Vegas + DVD
Tips, Tricks, and Scripts

Learning and using Vegas 4 and DVD Architect

Editing Multi-Cam Events

By Edward Troxel

When taping events, it is typical to use multiple cameras. This gives many advantages by providing multiple angles, a variety of shots, and multiple audio sources. Fortunately Vegas provides a rich environment for multi-camera editing.

The main key to taping multi-camera events is: **Once the cameras are started, never stop them**. If all cameras tape for the entire period of time, the clips will only need syncing once. If the camera is stopped and restarted, the second clip must be re-synced causing more work.

In this example, the event was captured with three cameras. Notice how the sound wave goes from very soft to suddenly very loud. This will be our sync point so all that must be done is aligning this point on all tracks. Alternate sync points may include a clapboard or a camera flash.

One way to align the tracks is to click and drag the clips until they are close. You can then use the "1" and "3" keys on the numberpad to fine-tune the align-ment. Once aligned, the sudden shift in sound should now be in a straight line.

With the help of Excalibur, this process can be further simplified. Simply place markers on the timeline where the sound change occurs, run the Sync Wizard, and the clips will automatically be moved so they are in sync.

Now that the clips have been aligned properly, it is time to start editing the mix. Vegas provides a variety of ways to accomplish this task. Four such methods will be discussed.

The first method works on the principle that the top-most track is always visible. So, to see the lower tracks, all that is required is cutting away the upper tracks. This concept is simple: If you want to see camera one, leave everything alone. If you want to see camera two, delete camera one. If you want to see camera three, delete cameras one *and* two.

Place the cursor on the position where you want to switch cameras, press CTRL-A to select all, and then press "S" to split all tracks. This must be repeated
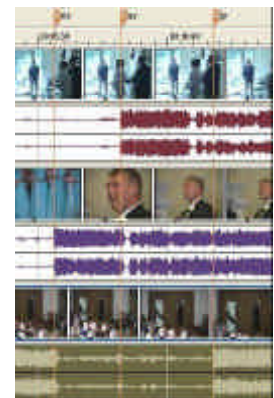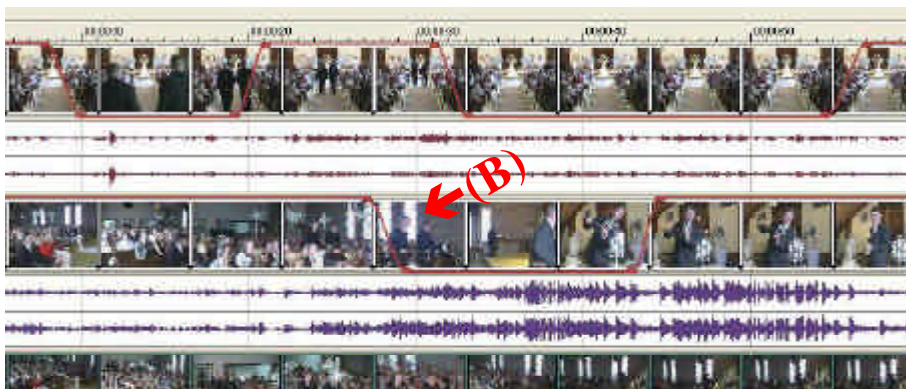
←(A)
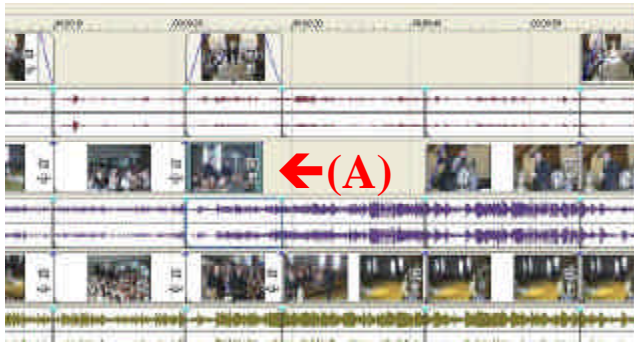
whereever a camera change is desired. After both sides have been split, simply delete the sections to be removed. Then, standard fades can be added to the events to dissolve into the lower camera angles.

In the example above, camera one is visible and fades into camera two. This then fades back to camera one which then fades to camera three. However, note that camera two has been trimmed left of the dissolve on camera one (See "A"). If this area had not been moved left, camera one would have dissolved into camera two instead of camera three as desired. Finally, in the example above, camera three then dissolves to camera two which dissolves back to camera one. This process would be repeated for the entire event.

Working from a similar viewpoint, the same process can be completed with absolutely zero splits to the camera tracks. To accomplish this, a *Composite Envelope* must be added to cameras one and two. Below is the exact same sequence using composite envelopes (shown as red lines).

When the composite envelope is all of the way to the top, that track will be visible. When at the bottom, the lower track will be visible. To adjust the opacity level, double-click the composite envelope line in two loca-

tions to create two points. The first point should remain at the previous level while the secont point will be changed to the new level. Notice how the red line starts at the top on camera one, moves to the bottom (revealing camera two) and then moves back to the top dis-

solving back to camera one. This will be a dissolve because the two points are not straight up and down.

As you continue across the timeline, notice that for camera one to dissolve into camera three, the composite level of camera two must be lowered before the composite level of camera one is lowered (See "B"). As before, if this had not been done, camera one would have dissolved into camera two instead of camera three.

The third method requires the addition of another video track - the "Master" track. In this case, all desired segments are moved to this master track. Similar to the
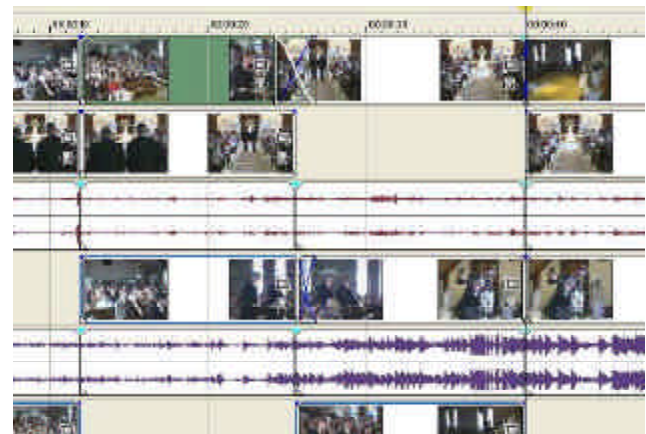
first method, it is necessary to split the tracks at all places where camera changes are desired. Once a segment has been split on both sides, move that segment to the "Master" track.

To move segments, either click and drag the correct camera clip or select the clip and press the "8" key on the numberpad as many times as needed to get the


←(B)

clip to the "Master" track. Once the clips are moved, dissolves can be created by manually resizing the clips or placing the cursor between the clips, selecting only the "Master" track, and pressing "/" on the numberpad which creates a dissolve length as specified in the "***Cut to overlap conversion***" length in *Options - Preferences*.

In this example, the "Master" track starts with camera two, moves to camera three, up to camera one, and back to camera three. Once finished, the "Master" track will contain the final event mix.

The fourth method uses a variation of the "Master" track. This method uses the Multi-Cam Wizard found in Excalibur. While a "Master" track is the end result, no splits will be made until all edit decisions have been completed.
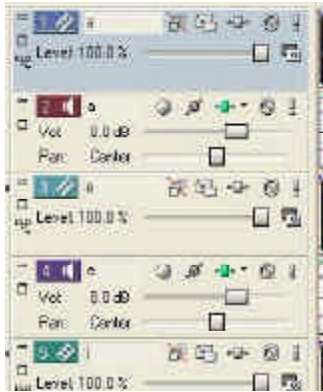
In order for Excalibur's Sync Wizard and Multi-Cam Wizard options to work, the tracks must be properly named. I use simple one letter names. These names are then used as marker names to indicate a sync point or camera change point.
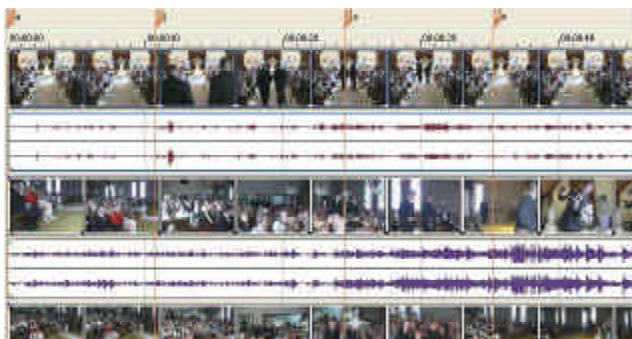
To edit the program, simply place markers where you want to switch to a different camera and name the marker for that camera. Repeat this process for the entire event giving you a series of markers on the timeline. Once completed, run the Multi-Cam wizard to automatically create the "Master" track containing the clips selected. To add to the flexibility, Multi-Cam Wizard includes options to cut by default, dissolve by default, or
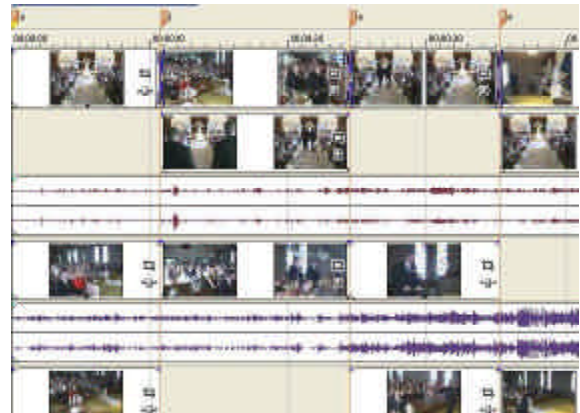
specify a manual dissolve length for any transition. As a bonus, it will even give a count of the number of edits made for this event.

As we have seen, Vegas provides many methods for editing muli-cam events. However, it is often difficult to see exactly what is happening on each camera. While the *solo* and *mute* buttons are helpful, seeing them all on one screen is helpful. When using a "Master" track, I like to create a PIP screen using Track Motion. The only downside to using the PIP view is that it will probably need to be rendered in order to provide a full framerate preview. I find the PIP view very useful.
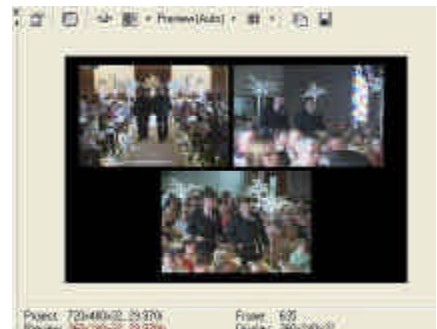
Explore the various options in Vegas and you will find a rich editing environment no matter how many cameras you use. Using these methods, multi-cam editing has never been easier.

Beginner's Corner - Changing the Speed of a Clip

By Edward Troxel

There are times when the speed of a clip needs to be modified to achieve the desired effect. Whether this means making a clip run in slow motion, fast forward speed, or reverse, speed modifications are very easy in Vegas.

There are two main methods available for changing speeds and each has advantages and disadvantages. For simple speed changes ranging from 25% to 400% when the entire clip needs to go a single speed, the simplest method is to resize the clip while holding down the CTRL key.

Suppose you have an initial clips that is about one second long and you wish to extend that clip over a period of four seconds. The quickest way to accomplish this is to hold down the control (CTRL) key and and resize the clip normally. The clip will then be stretched to fill the new length of time creating slow motion. If you look at the clip, a wavy line indicates that this clip has been stretched.

Now let's go the other direction, hold down the control (CTRL) key and make the clip smaller. This will make the clip play faster. Notice that while the pevious wavy line in the clip looked stretched, this wavy line looks like it has been squeezed together indicating the clip will play faster.

Using this method makes it easy to stretch or squeeze a clip to fit in a particular area. However, if you need an exact speed change, right-click the clip and choose properties. In this box you can manually specify any speed change rate from .25 (25%) to 4 (400%).

Although this method of changing speeds is very simple, a much more powerful method is available. By using a *Velocity Envelope*, speed changes range from (negative) -100% (full speed reverse) through 0% (freeze frame) up to 300%.

To use a velocity envelope, right-click the clip and choose *Insert/Remove Velocity Envelope*. This will add the velocity envelope line set to 100% (normal speed) as shown here. To set up a freeze frame, simply lower the line to zero percent. Similarly, to reverse a clip, lower the line to any negative percetage.
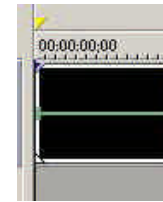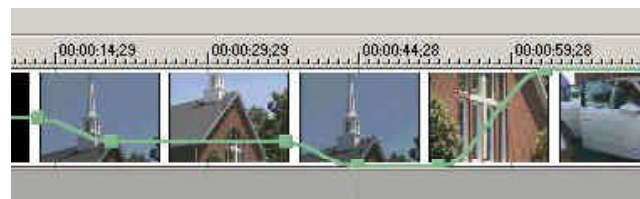
Normal Speed

Freeze Frame

Reversed

While these simple speed changes are very effective, it is just the beginning of what the *Velocity Envelope* can do. As with other envelopes, points can be added to the velocity line either by double-clicking on the envelope or right-clicking and choosing **Add Point**. Once that point is added, it can be moved to a new velocity as well. By adding multiple points, it is possible to have a clip gradually change speeds - a fairly unique ability among NLE's.

Taking a look at this sample clip, the power of the Velocity Envelope can be seen. Starting at a normal speed, the clip gradually slows down to a freeze frame, then down to full speed reverse, and quickly changes to fast forward speed. To verify exact speeds, right-click any point and choose *Set to...* allowing manual entry of the speed (this is especially important for freeze frame).

The speed options in Vegas are very powerful. Experiment with these two methods to explore their capabilities. In fact, try combining both to the same clip to achieve some interesting results.

Adding and Assigning Bus Tracks

By Edward Troxel

When starting new projects, it is frequently easiest to start from a template, or previously saved project, that has many elements already in place. One such example is a mixing project where specific tracks are to be routed through specific busses. Here is a post from the Sonic Foundry forums illustrating this problem:

Because of time constraints and traveling musicians, I have a series of songs that require drums to be recorded after most other tracking has been completed. Backwards from what I usually do but here is my question:

Is there a way to set up the same drum tracks in a dozen songs without having to place and name every track ie. snare, kick, leftOH, rightOH, etc. AND route their inputs. I have the sessions coming up soon and I'm trying to figure a way to save hundreds of mouse clicks.

I usually do drums near the beginning of sessions from a saved template...

Fortunately, all is not lost. With a little scripting, the proper tracks can be added to the timeline, the various busses can be added, and the tracks can be routed to those busses. The complete script in figure A gives the basic guideline. This must then be modified to meet the needs of each individual person.

To accomplish this task, the first step is to add each of the individual busses to the project. This can be accomplised using three lines of code for each bus:

```
var Bus1 = new AudioBusTrack();
Vegas.Project.BusTracks.Add(Bus1);
Bus1.Description = "Snares";
```

The first step is to create a variable for this new bus and assign it as a new Audio Bus Track. Once the track has been created, it is then added to the project. A project allows several different types of tracks so this track must be added specifically to the Bus Tracks. The Add command accomplishes this task. Finally, the bus description is set to a meaningful name. While not necessary, it will make it easier to locate the correct bus if any manual assigning of busses is required.

This process must be repeated for each bus you need to create. By assigning each bus a different variable name, we can then easily assign tracks to any particular bus as needed.

Once all of the busses have been created, it is time to start adding tracks. These three lines of code can be used for each track:

```
var Snare = new AudioTrack(0, "Snare");
Vegas.Project.Tracks.Add(Snare);
Snare.BusTrack = Bus1;
```

Like the bus track had to be created and assigned to a variable, the audio track must be created in the same manner. The new Audio Track will be created, in this case named "Snare", and assigned to the variable. The Add command then adds the Audio Track to the standard track list of the project. Finally, this new track is assigned to a specific bus.

Let's compare these two similar but different tasks. First, they are both adding new tracks. However, the first segment specifies an "AudioBusTrack" where the second segment specifies an "AudioTrack". While the code looks nearly identical, they perform two very different tasks.

Similarly, they are both added to the project in almost identical methods. The busses are added to Vegas.Project.BusTracks while the audio tracks are added to Vegas.Project.Tracks. Identical methods but two totally different target locations.

For the busses, the third line simply gives the bus a name. For the audio tracks, the third line tells where the audio should be routed. By default, all new audio tracks are assigned to the main project mix. This line is required to, instead, set the assignment to the bus.

And the reply to this script:

Brilliant! It does exactly what I need it to do and now that I can see how it works,. I can set up as many lines of code and it will do the whole drum kit, just as I wanted.

Complete Add Busses and Tracks Script - Figure A.

```
/**
 * This script will add busses and tracks assigned to those busses
 * Written By: Edward Troxel         Modified 07/17/2003
 * www.jetdv.com/tts
 **/

import  System.Windows.Forms;
import  SonicFoundry.Vegas;

try {
   var Bus1 = new AudioBusTrack();
   Vegas.Project.BusTracks.Add(Bus1);
   Bus1.Description = "Snares";

   var Bus2 = new AudioBusTrack();
   Vegas.Project.BusTracks.Add(Bus2);
   Bus1.Description = "Kicks";

   var Snare = new AudioTrack(0, "Snare");
   Vegas.Project.Tracks.Add(Snare);
   Snare.BusTrack = Bus1;

   var Snare2 = new AudioTrack(0, "Snare 2");
   Vegas.Project.Tracks.Add(Snare2);
   Snare2.BusTrack = Bus1;

   var Kick = new AudioTrack(0, "Kick");
   Vegas.Project.Tracks.Add(Kick);
   Kick.BusTrack = Bus2;

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