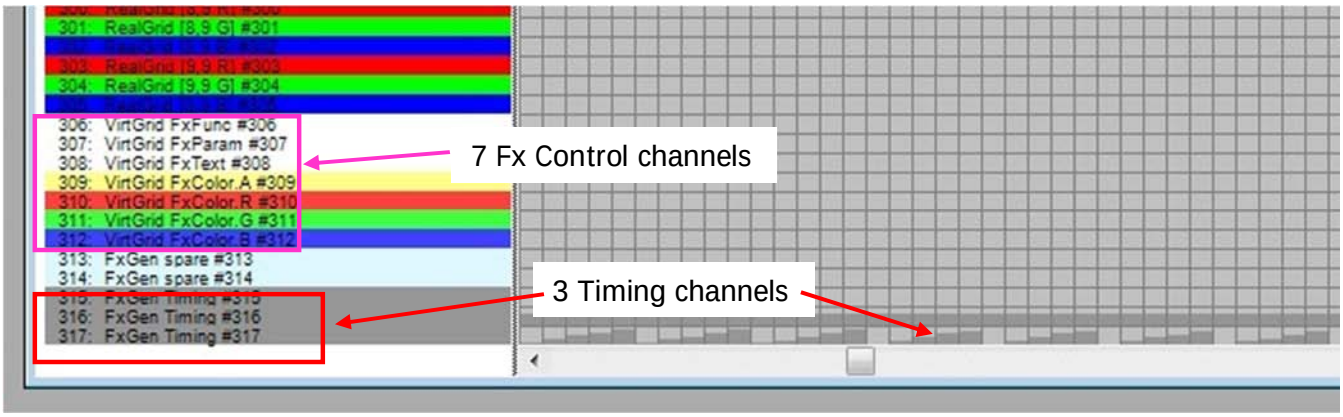
Save the virtual Profile, and also make a backup copy in case something happens to your first copy.

Step 2 – Create Real Sequence/Profile

The next step is to create your real Sequence and/or Profile. As with regular static Sequences, a Profile is optional, but useful if you will be using the same layout with multiple Sequences. If you do use a Profile, it will be different than the virtual Profile created in the previous step.

If you want to be able to record, export, or otherwise edit Fx channel output, you'll need to define real channels to hold all those channel values. RGB nodes usually require 3 channels each, while monochrome nodes or strings would normally just be one channel each. For example, the 10x10 RGB grid in the FxMixedMode demo requires 10 x 10 x 3 = 300 channels to hold its channel values. As with the virtual Profile, I find it easiest to just set the name of the prop's first channel and then use the Grid Editor later to draw the Preview bitmaps and automatically set the names and colors for the remaining channels.

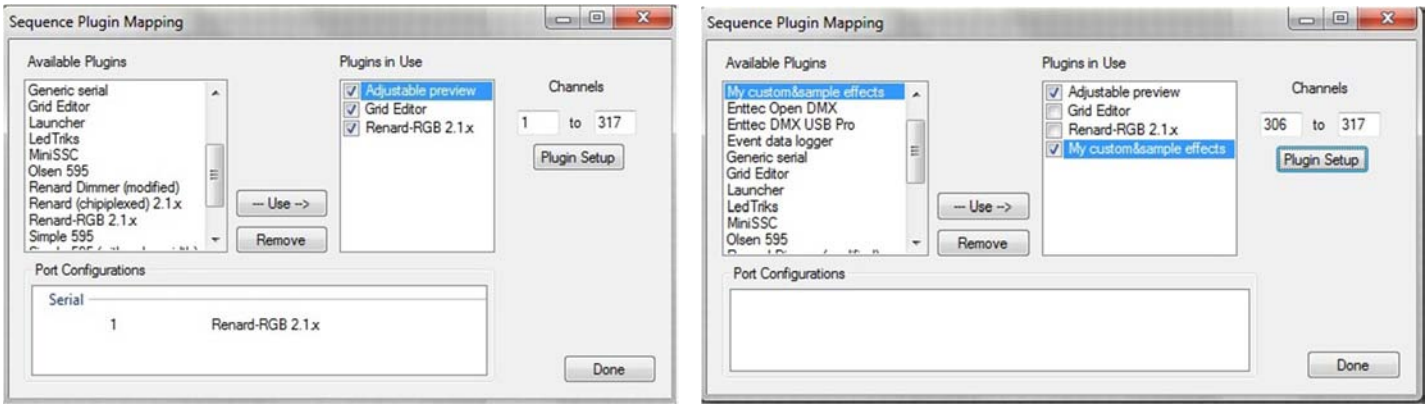
In additional to any physical channels needed for record/export, you'll also need to add some Fx control channels. The actual number of channels will depend on how many virtual props you have and how your control channel aliasing is set up. The aliasing technique I used with the sample effects takes 7 control channels per virtual prop/controller/panel (function code + text + ARGB color + misc parameter = 7), but in FxDemo I shared some of those between panels so it only took 30 control channels instead of 8 x 7 = 56 for all 8 panels. If you're using those same FxGen sample effects, start with 7 separate control channels for each prop and then you can re-alias them for sharing later if desired.



Fx control and timing channels

FxGen needs 3 additional channels for a "timing track" within the Sequence, so add another 3 channels for that. These will automatically be placed at the end of the channel range assigned to FxGen.

[17] See pages 48 – 49 of http://downloads.eshepherdsoflight.com/Howidid-DumbRGBPixelGrid.pdfgrid.pdf



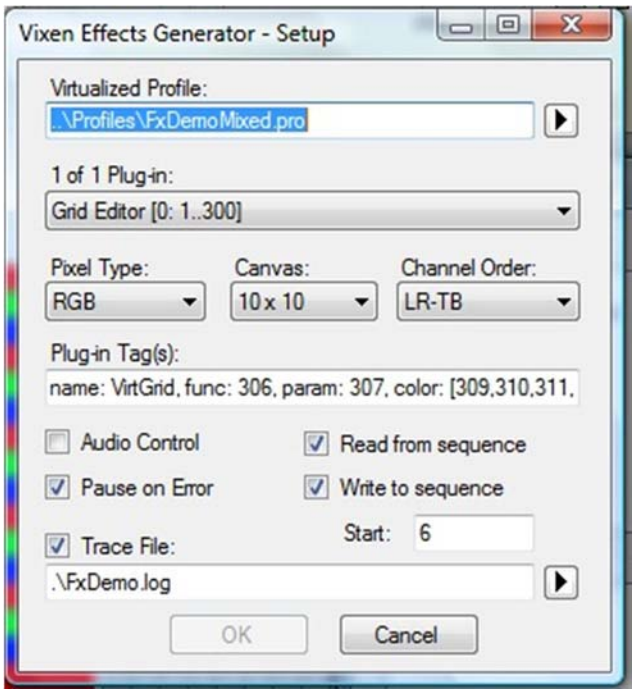
Add normal plug-ins then custom effects

After adding the channels, add the normal Output Plug-ins that you intend to use with the Sequence and associate them with the appropriate channel ranges. Then add the custom effects plug-in (ie, CustomFx.dll) and associate it with the Fx control channels. (FxGen itself cannot be added to a Sequence or Profile, as explained earlier).

At this point, it's probably wise to save the Sequence/Profile in case you need a backup for a later step.

Step 3 – Configure FxGen

The next step is to configure FxGen. With the sample effects plug-in selected, click on the Plugin Setup button. This will open the FxGen Setup window:



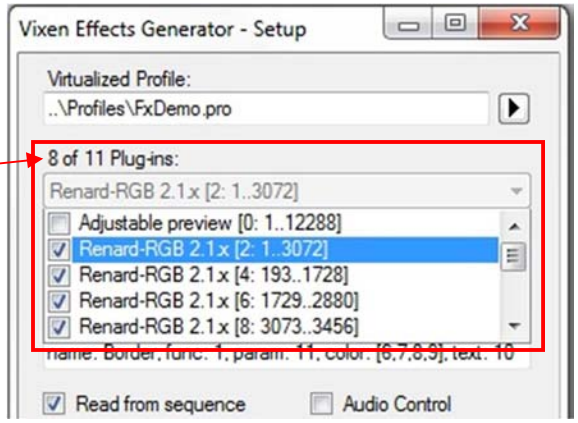
1. Choose virtual Profile
2. Choose virtual plug-ins
3. Set canvas channel mapping
4. Set parameter channel aliasing
5. Select FxGen processing options
6. Save config info

FxGen Setup window and workflow

The workflow for FxGen setup flows top-to-bottom. First, choose the virtual Profile by entering its path name or clicking the ▶ button to browse to the file. Relative path names start in the Plugins\Output folder, so you would normally want to specify an absolute path, or use "..\Profiles\" at the start of the path name to move to the Vixen Profiles folder.

FxGen will open the selected virtual Profile and take a look at which plug-ins are defined in there. It will use only the enabled plug-ins to populate its list of available plug-ins. Click the drop-down arrow to see this list:

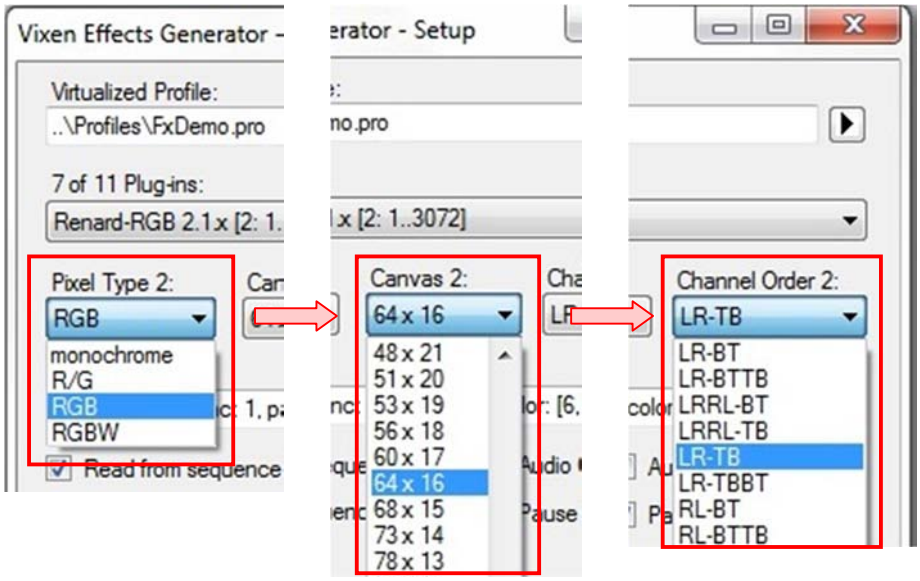
Number of plug-ins currently selected and total number available (enabled) in virtual Profile



Enabled virtual plug-ins shown in FxGen drop-down plug-in list

The plug-in name, channel count, and range will be displayed for each plug-in. Turn on the checkboxes of the plug-ins that you want to be controlled by FxGen. The number of plug-ins currently selected will be shown above the list. Turning a checkbox on or off will change the displayed count. FxGen requires at least one plug-in to be selected before you can Save the config info, otherwise there is nothing for it to do.

After choosing the plug-in(s), the Canvas mapping options must be selected using the 3 drop-down lists below that:



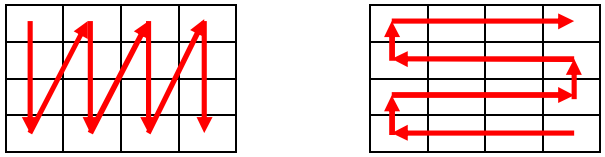
Canvas channel mapping options: Pixel type, Canvas dimensions, Channel order

The Canvas mapping options apply separately to each individual plug-in. Clicking on a plug-in will update the labels and choices available, so you can set these after selecting each plug-in (the number at the end of the labels indicates which plug-in is currently selected). The Canvas mapping options must be selected in a left-to-right order, because each option affects the choices available to its right.

First choose the Pixel type. This determines how many channels are assigned to each pixel in the GDI canvas (1 channel for monochrome, 3 for RGB, etc), and therefore also affects the number of pixels and canvas size that the given plug-in's channel range can hold.

Next choose the Canvas dimensions. FxGen assumes that as many pixels as possible will be used, so it generates a list of possible dimensions based on that. For example, 300 channels are enough for 100 RGB pixels, so one of the Canvas choices would be 10 x 10.

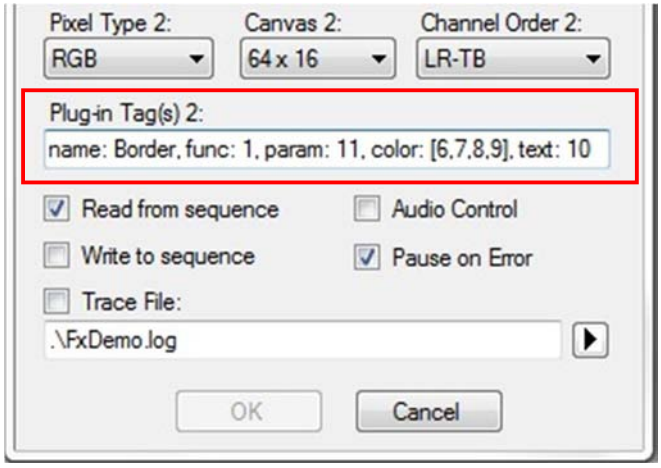
After the Pixel Type and Canvas dimensions are selected, the Channel order comes next. This option determines the actual order in which channels are assigned to the Canvas, so you can choose this option according to how you are actually going to string the lights. Various combinations of left-right and up-down are offered, including some zig-zag options, with the faster-changing dimension on the right. A couple of examples are shown below:



Example Channel orders

BTW, FxGen itself does not understand the Canvas channel mapping options; it only provides a UI to display them, and the actual enumeration and handling of the options is provided by the custom effects DLL. I only used a couple of these options myself, so there are probably some in there that don't work correctly. Please test before you string your lights!

After Canvas mapping, the Parameter channel aliasing needs to be set up. As with the Canvas mapping options, the Parameter aliasing also applies to each plug-in separately. Because the aliasing mechanism was completely arbitrary and implemented outside FxGen itself (within the custom Fx functions), the UI is just a text box:



Parameter aliasing and misc FxGen options

The FxGen sample effects expect 5 tags to be included within the aliasing text field. As explained earlier, the meaning of these tags is as follows:

- Name = prop name (for easier recognition within debug messages)
- Func = function code channel#
- Color = 4-channel ARGB color parameter (4 numbers are needed, but they must be consecutive)
- Text = value of FrameTag containing text string and font
- Param = misc parameter (speed, row#, column#, counter value in the table of sample Fx functions)

Below the Parameter aliasing tag are several general processing options. These apply to FxGen overall, not to specific instances of the virtual plug-ins. These options are as follows:

- Read from sequence — Determines whether FxGen will read channel data at the start of each frame. Reading channel data and mapping to a GDI bitmap adds processing overhead, so this option allows that to be skipped if it is not needed. However, since the current sample Fx use channels for their parameter values, this step is actually required, so this option is hard-coded On within FxGen.
- Write to sequence — Determines whether FxGen will write dynamically generated channel values back to the Sequence. This adds processing overhead and requires storage space within the Sequence, so this option allows the writes to be skipped. This option is useful for selective Fx recording (discussed as part of FxMixedMode).
- Audio control — Determines whether FxGen will play the audio instead of Vixen. This option was intended to compensate for laggy/jumpy behavior, but never implemented.
- Pause on error — Tells FxGen whether to display a popup error and stop. This is useful for debug and test purposes, but should be turned Off for live shows.
- Trace file — The checkbox tells FxGen whether to generate Debug info, and the text box and browser button allow a path name to be selected. If there is no trace file but the checkbox is On, popup Debug messages will be displayed instead. This option adds I/O overhead, so it should normally be turned off.

After selecting the general options, click the OK button to save the changes. Clicking the Cancel button will discard changes, but FxGen will prompt for confirmation first if there are unsaved changes.

When changes are saved FxGen will also write a <LastModified> date/time stamp to the Sequence, which can be helpful in keeping track of where something changed since Vixen updates the timestamp of certain files even though there are no user-initiated changes.

Step 3 – Invoke Effects Functions

To call an effect function within the Sequence, just set the aliased Function code channel to the appropriate value at the desired place, and then set any additional parameter channels as needed. For example, in FxDemo channels 3, 6, 7, 8, 9, and 10 must be set in order to display an effect in the Subtitle panel.

Set channel values to invoke an effect

For the sample Fx functions packaged with FxGen, from 1 to 7 channels will need to be set. These cells can be set manually, or using any other cell editing tools and techniques. For example, you could save a number of Vixen Routines ahead of time, and then just paste them into your Sequence when desired, similar to calling a subroutine within a program, or you could apply combinations of Add-in tools, Arithmetic or Boolean paste functions, etc to generate the cell values automatically. For text parameters, you will also need to add a <FrameTag>, as described earlier. There is no UI for this, so just use a text editor on the Sequence file.

Since the effects are just a group of cell values, you can use the regular Vixen playback tools to run the effect and see how it looks, then pause and make adjustments, etc.

Step 4 – Creating New Effects

There are several ways to add new effects. I'll give a few examples here, but there are other possibilities.

Bitmaps

If the effect can be represented by a series of bitmaps, the simplest way is to just create one or more Bitmap files and then just use the ShowBitmap effect to animate those during the Sequence. I wouldn't use this for streaming video, but it can be used for simpler low-res graphics.

ShowBitmap is quite powerful because of its animation attributes. FxDemo shows a couple of examples:

- A "sprite" or icon can be scrolled horizontally and/or vertically within a larger grid/window.
- A larger bitmap can be scrolled within a smaller grid/window.

The first approach gives a moving object effect, like the floating hippo in FxDemo.

The second approach gives more of a rotating object effect, like the globe in FxDemo. I used the globe bitmap for the rotating Earth effect on the Snow Globe in one of my sequences[18] and it worked well.

Another variation would be to just display a series of bitmaps, one after another sequentially within the Sequence. This could be used to simulate fades, flames, or any other type of movement. The bitmaps could be captured using your favorite graphics tools, and then cropped to fit on the RGB grid. (An example was shown in the original Grid Editor documentation[19]).

ShowBitmap does not currently handle animated GIFs, but that would be yet another way to do it. However, you can get the same effect using the other approaches described above.

There are probably other creative ways, which I'll leave as an exercise to the reader. ☺

Writing New Functions

A more aggressive way to create new effects is to write new functions. I implemented simple example/demo effects that were sufficient for my Sequences, but a lot fancier effects could be implemented. After experimenting with the FxGen examples you may decide you want to add some new effects. That is actually quite easy to do if you are familiar with .NET programming and GDI functions. The source code for FxGen and sample effects functions are provided in case anyone wants to do that. However, I ask that you share any new effects with the rest of the DIYC community, so we can all benefit and build on each other's efforts.

To add new custom effects, you'll need a developer tool such as Microsoft Visual Studio 2005 or equivalent. I've been using #develop (SharpDevelop)[20] 3.2 because it is Open Source and supports non-MS tools also, but either will work. (Some of the Project settings might need to be adjusted to work with Visual Studio, though). FxGen and Vixen 2.x are compiled to run on .NET 2.0, so you'll need that also.

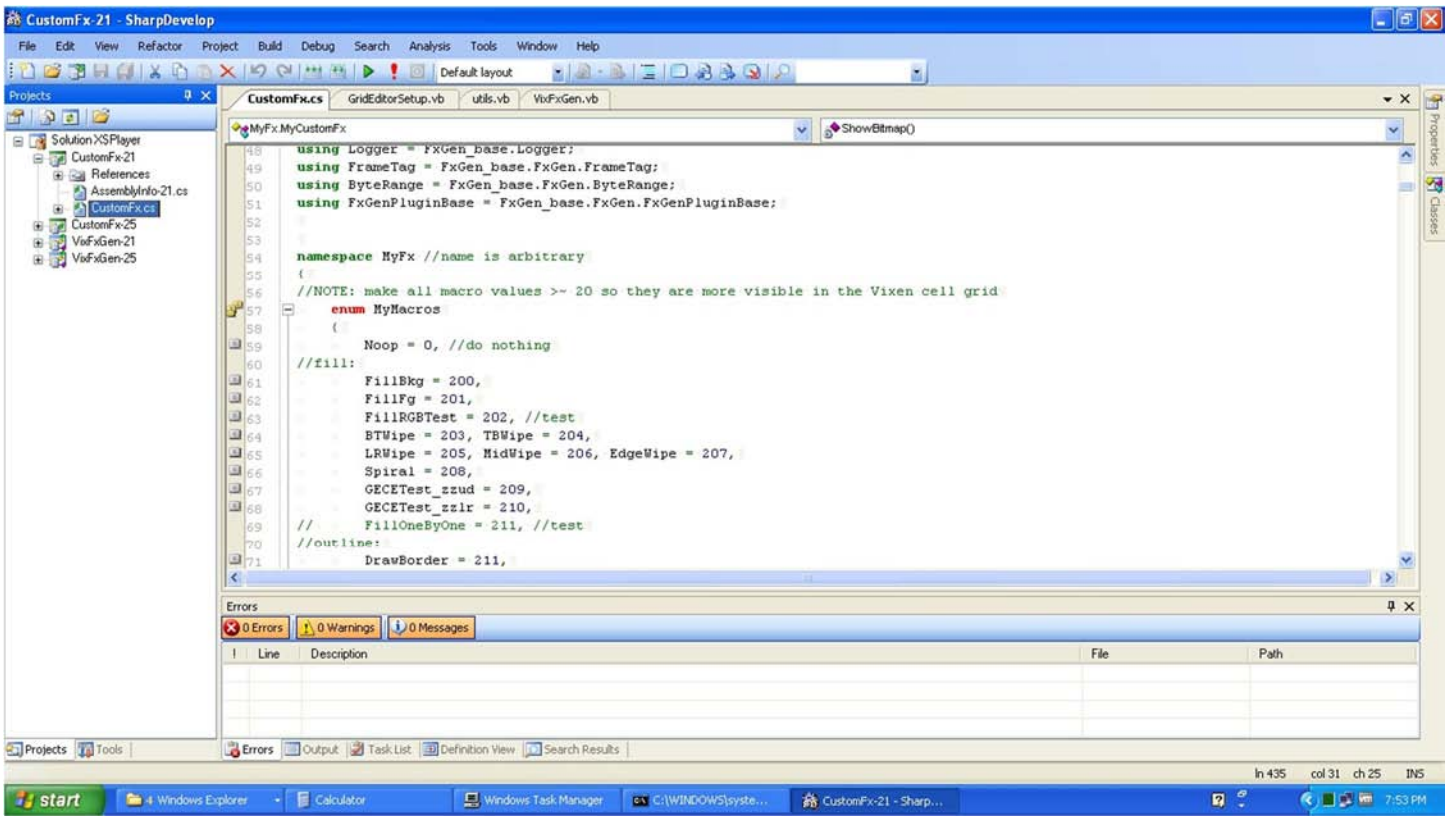
The FxGen source code consists of one Solution file containing several Projects. There are 2 Vixen plug-ins, FxGen itself (which can't be instantiated), and then CustomFx, which is the sample effects class derived from FxGen. There are 2 Projects defined for each of these plug-ins: one for Vixen 2.1 and the other for Vixen 2.5. However, there is only one source for each plug-in and conditional compiles used to get the Vixen versions.

[18] See 1:22 – 1:28 in http://www.vimeo.com/34711867
[19] See pages 53 – 54 of http://downloads.eshepherdsoflight.com/Howidid-DumbRGBPixelGrid.pdfgrid.pdf
[20] http://www.icsharpcode.net

The Solution is organized as follows:

```
XSPlayer
    VixFxGen-21
    VixFxGen-25
    CustomFx-21
    CustomFx-25
```

I tend to use VB for UI stuff, so FxGen itself is written in VB (because of the Setup UI). Since CustomFx is mostly just function calls and supporting logic, it is written in C#. You can convert them to whatever language you want, though.



Solution and Projects open in #develop

To add new effects, you can clone CustomFx and start your own "library" of effects, or you can just add them directly to CustomFx. Cloning requires a couple of extra steps, like changing the name of the DLL and changing the MyName property so you can tell them apart in the Vixen plug-ins list; otherwise it's the same.

To add an additional effect, use the following steps:

1. Choose a new function code
2. Add the new value to the MyMacros enum in CustomFx.cs
3. Write a function to generate the effect
4. Add a case to the switch statement in the FxGen function in CustomFx.cs to call the new function
5. Compile the DLLs and copy them to Vixen's Plugins\Output folder
6. Using a text editor, add a <FrameTag> for the new function to FxDemo or your own Sequence
7. Set a control channel to the new function code and then play back the Sequence to watch the effect

For the third step, you can look at some of the other effects functions to see what they do. They are somewhat messy, but shouldn't be too hard to figure out (send me questions if you need more info).

The sixth step is a paranoid check; CustomFx compares function code values in the Sequence to its compiled values to ensure that the values used in the Sequence do not get out of sync with the compiled values.

If your new effect requires additional parameter channels, you'll also need to modify the channel aliasing within CustomFx.cs. I can provide more details if you want, but I don't want to include all that info here if nobody needs it. If you modify the channel aliasing, then you'll also need to go back and update the Plug-in tags using the FxGen Setup window (described earlier).

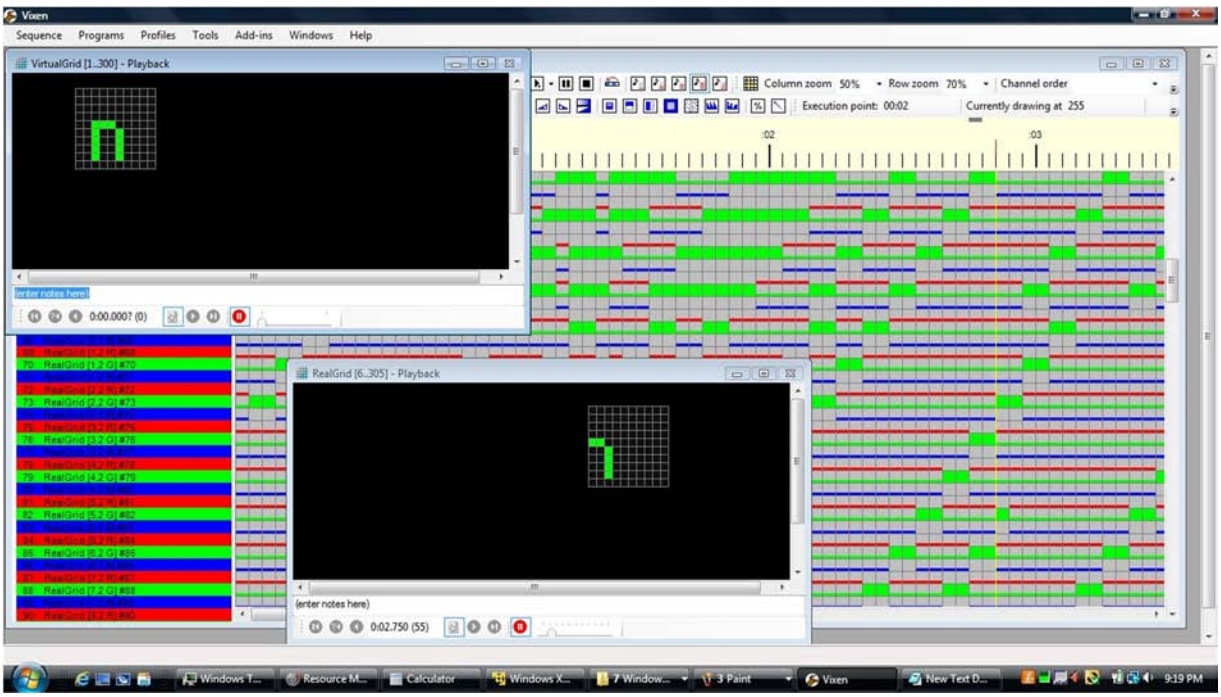
5. Bugs and Limitations

FxGen is a work in progress, so there are bugs, limitations, and missing features. It worked well enough for my style of sequencing, but it may not work as well in other environments. If you try it and run into problems, let me know and I'll try to take a look. I can probably fix the "easy" stuff, but the more extensive changes will need to wait until next season. ☹

This section lists the problems that I know about. I consider these to all be minor operational issues, since they have either do not interfere with the main FxGen functionality or there are simple work-arounds. However, fixing these problems is not so easy (which is why I haven't yet).

1. Timing slightly off

The timing of FxGen is supposed to be dead-on, as explained in the Timing Pipeline section. However, if you run a real and virtual prop mapped to the same channels, you may see that the timing is off slightly, as shown in the FxMixedMode demo (the grids should look the same):



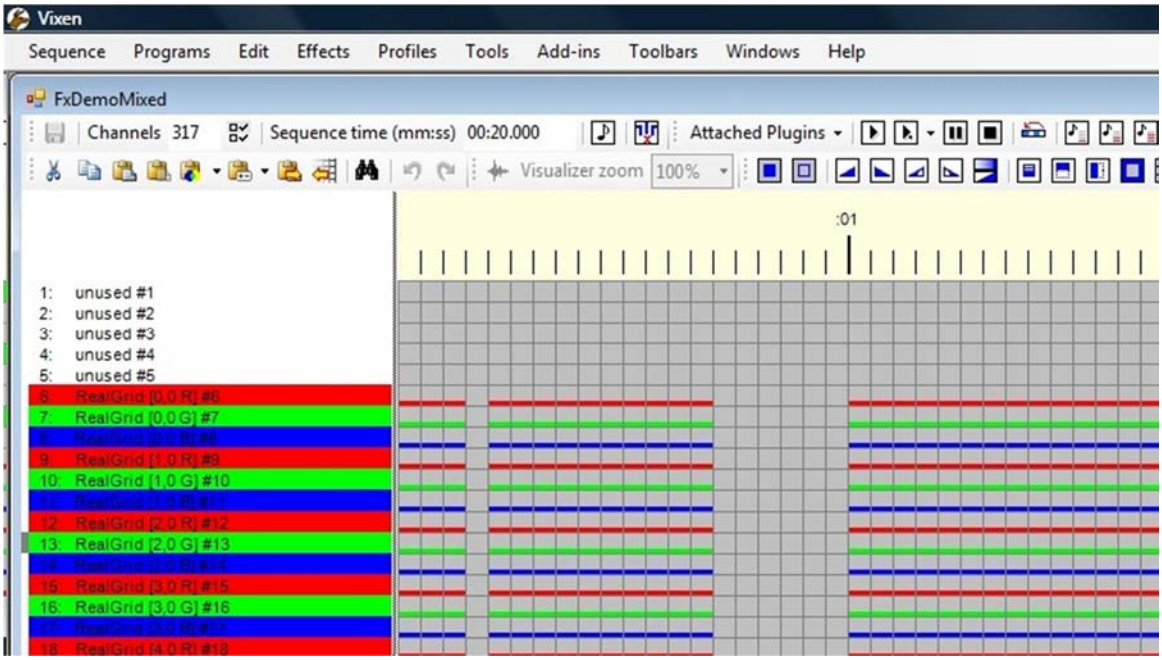
For animated text and bitmap effects this probably does not matter, but for effects that must be synced exactly, the work-around would be to just manually shift the cells left or right the appropriate number of columns using the regular Vixen editing tools.

2. Skipped frames

Vixen 2.x (or maybe .NET) will skip frames in order to try to maintain overall timing integrity. This is noticeable with Adjustable Preview and Grid Editor even without FxGen – the playback sometimes gets

"choppy" or "jumpy" because frames have been dropped. Since the problem seems to be most sensitive to graphics subsystem performance, the easiest work-around is simply to turn off Output Plug-ins such as Adjustable Preview or Grid Editor during live show playback. However, that makes working with FxGen more difficult because you can't see the results.

Dropped frames during FxGen recording are a problem. They will look like blank columns in the Vixen cell editor:

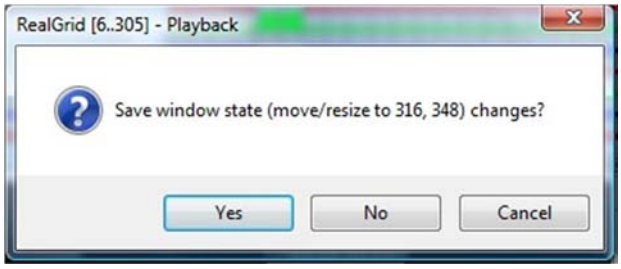


Dropped frames during FxGen recording

The work-around for this would be to re-try the recording, or to manually copy and paste good columns over the blank ones and apply "touch-ups" as needed. BTW, I've noticed that several frames are more likely to be dropped shortly after the Sequence starts than later on in the playback, so another work-around would be to place the effect later in the Sequence, record it, then copy/paste or shift the recorded cells to the left so they are in the desired position.

3. Saved Screen State

The Grid Editor tries to save/restore its window size and position for playback, but it gets a little confused about whether it is in a real Sequence or a virtual Profile, so it will sometimes prompt to save info after playback even though it did not change:



Grid Editor prompts to save display state changes at end of playback

I've actually tried to fix this a few times, but didn't quite get it right yet so for now the work-around is just to say No. Normally you would disable any Grid Editor plug-ins for live show playback anyway.

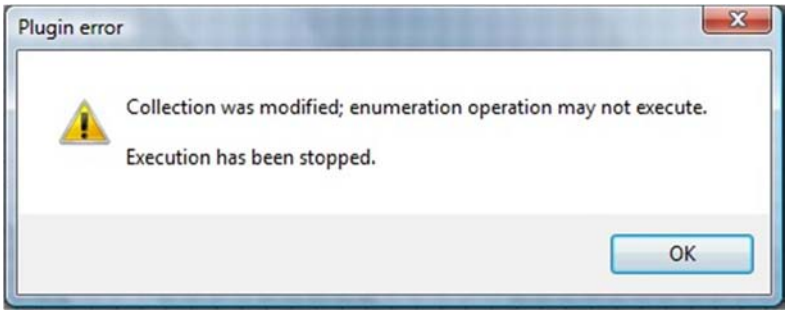
Since it's nice to be able to set the playback position of the Grid Editor for editing purposes, the size and position can be controlled by manually adding <Size> and <Location> tags to the <PlugIn> node as follows:

```
<PlugIn name="Grid Editor" key=??? id=??? enabled="True" from=??? to=???>
  <Display>
    <Location>160,150</Location>
    <Size>559,549</Size>
    <Scroll>50,129,575,575</Scroll>
    <ReadOnly>True</ReadOnly>
  </Display>
</PlugIn>
```

The <ReadOnly> tag will tell Grid Editor to not try to save any info (it can't in a virtual Profile anyway).

4. Collection Modified Error

At the end of playback, sometimes there will be some popup errors about "collection modified". These appear to be benign errors, but are annoying because there can be several of them:



Collection Modified error at end of playback

This error doesn't get caught inside FxGen even with high-level Try/Catch statements, so it may be coming from within a Vixen function or .NET itself. I've tried various ways to trap or avoid the error, but it still happens sometimes (not always); it's probably timing-related.

It's easy enough to dismiss these popup errors at the end of playback when editing Sequences, but for live show playback it would be a problem if they prevent the next Sequence from running. The work-around in that case would be to record FxGen effects to the Sequence and then disable FxGen before playback. If it's related to timing, then simply disabling Adjustable Preview and Grid Editor may also alleviate the problem.

5. Fx Cell Values

The encoding of Fx control values within cells in the Sequence is a little awkward, but manageable since it takes relatively few of them to run many effects. As noted earlier, small cell values are difficult to see, and the work-around is to use larger values in the cells. For example, I started the function codes at 200, and text string values at 100 so these would be more visible in the cell grid:

6. More Information

If you have any questions or comments for improvement of this article, please send an email to techguy@eShepherdsOfLight.com.

While working on FxGen, I was reminded of how God designed us. The framework + customization aspect of FxGen was representative of how He freely gives us a "framework" of raw materials such as resources, abilities and opportunities, and then allows us to "customize" those to create our own "effects". Hopefully the effects we create with our lives bring glory and honor to Him. In a smaller way, I hope that FxGen can help me or others create Christmas light displays that will honor Jesus Christ, who is the real Reason for the Season.

Also, the way we customize and use our effects is unique to each person. God gave us each a unique personality, so even those who have been given similar "frameworks" will create different life effects. That is also similar to FxGen, in that multiple people might start out with a similar set of resources and yet create vastly different Sequences and effects.

Other analogies could also be made, but I'll stop there.

License

Disclaimer

This project is a work in progress, not a finished product, so any of its components are highly experimental in nature. I got it to work okay for what I was doing, but it has not been tested thoroughly under all possible conditions, and some features have not been tested at all. This should be taken into consideration if you will be trying out any of this stuff. However, please feel free to ask questions or for clarifications on the info in this article; I tried to cover too much info, and as a result probably made some mistakes along the way.

Revision History

Revision	Date	Description
0.5	01/28/12	Started writing up initial version
0.7	03/2/12	Decided to do a little clean-up work on the code
1.0	10/4/12	Finished initial write-up; fixed a few bugs; initial version released

-eof-