October 2004
Vol 2 No. 9

Vegas + DVD Tips, Tricks, and Scripts

Learning and using Vegas and DVD Architect

Bump Map – Simple Cartoon-look Effect

By Joseph Lawrence

A frequent thread on DV forums is "*how can I get a 'cartoon look' from my video?*" The popularity of movies such as Waking Life and American Pop has inspired some video producers to search for a filter, effect, or technique to simulate the style of these movies.

For the 'money is no object' mindset, Boris Red 3GL is now compatible with Vegas and has its cartooner effect. Unfortunately, Boris Red comes with a price tag of $1600.

If time is of no concern, then a manual method for rotoscoping your video has been used successfully by some editors. This method entails saving each frame of the raw video as a sequence of individual images, importing these images into a drawing program such as Paintshop or Photoshop, using filters to create the desired cartoon appearance on one image, batch processing the remaining image sequence with the same filter pack, and finally re-importing the image sequence into Vegas to reconstruct the video. Here I will describe a less expensive and more efficient method to achieve a cartoon-look effect for your raw video without leaving Vegas.

To see how this effect works, begin by placing your clip on the timeline. Once the event is in place, click on the "Event FX" button on the event or the "Track FX" button on the track header and add the Sony Bump Map effect.

The Bump Map plug-in for Vegas adjusts texture and lighting effects to give a video clip the appearance of physical depth. The effect can be used with two

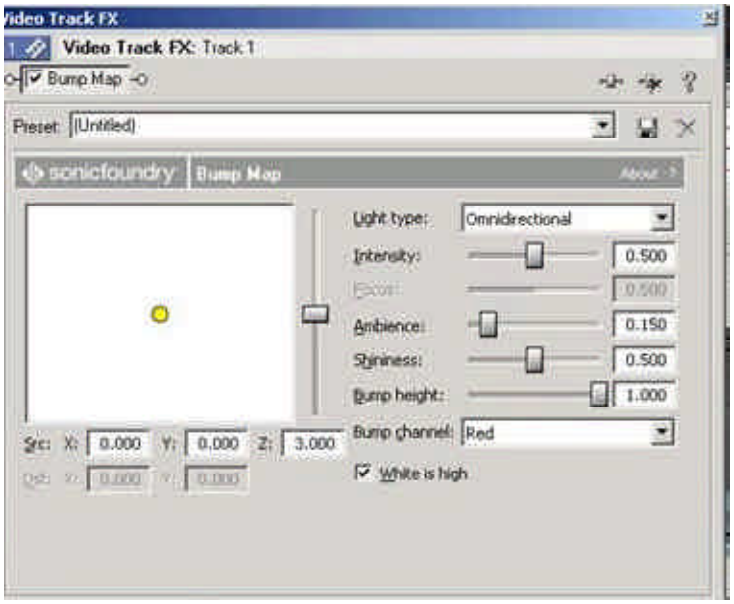


Figure 1

IN THIS ISSUE

Bump Map – Simple Cartoon-look Effect 1
Beginner's Corner - Using the Credit Roll 3
Scripting - Introduction to JScript (Part 3) 5

video tracks composited such that the bump map texture is applied to the child track or it may be used as a standard effect on a single track. In the single track usage, the selected track is both the source and bump map image with highlight regions appearing raised and dark regions appearing as depressions. The Bump Map dialog (Figure 1) provides adjustments for the directionality, location, and quality of the light source. The accentuation of the bump effect is controlled with the bump height slider and the affected video channel may be selected from the RGB and alpha list window.

For this example, I changed the following settings:

1) Light Type to Omnidirectional
2) Intensity to .5
3) Ambience to 1.5
4) Bump Height to 1
5) Bump Channel to Red
6) Light position to X=0 and Y=0

With very little experimentation you can adjust the image appearance to suit your cartoon-look preference. The raw video image (figure 2) was converted into the cartoon image (figure 3) using these settings.

The color video on texture produces an almost pastel chalk on paper appearance. The black contour edges support the hand-drawn illusion. Using the omnidirectional type light source produces finer and spatially uniform contour lines. Directional lighting accentuates one edge more than another.

To maintain a discernible texture without appearing too much like heavily shadowed bas relief, the light source position was set to on-axis (x: 0.000, y: 0.000). Intensity, Ambience, and Shininess were adjusted to give a pleasing tonal range. While the above settings work well for this particular video, you will need to adjust these settings as required for your source.

No additional filters were applied for this example. However, color correction or brightness and contrast could be used to further enhance the cartoon effect.

The Bump Map plug-in doesn't duplicate the quality achieved by hundreds of hours of professional rotoscoping and the thousands of dollars paid for that service, but it does provide a quick and simple way to achieve cartoon-like video approximating the style of rotoscoped movies at no additional cost to a Vegas user.

Figure 2

Figure 3

Joseph Lawrence is a public access TV volunteer camera/audio operator and he produces the Some Shorts program for Access Fort Wayne.

Contact Information

Send your tips, tricks, article ideas, script ideas, questions, or articles to:

editdv@jetdv.com

To register on the web to receive this newsletter, browse to: **www.jetdv.com/vegas**

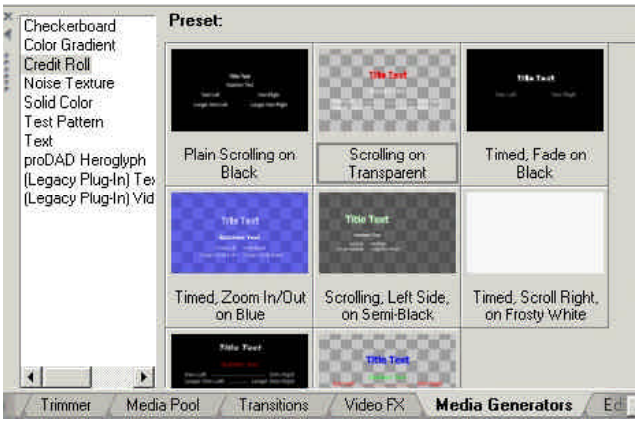
Thank you, Edward Troxel

Beginner's Corner - Using the Credit Roll

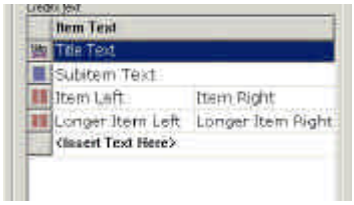
By Edward Troxel

Creating a credit roll is very simple in Vegas. Controlling that credit roll takes a bit more effort. Just by knowing a few tricks, you can gain full control over your credit rolls.

Your fist step is to place the credit roll generated media on the timeline. You have several presets to



choose from as shown in this image. Simply pick the desired look and drag it to the timeline. Remember the ones with the checkerboard backgrounds have a transparent background. Once on the timeline, the main credit roll dialog will open.

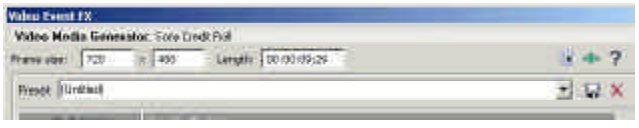


From this dialog, you can enter all the information and change the desired settings for this credit roll. Let's begin by looking at the text section. In this area, you can enter text for either a single entry per line or two entries per line. Notice also that each line can be one of three different types: Header, Single Item, or Dual Item. Click on the box to the left of the input area to pick the proper type for each line. Now simply click on any text area and type the new information.

As you continue to add lines, it may become necessary to either remove some lines or add some additional lines. While simple to do, it is not an immediately obvious process. To remove an unwanted line, simply click on the style selection box to select that line. Now press the **Delete** key on the keyboard. To insert a line, click on the style selection box where you want the new line and press the **Insert** key. Then you can pick the proper style and type in the desired text.

Entering text in the credits text box is not my favorite way of entering the information. Instead, I'd prefer to use Word, Excel, or even Notepad to enter all of the information. Fortunately, that is possible! For all single line items, simply enter the desired text and press Enter. For all dual line items, enter the text for the left side, press Tab, enter the text for the right side, and press Enter. For a blank line, just press Enter. Once all of the text has been entered and verified, simply copy the text, click on the style selection header there the text is to be placed, and press CTRL-V to paste the text. You will still need to remove any unwanted remaining lines and change the style for any header lines, but it is a much simpler method of entering the text.

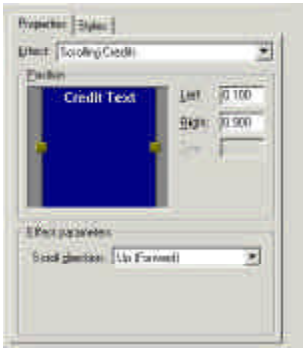
Once the text has been entered, simply close the dialog box and play the timeline. The text will probably not be scrolling at the speed you really wanted. To remedy this, first resize the credit roll event to be the amount of time you wish the scroll to last. Now

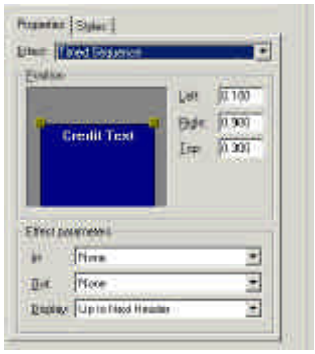


when you play back the timeline, you will see that the speed has not changed - even though you changed the length of the event.

In order for the time required to scroll to actually change, you have to also change the length setting of

the credit roll (same is true for other generated media objects as well.) Open the dialog again and look at



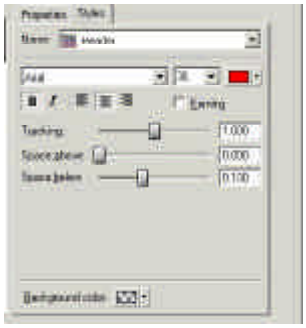


the top. There you will see settings for width and height and a timecode labeled Length. Change the text in the length box to match the new length of the event. Now the scrolling rate will be changed when you play back the timeline.

On the right side of the dialog you will find the **Properties** and **Styles** tabs. The Properties tab will let you select the effect of the credit roll. The two options are ***Scrolling Credits*** and ***Timed Sequence***. For Scrolling Credits, you can choose to have them scroll upward (forward) or downward (backward). You can also specify the left and right margins so the text will remain inside the Safe Area.

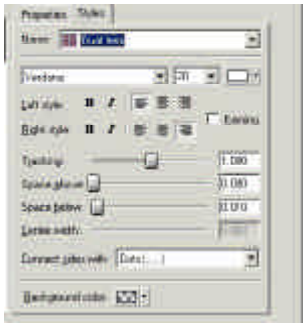
The Timed Sequence offers a different way to get multiple screens of information presented. You can control how each screen of text enters and exits and can also specify the top margin. Between these two methods, a wide variety of options are available.

As previously mentioned, each entry can be one of three styles: ***Header***, ***Single Item***, or ***Dual Item***. Each of these styles can be modified to allow a different look for the three categories. To modify the styles, click on the Styles tab as shown here. Simply choose the style to be modified and change the settings as desired.



On this screen, you can change the font, font size, color, bold, italics, justification, spacing above and below the text, and tracking. Adjust these as needed for both the header and single line styles.

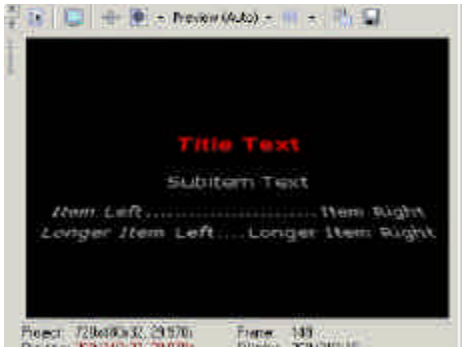
The Dual Item style allows a few more options. You can control some of the parameters individually for the left and right side. Plus, you can set how the two sides are to be connected. This allows you to create a movie style roll where the character's name is on the left and the actor's name is on the right.



One limitation that has bothered many people is the fact that there can only be three different styles. Many times it would be preferable to have more styles available. While you ***are*** limited to three styles, here's a work-around that may work in some instances. If the style you want to change is not on the output screen, simply add a new keyframe and change that style. From that point forward the entries of that type will now get the new style.

Another limitation is how to get a drop shadow. This can be done by clicking on Track Motion for that track and then clicking on the *2D Shadow* button left of the Track Motion timeline.

Take another look at the Credit Roll generated media. You will find there really is a lot of power available to give the results you need - especially when combined with the other features of Vegas. For example, turn on 3D on the track containing the credit roll, change the x-axis rotation to around -40, and you have an instant Star Wars style introduction!



Scripting - Introduction to JScript (Part 3)

By Edward Troxel

Last month we continued looking at the basics of JScript programming. This issue we will continue with the basic background information before actually writing a script.

Functions

In some cases, certain sequences of code are used over and over again. In these cases, it makes sense to isolate that code into a separate area called a function. By giving a function a unique name, it can be called multiple times without having to duplicate the code. The big advantage of having the code in a function instead entered multiple times is maintenance. If a bug is found in the code, a function only needs to be fixed once for all calls instead of trying to locate the same error multiple times.

The basic layout of a function is:

```
function FuncName(any parameters) : Type Returned{
    //function code goes here
}
```

As an example, look at the following definition (the code has been deleted for simplicity but will be shown in a later chapter):

```
function FindTrack(WhichTrack) : Track {
    //function code
    return;
(will return null (or nothing found) to the calling routine) OR

    return track;
(will return the found track to the calling routine)
(We'll look at the full code in another issue)
}
```

Examining the details of this function reveals a lot of information. First, the name of the function is FindTrack. Once again, the name is case-sensitive. In this case, the function has one parameter called WhichTrack. The last piece of information found on the first line is the type of value returned by the function. This function will return a value of type Track.

The return function would be used in associate with if statements to determine what value should be returned. If the proper track is found, **return track;** should be executed to send the found track back to the calling routine. Otherwise, if the track was not found, **return;** is used sending a value of "null".

To use the routine, the calling program would look as follows:

```
MyTrack = FindTrack("Master");
if (MyTrack == null)
    throw "Master track was not found.";
```

The results from calling the function is placed in the variable MyTrack. If MyTrack is equal to null after calling the function, the track was not found and the error is displayed. Otherwise, MyTrack now points to the "Master" track.

By putting this routine in a function, the main routine can now find multiple tracks without having to duplicate code. Use functions wisely to simplify your code.

Displaying Information and Error Catching

In the previous code samples, error checking has already been demonstrated. Look at the statement:

```
if (MyTrack == null)
    throw "Master track was not found.";
```

If the track is found, nothing happens. If the track is not found, error catching comes into play. In this case, the "throw" command indicates an error has occurred and the error message to be displayed. To use error catching, the main code needs to be placed inside a "try" section as shown with this paraphrased if statement:

```
try {
  If an error occurs
    throw "Error Message"

} catch (e) {
    MessageBox.Show(e);
}
```

When the throw statement is executed, the rest of the main section is skipped and the catch section is used. The catch section then displays the error message on the screen before exiting.

The error routine displays the error message by using the MessageBox.Show command. This command can be used anywhere in your code to display messages. Please note that the command name is case-sensitive. To display information, just put it in the parentheses. Look at the following examples:

MessageBox.Show("Hi There"); Displays: **Hi There**
MessageBox.Show(1234); Displays: **1234**
MessageBox.Show(12 + 3); Displays: **15**
MessageBox.Show("Events Found " + I);
Assuming I=3, Displays: **Events Found 3**

Imports

Vegas scripts depend on the .NET framework to operate. Since the .NET framework is built to be expandable, it is necessary to specify what pieces any script needs in order to run. For example, to make the Vegas API available, the line **import Sony.Vegas;** needs to added to the script. Likewise, System and System.IO need to be in many scripts. If using a message box, System.Windows.Forms is also needed. While not every import line is needed for every script, the following list would cover the majority of situations.

```
import System;
import System.Collections;
import System.Text;
import System.IO;
import System.Drawing;
import System.Windows.Forms;
import Microsoft.Win32;
import Sony.Vegas;
```
(or **import SonicFoundry.Vegas;** for Vegas 4)