Digital Video: Special Effects Filters

Now that we have taken a look at how to apply color adjustments and color correction for your digital video editing projects in VideoStudio Ultimate, we'll take a look at how to use **algorithms** to create digital video special effects. These algorithms are commonly available as "plug-ins" that add special effects to the software. Most of the software genres, such as digital imaging, digital audio editing, digital illustration, digital painting, sound design and visual effects, support plug-in architectures, just like your nonlinear digital video editing software does.

The reason software developers initially added a plug-in architecture is because they did not have the bandwidth (team) to be able to add all of the features that they wanted to their software packages. Plug-ins allowed "third party" developers to add special effects "filters" to these software genres, so that users could do special effects and other cool things with them.

In this chapter we'll take a look at a few of the 114 FX special effects that are included in Corel VideoStudio Ultimate X9. These plug-ins used to be third party plug-ins long ago, and are now provided as a standard set of "FX" in Corel VideoStudio Ultimate X9. Once all of the cool special effects (or FX) plug-ins had been developed by third party developers, they ended up becoming standard features for the various software development genres. Over time, these plug-ins were eventually acquired from third party software developers and added as standard features.

Pixel Processing: Pixel Based Algorithms

In the previous chapter we looked at algorithms that processed color values for each pixel in a frame over the duration of the video clip to provide color correction or color special effects processing. In this chapter we will look at algorithms that do processing on pixel locations to create a different type of FX.

Mirroring Pixels: Using the Reflection Algorithm

The first algorithm that I want to look at is used to create a standard **mirroring** effect. This can be used with logos; text; clip art; effects such as the animated film sprocket seen in Figure 11-1; and even video footage, if there is a good reason to mirror it, which will, of course, turn it upside down!

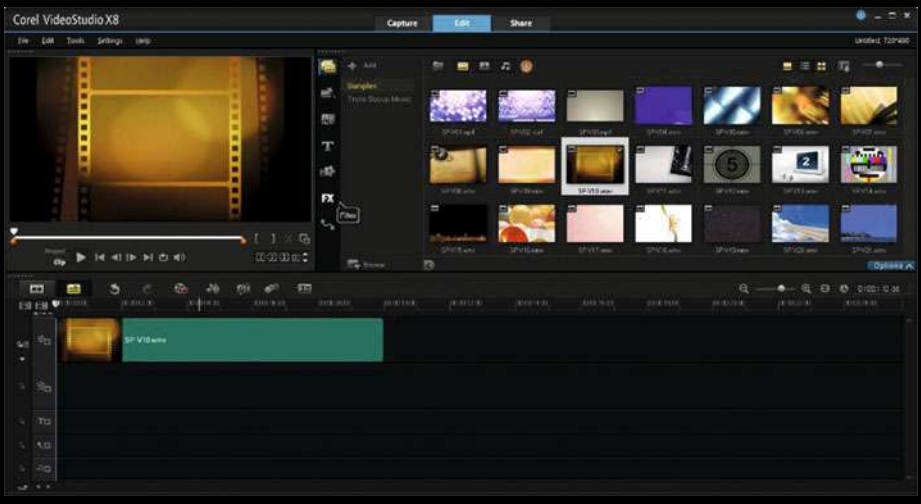

Figure 11-1. Create a new project, and drag out the SP-V10 clip

Let's create a new project called **DVE_Fundamentals_CH11**, and drag your **SP-V10.wmv** Media Bin Asset onto your time line, as is shown in Figure 11-1. Preview the animated film and lighting effect using the **Play** button in the preview area's transport, or you can drag the position marker to preview the content as well if you wish to do it that way.

In the Asset Bin click on the **FX** icon to open the **effect filter algorithm** assets that come with VideoStudio Ultimate. I counted these and there are **114** of them, so you cannot see them all at once. Use the scrollbar at the left of the asset preview pane to take a look through them as there are a ton of powerful algorithmic tools at your disposal in VideoStudio Ultimate X9.

Find the Reflection algorithm, and select it and drag it on top of the SP-V10.wmv video clip on the time line, as is seen in Figure 11-2. Your clip coloring will invert from teal to pink when your mouse is over it; at that point, you can release your mouse button, and your FX algorithm will then attach itself to the target video clip.

Figure 11-2. *Click the FX bin, and drag out a Reflection Filter*

A small FX will appear in the upper-right hand corner of the video clip element in the time line view to designate an FX.

Right-click on the clip once you have added an FX, as is shown on the left in Figure 11-3. Select the **Open Options Panel** menu item, which will replace part of the Asset Bin pane with a **Video** Tab on the left, and an **Attribute** Tab on the right.

Figure 11-3. Right-click on clip, and select Open Options Panel

Click on your Attribute Tab on the right, which is shown circled in red in Figure 11-4. Mouse-over the **Customize Filter**, under the algorithm selector UI pane, and then click on it.

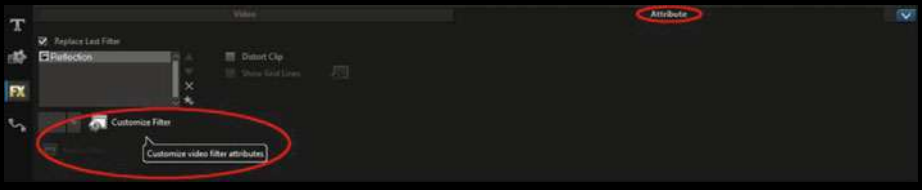

Figure 11-4. Select the Attribute panel, then Customize Filter

This will open the **NewBlue Reflection** dialog, as seen in Figure 11-5, where you will be able to customize precisely just how the Reflection (mirroring) functions in 2D space. There is a **Shape** X,Y Position setting; an **Image** X,Y Scale setting; and a **Background** Opacity, Color, and Feathering setting area. You will also be able to control the **Reflection** itself, and add **Ripples**.

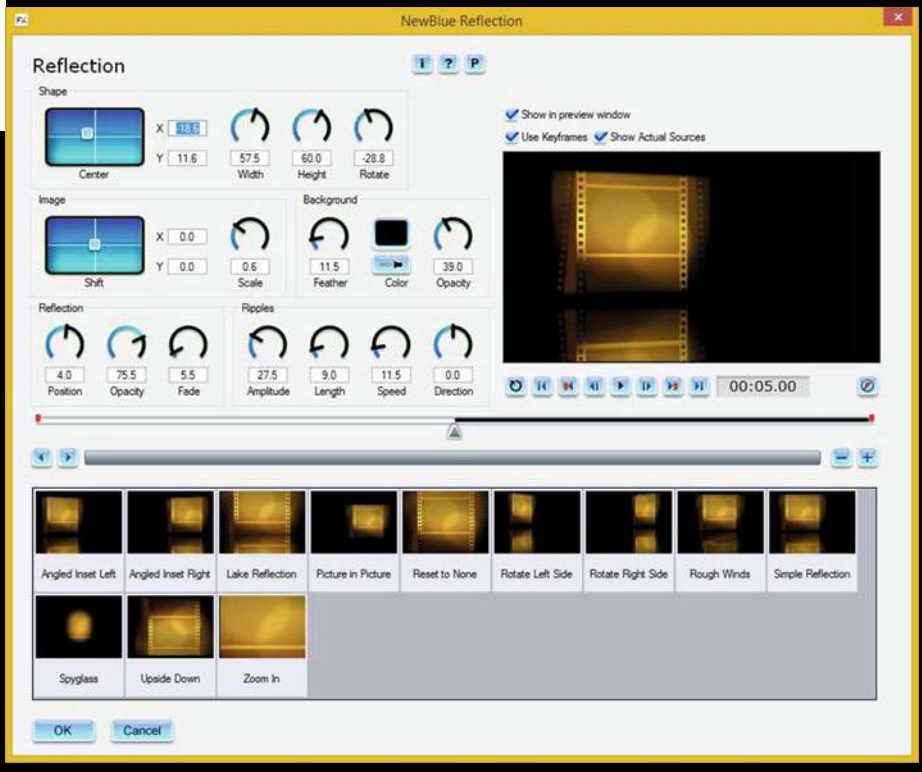

Figure 11-5. Use NewBlue Reflection dialog and customize effect

Next, let's take a look at the Boris FX Graffiti Engine!

Boris FX: The Boris Graffiti Title Algorithm Engine

Although the Reflection algorithm has over a dozen settings, the **Boris Graffiti 6.1** Titling Effect Engine (algorithm) is far more complex as it's actually a software package within another software package! Click the **Media** icon in the Asset Bin and drag the **SP-V16.wmv** video clip onto your project's time line, **as is shown** in Figure 11-6. This is a clip of some balloons rising up into the air, and it will be a great background for titling FX.

Figure 11-6. Add the SP-V16 clip to the project via drag & drop

Find the **Boris Graffiti 6.1** algorithm, and select it and drag it on top of the **SP-V16.wmv** video clip in the time line, so your clip turns pink, as is shown in Figure 11-7, then drop it.

Figure 11-7. Use Skin Tones and Black preset for more contrast

Next, use the same work process shown in Figure 11-3 and 11-4, and open the **Boris Graffiti 6.1** dialog, as seen in Figure 11-8. I selected the **Shatter Glow** Preset and clicked the green **Apply** button seen in the lower-right hand corner of the dialog.

Figure 11-8. Select the Shatter Glow Preset, and click on Apply

To access the complete Boris Graffiti software, click on the **Advanced Mode** button at the bottom left of the dialog, seen in Figure 11-8 circled in red. This will open up Boris Graffiti 6.1 (the entire software package), which can be seen in Figure 11-9. As you can see, it's a complete titling software package.

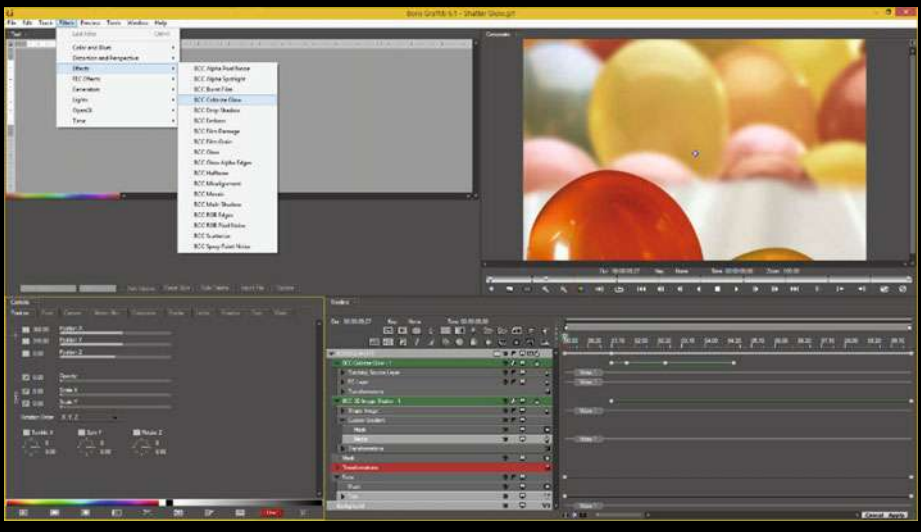

Figure 11-9. Advanced Mode button opens a Boris Graffiti Engine

Fluid Dynamics: Using the Rain Filter Algorithm FX

Let's add another video clip to the project and drag the **SP-V12.wmv** video onto your time line, as shown in Figure 11-10. Preview the animated countdown effect, using the **Play** button in the preview area's transport. This will be the perfect clip for adding a rain effect over the "5-4-3-2-1" countdown video clip.

Figure 11-10. Drag & Drop an SP-V12 clip onto the Time Line view

In the Asset Bin click on the **FX** icon to open the **effect filter algorithm** assets. Find the **Rain** algorithm, and select it and drag it on top of the **SP-V12.wmv** video clip on the time line as is seen highlighted in pink in Figure 11-11. This will add a fluid dynamics rain effect on top of your countdown video clip.

Figure 11-11. Drag & Drop the Rain FX on your SP-V12 video clip

Drag your preview marker at the top of the time line over the clip now that it has the FX attached to it, and preview the Rain. The default setting should provide a nice rainfall effect that fits in perfectly with the black and white countdown video as you can see in the upper-left hand corner of Figure 11-12.

Figure 11-12. *Preview the Rain filter with the Scrubber Preview*

If you want to customize the Rainfall, use the same work process shown in Figures 11-3 and 11-4 and open the **Rain** dialog, as seen in Figure 11-13. I used the default 392 **Density** setting along with 17 percent **Length**, 9 percent **Width**, 15 percent **Background Blur**, and a 50 percent Variance. The default uses 54 percent for **Body**, and 64 percent for **Opacity**.

Figure 11-13. *Fine-Tune the Rain Algorithm with the Rain dialog*

Mercalli Video Stabilization: Algorithmic Steadying

Let's add a fourth fireworks video clip to the project and drag the **SP-V15.wmv** video onto the time line as seen in Figure 11-14. Preview the animated fireworks effect, using the **Play** button in the preview area's transport. This will be the perfect clip for testing this motion stabilization over the exploding fireworks.

Figure 11-14. Drag & Drop the SP-V15 clip on your Time Line view

Since only three of the four video clips are now showing in the time line view editing area, let's use the **Fit Project in Time Line window** icon, as seen circled in red in Figure 11-15. I am doing this so the rest of your figures for this chapter will show an entire project so you see everything in proper context.

Figure 11-15. Click on the Fit Project in Time Line window icon

In the Asset Bin click on the **FX** icon to open the **effect filter algorithm** assets. Find your **Mercalli 2.0** software, then select it, and drag it on top of your **SP-V15.wmv** video clip, on the time line view, as seen highlighted in pink in Figure 11-16.

Figure 11-16. Drag & Drop Mercalli 2.0 on the SP-V15 video clip

Next, use the same work process shown in Figures 11-3 and 11-4, and open the **proDAD Mercalli 2.0.12** dialog as is shown in Figure 11-17. Select all the stabilization options you want the algorithm to apply, and then click on the **OK** button to process.

Figure 11-17. A Mercalli algorithm offers stabilization options

As you can see in Figure 11-17, this Mercalli 2 software package (another "software within a software package" scenario) offers the ability to compensate (or stabilize) camera **rotation** or "roll," camera **zoom**, or (slight) field of view (FOV) changes as well as **tilt** in a camera, on either the horizontal or on the vertical axis. There are also border quality settings as well.

You can also **smooth out panning motions** using this smart pixel processing algorithm, and even select a **Virtual Stabi-Cam** option that tells the algorithm to simulate the footage as shot using a Stabi-Cam rig or outfit, if you do not own one yet.

I tried to cover some of the more mainstream types of 2D digital video effects algorithm genres in this chapter, such as spacial (mirroring), fluid dynamics (rain), titling (graffiti), and motion stabilization (Mercalli), but, as you might imagine, it would take an entire book (or three) to cover all of the 114 FX special effects in VideoStudio Ultimate, which is why it's such a great buy!

The best way to learn, and get experience using these 2D algorithms, is to try using them on some of the sample clips as we have been doing during this chapter. The more you use these FX algorithms in your day-to-day digital video editing work the more comfortable and familiar you will be with what they can do!

Summary

In this eleventh chapter, we took a look at how to apply filter or effects algorithms to your digital video clips. These will process all of the pixels, in all of the frames in your digital video clips assets, using your plug-in special effects related filters, whereas the color effect (FX) filters you looked at in the previous chapter only processed the color channels for each pixel. We first looked at the **Reflection** filter, which mirrors pixels around an X and Y axis that you can define, along with other advanced parameters controlling how the reflection looks.

Next you looked at the **Boris FX Graffiti 6.1** software in the form of a plug-in filter for Corel VideoStudio Ultimate X9. You applied one of the cooler titling effects adding a glow and exploding some text, and saw how to access your full version of the Boris FX Graffiti software hidden inside of VideoStudio X9.

Next, you looked at the **Rain** filter, an example of **fluid dynamics** algorithmic processing; and the advanced parameters in the Rain dialog, including a Density setting along with Length, Width, Background Blur, Variance, Body, and Opacity setting.

Finally, you took a look at advanced **Motion Compensation and Correction** algorithmic processing, using a software package called **Mercalli 2.0.12** from proDAD. This provides advanced post processing to digital video footage to correct things that may result from shooting without using a professional tripod setup.

In chapter 12, you'll start to learn about more advanced digital video data footprint optimization features, by learning about digital video compression algorithms and codecs.

# of Digital Video: Compression

Now that we have taken a look at how to apply color adjustment, color correction, motion correction, and special effects for your digital video editing projects in VideoStudio Ultimate, we'll take a look at how to use **compression algorithms** to create optimized digital video assets. These algorithms are also commonly available as plug-ins, which add different data formats to the digital video editing software. Most of the software genres, such as digital imaging, digital audio editing, digital illustration, digital painting, sound design, and visual effects, support these codec architectures for "exporting" new media data, just like your nonlinear digital video editing software does.

The reason that compression codec (Code-DECode) software uses this plug-in architecture is because software packages are required to pay licensing (or royalty) on the use of each codec patent and technology. If a software package wants to support a codec, they simply license it and plug the codec algorithm into their software with an "under the hood" plug-in architecture. A good example of this is the Editshare Lightworks user base, who have been requesting the addition of the open source WebM codec to be added to Lightworks. All Editshare would have to do is to plug it in, since there is no license fee or royalty involved.

W. Jackson, *Digital Video Editing Fundamentals*,
DOI 10.1007/978-1-4842-1866-2_12

In this chapter we'll take a look at the concepts behind data footprint optimization, or, more simply put, achieving the smallest possible digital video asset file size.

Then we will take a "hands-on" approach to changing some of the settings for one of the video codecs, creating a custom preset, to be used to create an optimized digital video asset.

This will show you how to optimize a video codec setting in VideoStudio Ultimate X9. However, I will recommend something dedicated to video compression, such as Sorenson Squeeze Pro 10 for professional usage. I cover Sorenson Squeeze Pro Desktop 10 in my *Pro Android* book series from Apress (2014 through 2017).

Data Footprint Optimization: Pure Theory

In the first section of this chapter, we will take a look at some of the main concepts and theory behind digital video data footprint optimization. These tie back into those foundational concepts you learned about in chapters 2 and 3 regarding pixels, resolution, aspect ratio, color theory, alpha channels, frames, bit-rates, and similar technical information regarding how video actually works. We will discuss how these attributes of digital video assets can affect the data footprint optimization process in this first section of the chapter, starting out with the most important factors (resolution, frame rate, and bit-rate).

Pixel Scaling: The Bane of Image and Video Quality

The objective in delivering digital video on today's consumer electronic devices, which we will be covering in chapters 13 and 14, is to get the smoothest playback. One of the factors in the achievement of this objective is to prevent pixel scaling. This scaling algorithm, which is often a **Bilinear** scaling algorithm, has to scale every frame of video, which can quickly eat up all of your device's processing power. At 30 frames per second, you are asking your CPU to scale a ton of pixels and to perform that scaling algorithm 30 times each second! Essentially, scaling video is like saying: "CPU, I require 100 percent of your processing power, don't do anything else!" This will grind and device to a halt and greatly affect your user experience (UX).

To avoid using scaling, you'll want to make video assets that match all of the standard device screen resolutions. These include the following: **Wide SD** (800 by 480), **Pseudo HD** (1280 by 720), **True HD** (1920 by 1080), and **Ultra HD** (4096 by 2160). Let's get into the popular wide-screen (16:9) and non-wide (4:3) resolutions in the next section of the chapter and then take a look at how digital video can be either captive (on disk), or stream from a server.

Digital Video Resolution: Popular Video Standards

Let's start out covering primary resolutions used in commercial video. Before HDTV (High Definition Television) came along, video was called "SD," or Standard Definition, and used a standard vertical resolution of 480 pixels. The original aspect ratio for SD was 4:3 (640 by 480). More recently, a wide-screen aspect ratio was added, making SD video 800x480 resolution. HD resolution video comes in two resolutions; 1280 by 720, which I call "Pseudo HD," and 1920 by 1080, which the video industry calls "True HD." Both use the 16:9 wide-screen aspect ratio and are now used not only in Film, HD Television, and for iTV Sets, but also in Smartphones, eBook Readers, Phablets, and Tablets.

There's also a **1920x1200** resolution that is a less wide, or taller, **16:10** pixel aspect ratio. It is becoming more common as a wide-screen device aspect ratio, as are your **16:8**, or **2:1**, aspect ratio, with **2160x1080** screens on the market, since 2013.

There is also a 16:10 Pseudo HD resolution, which features **1280 by 800** pixels. In fact, this is a common Laptop, NoteBook, NetBook, and Mid-Size Tablet resolution. I wouldn't be surprised to see a 16:8 **1280 by 640** screen offered at some point in time.

Generally, content developers will try to match the video content resolution to the device display "pixel for pixel." The reason for this is to avoid scaling. If the resolution does not match the screen resolution, fill your backplate with black and then center the video on the screen with the extra pixels along the edges. Black pixels turn that portion of the display "off."

Similarly, manufacturers will try to match display screen resolution to popular content resolutions. Blu-ray is 1280x720, and so there are a lot of 1280x720 screens and 2560x1440 screen sizes. Two times 1280x720 on each axis scales up perfectly with a 4 pixel (2x2) matrix, for each pixel in the Blu-ray content. Even 2X or 4X scaling (up or down) is faster and gives superior visual results. Scaling down (termed: downsampling) 2X or 4X is preferable to scaling up (termed: upsampling).

Digital Video Playback: Captive versus Streaming

Regardless of the resolution you choose for your digital video content, video can be accessed by your applications in a couple of different ways. The way I develop applications, because I am a data optimization fanatic, is **captive** within the application. This means the data is inside of the application's distribution file itself. In the case of Android applications, the data file would be stored inside of a **/res/raw/** raw data resource folder. You'll see this in the next chapter on programming, when I show you how to code a digital video playback engine for Android OS.

The other way to access video inside your application is by using a **remote video data server**. In this case, the video is **streamed** from the remote server, over the Internet, and over to your user's hardware device as the video is played back in real time. Let's hope your video server does not crash, which is one of the downsides of streaming video, relative to captive video.

Video streaming will inherently be more complicated than playing back captive digital video. This is because a device is communicating in real time with a remote data server, receiving video data packets, decoding the data packets as a video plays, and then writing the frames to the hardware display. This video streaming capability is supported using WebM using WebM format, or by using MPEG-4 AVC H.264 or MPEG-H EVC H.265 data formats.

Digital Video Compression: Bit-Rates and Playback

Another important digital video concept that's important to data footprint optimization is the concept of bit-rates. The bit-rate is the critical setting used in your video compression process, as the bit-rate represents the target bandwidth or the data pipe size. This defines the network capability that's able to accommodate a certain number of data-bits of video streaming through it every second.

Bit-rate settings should take into consideration the CPU processing power within any given hardware device, making video data optimization even more important to your DVE application's playback quality, as some devices still have only one CPU core.

This is because once the bits travel through a data pipe they also need to be processed and then displayed on the device screen. In fact, captive video assets, included in Android .APK files, only need optimizations for processing power. The reason for this is because if you are using captive video files, there is no data pipe for the video asset to travel through, and thus no data transfer overhead. So bit-rate for digital video assets needs to be optimized not only for **data bandwidth** but also with an anticipation of variances in **CPU processing capability**.

In general, the smaller the video data file size you are able to achieve, the faster the data will travel through a data pipe, the easier it will be to decode that data using the codec and the CPU, and the smaller the application file size will be.

Single-core CPUs in devices such as smartwatches may not be able to decode high-resolution, high-bit-rate, digital video assets, without "dropping" frames. This is a "playback quality" issue, so make sure to thoroughly optimize lower bit-rate video assets, if you are going to target older (or cheaper) devices!

Digital Video Codecs: An Encoder and a Decoder

Digital Video Codecs have two sides: their encoding side and their decoding side. The decoder side of your digital video codec will always be optimized for speed, because smoothness of playback is your key issue, and the encoder side will always be optimized to reduce the data footprint for digital video assets that it will be generating. For this reason an encoding process will take a long time, depending on how many processing cores a workstation contains.

All digital video content production workstations should support at least 8, 12, or 16 processor cores. The logic behind this statement is that the encoding should process faster, and your special effects algorithms should be rendering more quickly, the higher the number of processing cores installed.

More than one software manufacturer makes MPEG-4 encoder software, so there can be different MPEG codec encoder software that would yield different (better or worse) results, as far as encoding speed and file size goes.

One professional encoding solution that is available, if you wish to encode digital video assets is **Sorenson Squeeze Pro Desktop** from Sorenson Media, which is currently at version 10.

Since we are using the more affordable Corel VideoStudio Ultimate in this book so that all readers can follow along with the Trial Version, or purchase the full version for under $100, I will show you how to create custom compression settings using this software later on during the chapter.

There is also the professional open source DVE solution, called Editshare Lightworks 12.6, but the free version does not currently support output using an MPEG-4 H.264 AVC and WebM VP8 or VP9 codecs. Since the WebM VP8 or VP9 codecs are open source, Editshare could add WebM support, if they wanted to.

So for now, I'll have to use VideoStudio Ultimate, since it has the proper codec support for Android Studio 2 and HTML5.

Most developers should use MPEG-4 AVC H.264 and WebM VP9 for the best visual results, codec data footprint optimization performance, and the most widespread playback support across the two largest platforms (HTML5 Browsers, HTML5 OSes and Android).

Digital Video Optimization: Key Encoder Settings

When optimizing for digital video asset file size, using encoder settings, there are a number of important settings that directly affect your data footprint. This is true irrespective of what brand of DVE software package you are using, since the codec features dictate the encoder dialog, not the DVE software package features. I will cover the key settings in the order in which they affect video file size, from the most impact to the least impact, so you know which parameters to "tweak" or adjust in order to obtain the results that you're looking for.

Just like digital imagery compression, video resolution, that is, the number of pixels within each frame of video, would be the optimal place to start the data optimization process. If you are targeting 1280x720 smartphones (or tablets), you do not need to use a 1920x1080 resolution in order to get great visual results from your digital video asset.

In fact you don't want to scale unless it is by a factor of 2, so you could use **1280x720**, for **1280** and **2560** smartphones, or tablets, **1920x1080** resolution for **1920** and **3840** smartphones, or tablets. You could also create your **640x360** version for less expensive smartphones, like Firefox sells in emerging countries and finally, an **800x480** version, for entry-level smartphones.

The next level of optimization would come in your **number of frames** used for each second of video (**FPS**). This assumes the actual seconds contained in the video itself can't be shortened through editing. This is known as the **frame rate**; so instead of setting a video standard, **30 FPS** frame rate, consider using the film standard frame rate of **24 FPS**, or, the multimedia standard frame rate of **20 FPS**. You may be able to use a low 15 FPS frame rate, depending upon the content; you will just have to try it!

Note that 15 FPS is half as much source data as 30 FPS, a 100 percent reduction of data going into the encoder. For some video content this will playback the same as 30 FPS content. The only reliable way to test how low you can get the FPS before it will start to affect your video playback quality, is to **set**, **encode**, and **review** the result with different frame rate settings during your digital video (encoder) content optimization work process.

Your next optimization setting to "tweak," or experiment with settings for in obtaining a smaller data footprint will be the **bit-rate** that you set for the codec to try to achieve. Bit-rate settings equate to the amount of compression applied. This sets the visual quality for video data. It is important to note that you could simply use 30 FPS 1920x1080 video and specify a low bit-rate "ceiling." If you do this, the results will not be as high quality visually, as if you first experimented with low frame rates and lower resolutions and used the higher (quality) bit-rate settings.

Your next most effective setting for obtaining optimized digital video assets is the number of **keyframes**. The codec uses your keyframe setting to know when to **sample** the digital video. Video codecs apply compression by looking at a frame, and later encoding only the changes, called the "offsets," over your next few frames of video. This is so that it does not have to encode every single frame within your video data stream. This is why a talking head video will encode smaller than a video where every pixel moves on every frame, such as video with fast panning, or rapid zooming, for instance. Only the pixels that change in a talking video – this should be the mouth and blinks of the eye, are encoded as offsets, whereas with panning and zooming, every pixel on each frame will be moving from one location to another and therefore there are no fixed pixels that can be optimized.

The keyframe setting in the encoder will force the codec to take a fresh frame sample from the video data asset every so often. There is usually an **auto-keyframe** setting for keyframes; this gives the codec algorithm itself control over deciding how many keyframes to sample, and when to sample keyframes.

Manual control of keyframes can be achieved by not using an auto-keyframe setting, and setting a number of keyframes per second or a number of keyframes over the entire video duration.

The next most optimal setting for obtaining the smallest data footprint is the **quality** or **sharpness** codec setting, which is usually implemented in a codec dialog by using a **slider bar**.

Sharpness will control the amount of **blur** that the codec will apply to your video pixels before compression. In case you are wondering how the trick works so that you can also apply it yourself in GIMP or DVE, during your digital image optimization work process, applying a **very slight blur** to an image or video, which is usually not desirable, allows for better compression. The reason for this is that sharp transitions in an image, such as sharp edges between colors, are more difficult for the codec to encode optimally, that is, using less data footprint.

More precisely, no pun intended, sharp abrupt transition in color will take more data to reproduce than soft transitions will. I would recommend keeping the quality or sharpness slider between an 80 percent and 96 percent quality setting, and try to get the data footprint reduction with other variables that we have discussed here such as resolution, bit-rate, or frame rate (that is, FPS).

Ultimately there will be a significant number of different variables that you'll need to fine-tune in order to achieve the best data footprint optimization for each particular video data asset. Each will be different (mathematically) to the codec, as each video asset may be a different array (collection) of pixel color data. For this reason, there is no "standard" collection of settings that can be developed to achieve any given result.

Your experience **tweaking** various settings may eventually allow you to get a better feel, over time, as to those settings you'll need to change, as far as the parameters go, to get your desired compression result with different types of uncompressed video source assets. There's no silver bullet to data footprint optimization; you simply have to spend the time determining the optimal settings to use for each digital video asset you create in a nonlinear editor (NLE) digital video editor (DVE) software.

Digital Video Asset Optimization: Why It Matters

There are two primary reasons that you want to optimize digital video assets. On the **server side**, it is so that your server has less data to serve for each video. For a given server bandwidth capacity, the smaller each video asset is the more users can be served by a server, as **Video Size * Number of Users = Capacity**. On the **client side**, video asset optimization can make your apps **distributable file** size smaller if you are using captive video, and, if you have optimized your resolution to your screen size, also optimizes **system memory usage**, which in turn optimizes CPU usage, since less memory access uses less CPU overhead. This is especially important when using video files in smartwatch apps.

VideoStudio X9: Creating Codec Presets

The first thing that we want to do, before we start compressing data into file formats with codecs, is to make sure that all of your acceleration optimizations are enabled in your VideoStudio code. You will do this so that you get the maximum performance from your workstation, so that the compression process performs faster, as you'll be trying out many different codec settings. Launch VideoStudio Ultimate, and use the **Settings ➤ Preferences** menu sequence and click on the Performance tab, shown in Figure 12-1. Select 1280x720 for your baseline **Smart Proxy**, and select the acceleration options shown numbered as #3 and #4. Click the **Yes** option (#5) and then the **OK** button (#6) to set the options. These will invoke code in VideoStudio that will make your video encode and decode faster, as well as using any GPU acceleration adapter in your workstation. If a GPU isn't present, the option will be grayed-out and won't be an option (i.e., not selectable).

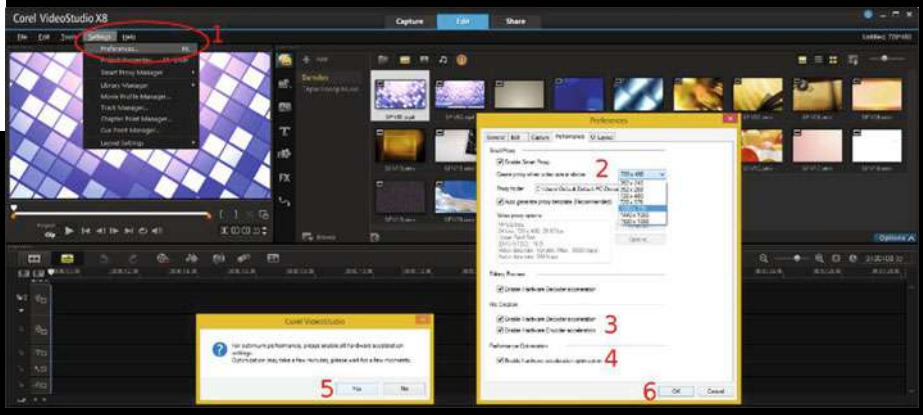

Figure 12-1. Use Settings ➤ Preferences and enable acceleration

Let's, take a look at your next menu item down, which is your **Project Properties** menu item, as is shown in Figure 12-2, labeled with a #1. This opens the dialog shown as #2, where you can select the **AVCHD** codec type from a **Project Format** drop-down selector. This will populate the codec preset display pane with different codec configurations, as seen in the dialog with a #3 and then you can select the **18Mbps** preset and click the **OK** (#4) to open the **Modifying Your Project Settings** (warning) dialog #5 where you can finally click on the **OK** button (labeled with #6).

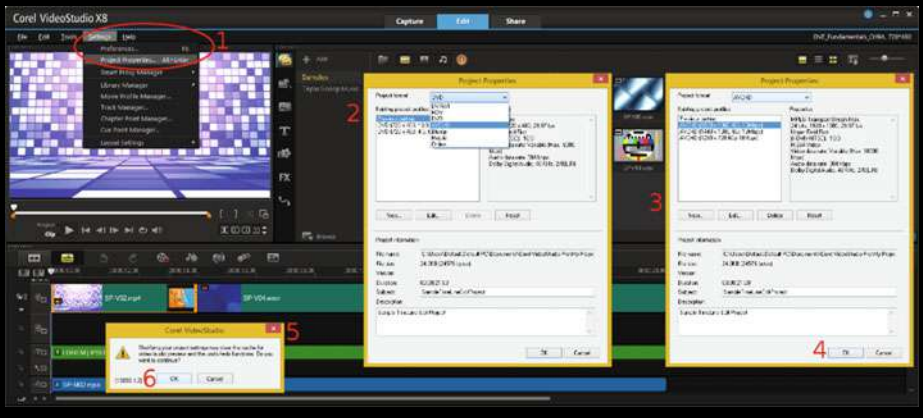

Figure 12-2. Use Settings ➤ Project Properties and enable AVCHD

What you're doing here is selecting one of those presets that comes with VideoStudio Ultimate X9, to use as a **baseline**, so that we can see what some of the different setting parameter variations will do to the data footprint (resulting file size).

Next click on the **Share** tab and the **AVC/H.264** option and you will see the setting that you selected, circled in red, and you can click the **Start** button to create a baseline video file, as is shown on the right in Figure 12-3.

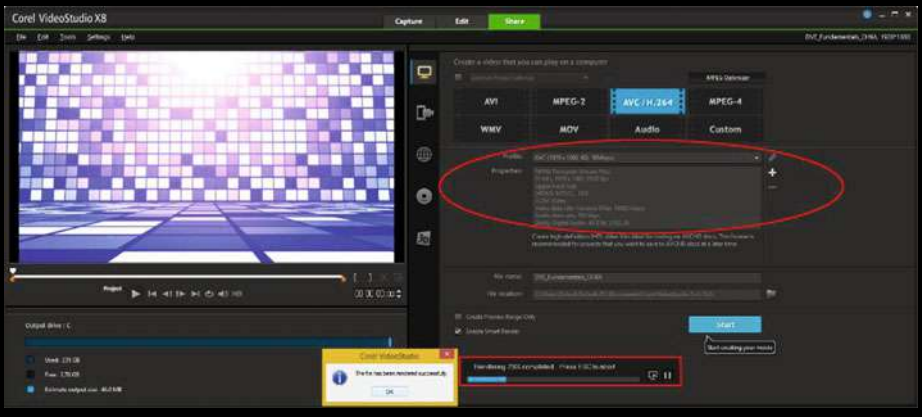

Figure 12-3. Use the Share tab to export the 18 M bit-rate AVCHD

Now that we have an 18Mbps baseline to compare our video optimization against, let's go back into the Project Properties dialog and click the **Edit** button and edit that preset to create one that gives us a smaller file size. This will open the Edit Profile Options dialog, with the **Corel VideoStudio** tab, where a preset name can be entered; **General** tab, where general settings such as Encoder or Frame Rate can be entered; and a **Compression** tab, where you set optimization settings, seen in Figure 12-4.

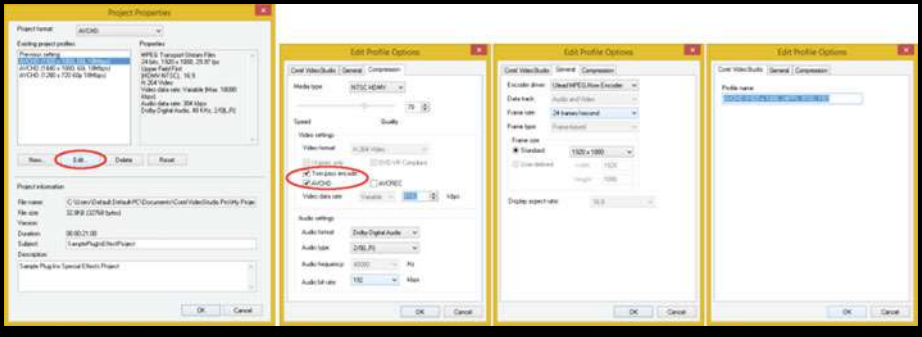

Figure 12-4. Edit 18 M preset; set 2-pass encode, 8192 k, 192, 24

As you can see, I selected the **Two-pass encode** option to give me better compression results (but longer encoding times), and set a 8192 kbps (8Mbps) bit-rate and a 192 kbps bit-rate on audio. I also set 24 FPS and named the codec accordingly.

To reset the settings back to the 18Mbps preset, you can use the Reset button, shown circled in red in Figure 12-5. This should give you a confirmation dialog, seen on the left. Notice that there is a Previous settings entry so you can still access your optimized settings.

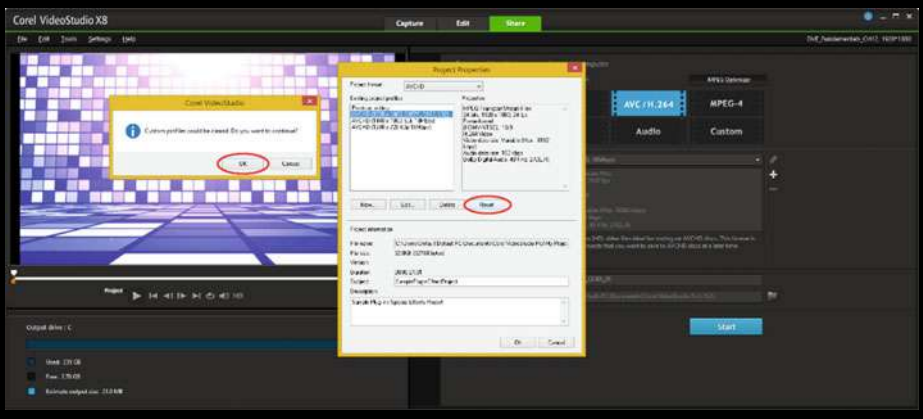

Figure 12-5. *Reset your altered preset, back to the 18 M version*

Next, let's use that **New** button, and create a new preset using your Previous settings preset data holder. This will give you the same series of dialogs, which should be filled out like they were previously, as was shown in Figure 12-4.

Click on the New button, **as shown** in Figure 12-6, and **OK** button, both shown circled in red, to create your new preset.

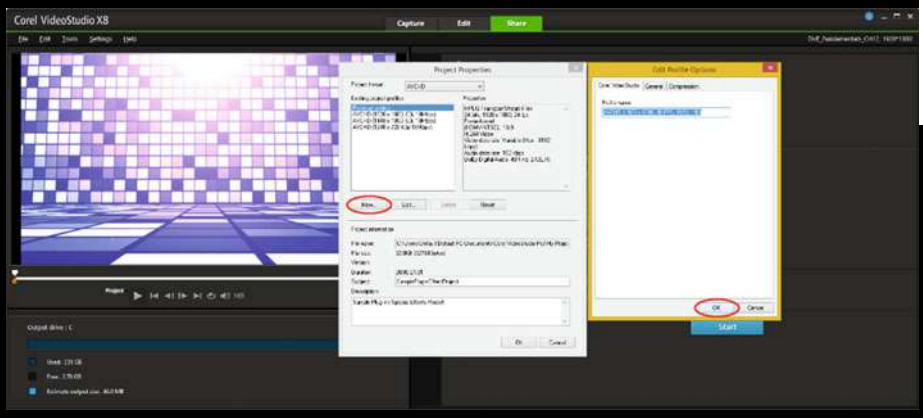

Figure 12-6. Select Previous Settings and click the New button

Find the **Same as Project Settings** option, located in the top of your **Share** tab, and select it, so your new settings will be shown in your **Properties** area, as shown in Figure 12-7, then use the **Start** button to create an optimized digital video file.

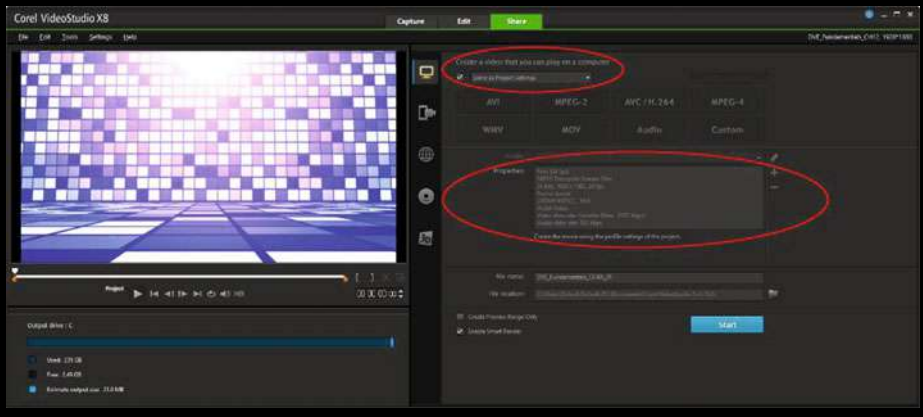

Figure 12-7. Select Same as Project Settings; Start Compression

Next open your file management utility, for Windows this is Windows Explorer, and go to your **Documents** folder, and **Video Studio Pro** subfolder, where your files are being output. For my installation this is Documents/CorelVideoStudioPro/18.0, as you can see in Figure 12-8. As you can see, I have reduced the file size by 55 percent from 45 megabytes to 20 megabytes, yet, when I play the files back, side by side, they're identical in quality. You just removed 25 MB of unnecessary data from your video asset!

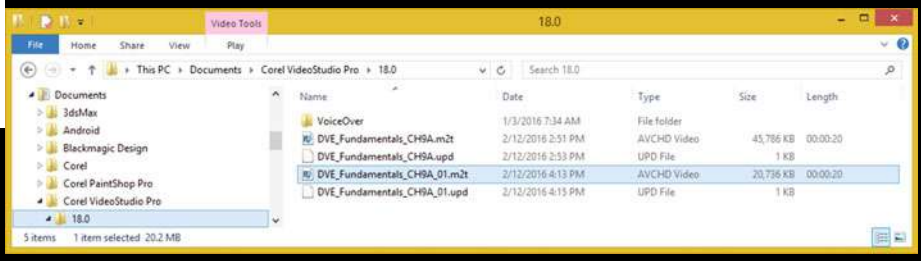

Figure 12-8. The 24 FPS 8192 setting reduces data footprint 55%

To create your Pseudo HD 1280 by 720 version, click your **Edit** button (see Figure 12-4) again, rename your preset, and go through the same work process shown in Figures 12-5 and 12-6. I like to do it this way, because I can start with a preexisting preset and simply tweak those settings I need to invoke further data footprint optimization. Since I'm using half the amount of pixels, by reducing the resolution, I set the bit-rate to 4096. All of the key settings can be seen in Figure 12-9, highlighted in blue, and when I preview the video asset it is still crystal clear and plays back smoothly, so my settings are working well.

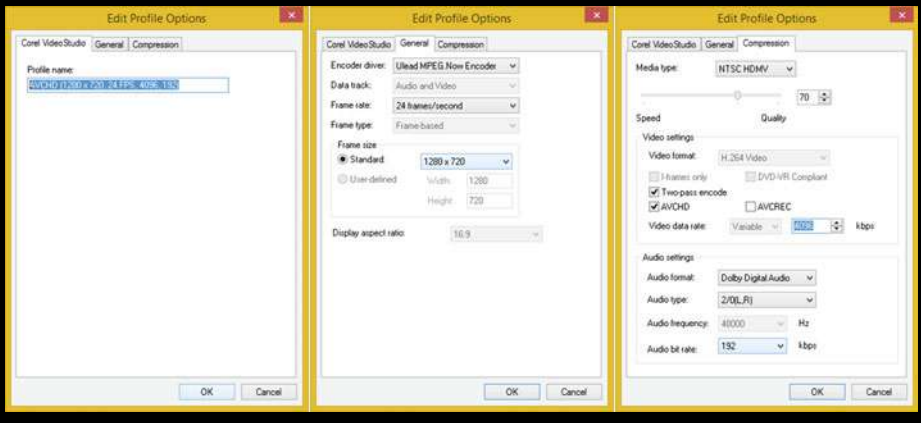

Figure 12-9. Create a 1280x720 24 FPS 4096 kbps 192 kbps preset

Notice that if I don't reduce the bit-rate to match this lower resolution, that the file size will be the same as the HD 1920x1080 version, because bit-rate specifies how much data can be used by the codec. This is shown in Figure 12-10, in version 02, so that you can understand this relationship. To create the preset, I used 1920x1080, 24 FPS, 8192 kbps video, and 192 kbps audio. To get down to 10 M I reduced the video bit-rate to 4096.

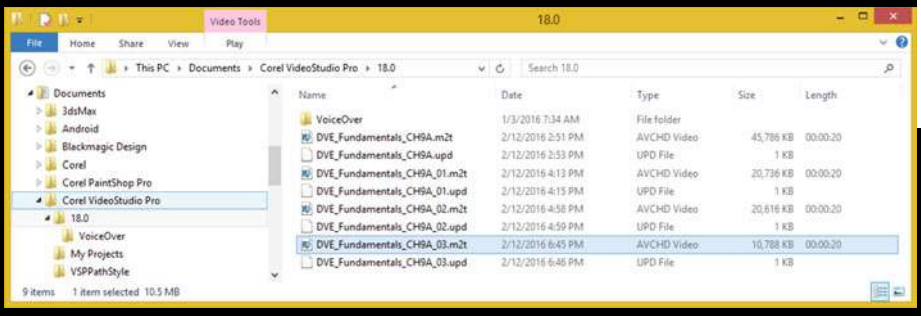

Figure 12-10. Drag & Drop an SP-V12 clip onto the Timeline view

It's important to note that you will continue optimizing the video asset using the above work process until your quality starts to decline. At this point you'll go back to the previous setting that gave you the best quality using the smallest data footprint. This process is not difficult once you master a work process, but it can be tedious as some assets will require more iteration through the work process than others will, as each DV asset contains a unique collection of frames, pixels, and colors specified, and sent to a codec, as purely numeric data values.

Summary

In this twelfth chapter, we took a look at the concepts behind data footprint optimization, as well as the work process for digital video data footprint optimization using VideoStudio Ultimate X9.

We looked at how the source video resolution, keyframes, aspect ratio, frame rates, bit-rate, and auto-keyframe settings can be set using the codec presets dialog, and how these should contribute to the digital video asset data footprint reduction.

Next, we took a look at how some of the key settings can be accessed in Corel VideoStudio Ultimate X9, and established a work process for creating resolution optimized versions of each digital video asset, reducing the data footprint close to 80 percent, from more than 45 megabytes to a mere 10 megabytes.

I like to optimize down to the single digit in megabytes (9 MB or smaller), in order to make sure my video assets stream, and playback, smoothly for all hardware devices and platforms.

In chapter 13, you'll learn about digital video programming concepts and terminology using popular, open source software development platforms, as well as how to use scripting in Corel VideoStudio Ultimate X9.