Review: Sample Modeling™ The Trumpet
Created by Giorgio Tommasini & Peter Siedlaczek
by Rick Paul - *5th May 2008* -

You may not have heard the company name Sample Modeling before. However, if you've been keeping an eye on the world of sample libraries and sample-based instruments, you have no doubt heard of its two principals, Dr. Giorgio Tommasini and Peter Siedlaczek. Tommasini is one of the inventors of patent-pending technology previously employed in the excellent Garritan Stradivari Solo Violin (GSSV) and Garritan Gofriller Solo Cello (GGSC) software instruments, which dramatically raised the bar for playability and believability of solo violin and cello emulation, respectively. Siedlaczek has been among the preeminent developers of orchestral sample libraries since the days of hardware samplers, with titles bearing his name including Advanced Orchestra, Classical Choir, and Smart Violins, among others.

What happens when you take an experienced, respected sample library developer, pair him with a technological innovator who has already helped take the sample-based instrument industry to a new level, and send them out on a mission to create a highly playable, believable software instrument, or rather a set of instruments, that is notoriously difficult to simulate due to the wide range of expressive capabilities it makes available? That's what we're here to learn as Siedlaczek and Tommasini tackle "The Trumpet".

Background

At its most basic level, The Trumpet is a family of trumpet and trumpet-like software instruments based on the Native Instruments Kontakt 2 (or Kontakt 3) sample playback engine. Individual instruments include three different B-flat trumpets, a cornet, a flügelhorn, a German trumpet, and a piccolo trumpet. The three B-flat trumpets can also be played with or without a variety of mutes, including straight, bucket, cup, and Harmon mutes.

The Trumpet is the result of many years of research by Giorgio Tommasini and Peter Siedlaczek. As the company name, Sample Modeling, might suggest, The Trumpet employs techniques and technology that borrow from the worlds of both sampling and physical modeling. Sonic realism comes from chromatically sampling actual acoustic instruments, played by a high caliber musician, who also recorded typical articulations and phrases for analysis purposes. Where typical sample libraries might provide alternate samples with different articulations, The Trumpet uses an "adaptive model" to reconstruct all articulations, and morph between sounds as needed to reflect multiple characteristics of tone and playing style (e.g. dynamics, vibrato, legato, and portamento) in real time. This model takes into account the physical properties of the instrument and performance characteristics. Advanced scripting analyzes what the user is playing, then employs the sampling and modeling techniques needed to recreate articulations and phrasings that are virtually indistinguishable from an instrumentalist playing the actual acoustic instrument. As with a real trumpet, no two instances of the same note from The Trumpet will sound identical due to the continuous pitch, dynamics, and timbral modulations used in Sample Modeling's approach.

The Trumpet includes the Native Instruments Kontakt 2 Player sampling engine for out-of-the-box use. Sample Modeling also includes Kontakt 3 versions of the instruments in the package for those users who already have the latest version of the NI sampler and would prefer to use The Trumpet that way. Note that changes were necessary to make The Trumpet work properly under Kontakt 3, so the default Kontakt 2 instruments should not be used under Kontakt 3.

The Trumpet is available for Windows XP (or higher) systems in the VST and RTAS plug-in formats. Plug-in formats on the Mac include VST, RTAS, and AU. A standalone version of the Kontakt 2 Player is also included for those who would prefer to use The Trumpet as a live performance instrument without a DAW. It is important that the system used to run The Trumpet have a 24-bit audio card with low (less than 7 ms) latency. While specific minimum CPU ratings are not provided, because The Trumpet uses state-of-the-art techniques, Sample Modeling highly recommends running it on a state-of-the-art system. As an example, they reference having thoroughly tested The Trumpet on a PC with an Intel Core2 Duo E6600 CPU at 2.40 GHz with 2 GB RAM, Windows XP, 2 SATA hard drives, and a 24-bit audio card with low latency ASIO drivers, and running a CPU load of approximately 7-12 % on that configuration. That said, they also indicate that The Trumpet may work satisfactorily on lesser systems.

Other important requirements for running The Trumpet include a master keyboard with at least a pitch wheel, modulation wheel, and an expression pedal mapped to CC#11 for real-time playing. In fact, if The Trumpet has not yet sensed CC#11 information being transmitted to it, it puts up an error message to let you know that the instrument will not work properly. While this might seem like it is getting a bit too adamant, I find it is actually helpful information in the case where the instrument isn't transmitting sound, and you're not sure why. Sample Modeling also recommends using a MIDI controller with multiple configurable controls for sending CC information as The Trumpet also uses a number of additional continuous controllers that are less critical for most playing. See below for more information on CC usage.

Sample Modeling sells The Trumpet direct, with fulfillment through Digital River's Share-it! service. Customers based in the USA will pay $240 (USD), British customers will pay £120 (plus VAT), and European customers will pay €149 (plus VAT). Customers in other countries can choose to pay in one of these currencies (with no VAT outside the EU), in which case any currency conversion will be done by their financial institution (e.g. credit card company). Alternately, they may pay in any of the other international currencies supported by Share-it!, in which case Share-it! will do the currency conversion at an added cost. Despite Sample Modeling's web pages' showing what looks like an extremely nicely designed product box (see the image at the top or bottom of this review), The Trumpet is sold as a download-only product. A "backup CD" is, however, available at extra cost. By default you are given a window of a few weeks to download the software from a link sent to you by e-mail from the Share-it! service. However, there is also an extended download option, at extra cost, to allow making the link valid for up to two years. Unless you are on a very slow Internet connection, your best bet will be to forego the backup CD and extended download options, but back your software up to CD-R as soon as you download it.

Once you've gone through the purchasing transaction, you will receive an e-mail from the Share-it! service with a serial number for the product, as well as instructions for downloading the product, which include the URLs you'll need for the WinXP or Mac versions and estimated download times and sizes. The Windows version weighs in at 301.7 MB, while the Mac version is 333.3 MB. The download times predicted may seem a bit scary (e.g. for the

Windows version, my notice predicted 703 minutes for modem/ISDN, 55 minutes for DSL/cable, or 27 minutes for T1/1.5Mbit). In practice, though, actual download times are likely to be considerably quicker. For example, on my DSL connection, it took only 32 minutes to download the Windows version -- i.e. about 58% of the estimated time, and I know cable download rates tend to be quite a bit quicker than DSL.

Both the Windows and Mac products are delivered in ZIP archives. Once you've downloaded the file, you simply extract the files from the archive, use the provided installer to install the files for The Trumpet and the Kontakt 2 Player (you can skip the player installation if you have a recent enough version of the player installed already). The installer allows placing the Kontakt 2 Player software and The Trumpet's sample library in different locations. Once the installer has completed it is necessary to manually copy the documentation from the installation area to a folder created by the product installer. It would have been nice if the installer program had also set up the documentation files, though copying them manually is only a minor inconvenience.

The documentation for The Trumpet is supplied in the form of six Acrobat (PDF) documents. While many developers might include five of these six documents as chapters within a single paper or on-line manual, I actually prefer having them as separate PDF documents. The reason is because most of the documents are not likely to be needed frequently while using the product. Separating the documents out into individual PDF files as Sample Modeling has done allows loading only the information you need at any given time. In fact, there are a few short documents you might even want to print out for reference such as "THE TRUMPET Keyswitch Assignment", which is a single page document with a graphic depicting the keyswitch layout used, and the "Quick Start" guide, which includes keyswitch information (without the graphic layout) and adds MIDI continuous controller assignment information and some other helpful information (e.g. how to access mutes).

Product authorization is via the standard NI Service Center. Even if you already have an up-to-date Kontakt 2 Player installed, you will need to use the Service Center to authorize The Trumpet. Prior to being authorized on your system, the product will run for up to 14 days.

Kontakt 2 Player support comes directly from Native Instruments. Sample Modeling also has user forums available for questions specific to The Trumpet, user demos, and more.

I conducted my testing of The Trumpet under Cakewalk's SONAR 7 Producer Edition, version 7.0.3 (this represents the standard patch update to 7.0.2 then an additional "only if needed" patch provided by Cakewalk to address one specific issue). The test system has an Intel Core 2 Duo E6600 CPU with 2 GB of RAM and Windows XP SP2 (fully patched to latest Windows Update levels). The audio interface was the E-MU 1820M, running v2.0 DAS drivers and PatchMix software.

Technology Refresher

The core technology underlying The Trumpet builds upon and enhances that originally employed in the Garritan Stradivari Solo Violin (GSSV). You can read about the earlier technology in more detail in the "Building a Better Sample Violin" section of my review of GSSV. However, I will provide a brief synopsis here for convenience.

When simulating a complex instrument like a trumpet on a sampler, some of the inherent challenges include achieving:

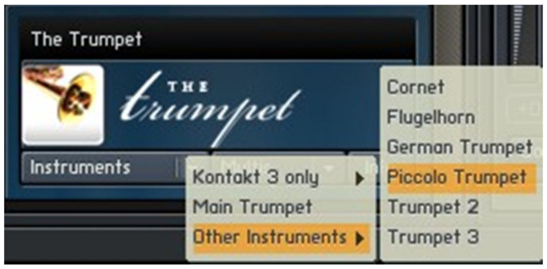
- Real time playability
- Real time, continuous transition (morphing) across several dynamics of the same note
- Real time, continuous transition between vibrato levels
- Real time control of the vibrato intensity and rate
- Real time portamento and legato
- Real time construction of all articulations
- Timbral characteristics indistinguishable from the original samples

At the architectural level, The Trumpet employs the same techniques as GSSV to address these objectives. In particular, it builds upon the KONTAKT 2 sampling engine, script processor, and convolution effect as follows:

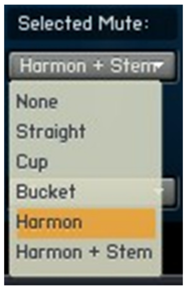
- Pre-processed samples, which are phase aligned for each harmonic by the sample library development team, are used to allow crossfading without artifacts.
- A specially constructed impulse response, derived from in-depth analysis of the sounds produced by the instrument (not just its body characteristics), is used to recreate realistic vibrato and portamento from non-vibrato sounds.
- A MIDI processing script analyzes what the player is doing (e.g. with respect to note overlap, timing, etc.), and translates this into a sophisticated series of commands to the sampler to emulate trumpet techniques.

The Instrument

While the core technology may be similar to what has been used in the Garritan Stradivari Solo Violin and Gofriller Solo Cello, Sample Modeling has applied that technology in new areas, and developed some innovative new playing techniques, with The Trumpet. Let's take a look at key features of The Trumpet and how they are accessed in real-time and/or sequenced playing.



At the very highest level, once you've instantiated the Kontakt 2 Player (or Kontakt 3, if you prefer), you load one of seven instruments. There are three different B-flat trumpets, making it easy to build sections. Other instruments available include a cornet (similar to a trumpet, but with a conical, rather than cylindrical, bore, and a somewhat mellower sound), flügelhorn (also with a conical bore, and an even mellower, darker tone), piccolo trumpet (half the tubing length of the B-flat trumpet, excels at playing higher parts), and German trumpet (rotary valve instrument with a more tapered attack, often favored in German and Austrian orchestras).



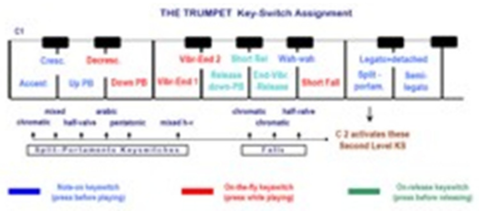
By default, all instruments are unmuted. The standard B-flat trumpets also have multiple mute options, including straight, cup, bucket, and Harmon mutes. The Harmon mute can be configured with or without the stem. While mutes can be turned on and off in real time through the Kontakt 2 Player interface using a menu selection, they are not programmable via MIDI. Thus, if a part requires the instrument to be unmuted part of the time and muted part of the time, it would be necessary to load two (or more, if multiple types of mutes are needed at different times) instances of The Trumpet, with each on a different MIDI channel. The muted and unmuted parts would then be placed on different MIDI channels in the sequence driving the instrument. I would have preferred to have this capability be available under MIDI control.

There is one additional type of mute that is under MIDI keyswitch control (and also available only on the B-flat trumpets). This is the plunger mute, and two different types are available. (The Trumpet's manual suggests presumably one is used and one is unused. This notion makes me glad we are talking about a virtual instrument.) When the A#1 keyswitch is engaged CC#11 essentially becomes a wah-wah pedal, simulating the movement of the plunger toward and away from the bell of the trumpet. The velocity with which the A#1 keyswitch is struck determines which of the two plunger mutes is employed. Note that the plunger mute keyswitch must be struck before the start of a phrase for the wah-wah effect to work on any notes in the phrase. If you strike it in the middle of a phrase (i.e. set of overlapped notes) and hold it, it will start working only on the next phrase. While I would not expect this to be a limitation in practice, it is different from how an actual plunger mute would work.

Muted or unmuted, most common trumpet playing techniques are accomplished simply by playing your MIDI keyboard while using an expression pedal, modulation wheel, and pitch bend wheel in parallel. For example, instrument attack is controlled via note-on velocity on the keyboard. Slurring versus tonguing is governed by whether notes overlap or not. Within a phrase (i.e. set of overlapped notes), whether notes are played legato or with portamento, and portamento time when portamento is played, are controlled by note-on velocity. Instrument volume is controlled via the expression pedal (CC#11). Vibrato intensity is controlled via the mod wheel (CC#1), though at very high mod wheel settings you get shakes, complete with a natural vibrato-to-shake transition. Pitch bend, up to two semi-tones up or down, is controlled with the pitch wheel. Most of what "just naturally happens" when you play these various techniques, such as smoothly transitioning between dynamic levels, non-vibrato and vibrato, non-portamento to portamento and back, and so on, simply happens behind the scene thanks to the technology used in The Trumpet.

Certain, somewhat more advanced, techniques require a bit of extra work on the part of the user, though. For example, controlling vibrato rate requires use of an extra continuous controller (CC#19). Even if your keyboard has a slider or wheel to control this, you may be out of hands by this time, so some of these controls may be best overdubbed or programmed in a sequencer. Other techniques use keyswitches, for example to invoke real-time note release characteristics, such as a slight vibrato on note release (A1). Let's take a quick look at some of the playing techniques and how they are implemented within The Trumpet.

In the continuous controller category, some of the controls provided affect default levels of an effect that is provided automatically when simply playing The Trumpet. For example, CC#20 controls the depth of pitch modulation on note attacks, in case you want to exaggerate the effect or minimize it. Similarly CC#22 controls on-transition flutter intensity, CC#26 controls the duration of attack pitch modulation and note-on keyswitches, and CC#27 controls the duration of default release times (and note-off keyswitches). Other controllers are used to explicitly create special effects. For example, CC#21 controls continuous flutter intensity for "dirty sound" effects, and CC#23 controls frullato, or flutter tongue intensity. CC#25 is used to link initial note volume to velocity, rather than CC#11, with a ramp up or down from the velocity-based setting to the CC#11-based setting. The value of CC#25 is used to control the intensity of this effect, varying from no ramp at all (i.e. initial attack control is determined only by CC#11) at CC#25=0 to complete control (i.e. attack and initial dynamics determined soley by velocity, then linear progress to the CC#11 value) at CC#25=127. This makes it easier to control effects such as sforzandos, which would be very difficult to play purely by manipulating the expression pedal in real time.



Other advanced playing techniques are controlled via keyswitches. The Trumpet uses keyswitches to make it possible to easily perform articulations that would be impossible, or too difficult, to perform with other controls. To this end, The Trumpet provides two high-level types of keyswitches, modulating keyswitches and non-modulating keyswitches.

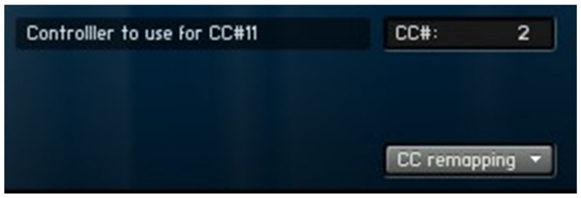
Modulating keyswitches modulate the attack or release of the note. Within this category, there are four subcategories. Note-on keyswitches affect the next detached note when held down. C1 results in the note's being accented, and D1 results in an upward pitch bend into the note. Legato-portamento keyswitches also affect the attack of the note, but affect both detached and overlapped (legato) notes. C#1 results in a crescendo into the note's initial volume. On-the-fly keyswitches immediately affect the note that is being played, causing a note ending to occur in the process. D#1 results in a decrescendo of the note being played, E1 results in a downward pitch bend, while F1 and F#1 result in different types of vibrato note endings. Note-off keyswitches affect subsequent note releases when they occur, but do not result in immediate note releases. G1 results in a downward pitch bend on note release, G#1 results in a short note ending, and A1 results in a vibrato release. In all cases of modulating keyswitches, the note being played is actually modulated, rather than playing back some kind of stock articulation or ending. This results in a much more natural result than would just splicing on prerecorded articulations. The intensity of the modulation effect is controlled by note-on velocity for the keyswitch.

Non-modulating keyswitches are used for several specific functions. A#1, which invokes the plunger mutes, has already been mentioned above. B1 creates a default fall off, with duration determined by keyswitch velocity, though releasing the keyswitch or playing a new note will also end the fall off immediately. C#2 and D2 override The Trumpet's default behavior of creating legato or portamento, depending on note-on velocity, when notes are overlapped. Instead, detached notes (e.g. for staccato behavior) or semi-legato notes, respectively, are created. The basic idea with these is that it is very difficult to avoid accidental overlapping of notes when playing very fast passages, so this makes it possible, for example, to play rapid staccato patterns. C2 changes portamento behavior to create automatic, harmonically based split portamento when it would be more natural based on the intervals between notes. This saves the user from having to figure out what intermediate notes need to be played to create a natural effect. C2 can also be combined with additional keyswitches to create additional types of automated split portamento and falls. For example, C2 and D#1 together produce an Arabic scale split portamento, while C2 and A#1 together produce a half-valve fall.

The Trumpet's innovative use of real-time, velocity-sensitive keyswitching, with behaviors that can be invoked after the start of a note, provides the potential for creating extremely expressive playing, using articulations and modulations beyond what we normally consider to be within the range of expressiveness available through keyboard controllers. As an added bonus, the keyswitches are laid out on the keyboard in such a manner that, while perhaps not inherently intuitive (is there ever such a thing for keyswitches?), quickly starts to feel logical, and thus becomes playable in real time. Now, if only we could grow an additional arm to play keyswitches with one hand while dealing with vibrato on the mod wheel in the other... This is of minor inconvenience, though. For live performance, I would suggest temporarily giving the keyswitches priority over vibrato control any time special articulations are needed. For recording, it is easy enough to play one set of controls in one pass and overdub or manually add the second set of controls in a second pass. (My tendency in this case is to play the vibrato along with the notes, then overdub the keyswitches.)

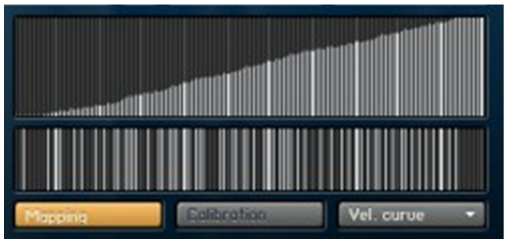
A Little More Control

The Trumpet provides several additional features for monitoring, making general adjustments, and customizing The Trumpet and its Kontakt 2 Player instance to your needs. Let's take a look at these.

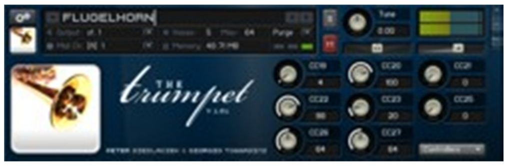


Controlller to use for CC#11 — CC#: 2

Given the importance of MIDI continuous controller #11 to The Trumpet's operation, those who don't have an expression pedal, or who might prefer to use another controller, such as a breath controller, for these purposes, will be relieved to know they won't be left hanging. The Trumpet allows remapping another CC number to CC#11. For example, the screen shot at right maps CC#2 to CC#11 to allow a breath controller to drive The Trumpet.



All MIDI keyboards respond differently, and an uneven velocity response can affect the way a software instrument responds. To help compensate for this, The Trumpet provides velocity remapping, which is turned off by default. There are two ways to use this feature. One is to draw a velocity-mapping curve with your mouse on a graph provided in a special Velocity Curve view. This allows you to tailor the curve to your liking. The second way is to use an automatic calibration capability. To use this, you press a Calibration button, then hit any key on your keyboard repeatedly with random velocities, trying as best you can to cover the entire velocity range of your keyboard over time. This progressively updates the velocity curve graph as new velocities are sensed. Once the graph is as filled in as it's going to get, you press the Calibration button again to turn calibration mode off, then press the Mapping button to turn velocity remapping on. You can also modify the velocity map you automatically calibrated if you'd like to override parts of the automatically calibrated map.



The Trumpet also provides a Controllers view. This view allows you to easily monitor what is going on with continuous controllers on your keyboard, for example to troubleshoot issues when a controller may be sending out more information than you think it should (a problem I have seen occasionally with one of my keyboards). The Controllers view also allows setting values directly with the mouse. Note, though, that the control manipulation capability this provides will be best used with "set it and forget it" parameters as SONAR's automation writing capabilities do not record changes you make to these controls. (Automating these controls in SONAR would be simple to do by either sending CC events from the MIDI track itself or creating MIDI envelopes to automate any CC values desired.)

Playing Around

I always like to try out the products I review on real life recording projects. While I don't often have projects that need trumpets or other similar instruments, it just so happened that I was working on just such a project when Giorgio Tommasini approached me about reviewing The Trumpet. In particular, this project would feature a Dixieland-style combo as part of the backing for a song with vocals. While the Dixieland winds (cornet, clarinet, and trombone in this case) were technically part of the backing tracks, they were featured prominently, with the cornet part front and center due to its range and relative volume. In my mind's eye, I was picturing a New Orleans street corner band, but with Louis Armstrong visiting from that great jazz hall in the sky to sit in on the cornet. To put it another way, I wasn't looking for a synthy-sounding part, or even a straight classical cornet or trumpet part, but a highly expressive part with some growl and real emotion. Would The Trumpet be able to deliver?

Prior to receiving The Trumpet, I'd been working on all three Dixieland wind parts using instruments from the Garritan Jazz & Big Band collection. I should be clear that these instruments are no slouches with respect to their potential for creating expressive parts. However, the Garritan Stradivari Solo Violin and Garritan Gofriller Solo Cello represented major strides over the corresponding instruments in the Garritan Personal Orchestra, and I was hoping Sample Modeling's The Trumpet would make a similar advance over earlier instruments of the types it covers. I would not be disappointed.

Anytime I get a new "toy", my first inclination is to play with it, not worrying yet about what I'll actually end up doing with it later. Usually the "toys" themselves will provide some inspiration with respect to possible uses. In this spirit, my first effort with The Trumpet was to fire up its standalone mode and try out the various instruments. I quickly found myself playing the lead trumpet parts from Maynard Ferguson's version of "Gonna Fly Now" (perhaps better known as "Theme from *Rocky*"), the introduction to Richard Strauss' "Also Sprach Zarathustra" (made immensely popular through Stanley Kubrick's *2001: A Space Odyssey*), and various Tijuana Brass songs, then switching over to the flügelhorn for Chuck Mangione's "Feels So Good". Yeah, I know that's probably not all that creative, but what can I say, I grew up on 70's pop radio! Besides, sometimes cliches can be fun, and this was one of those times. The bottom line is each of the instruments sounded excellent, with respect to both the faithfulness of the tone to the instrument being modeled and to the quality of the tone on that instrument.

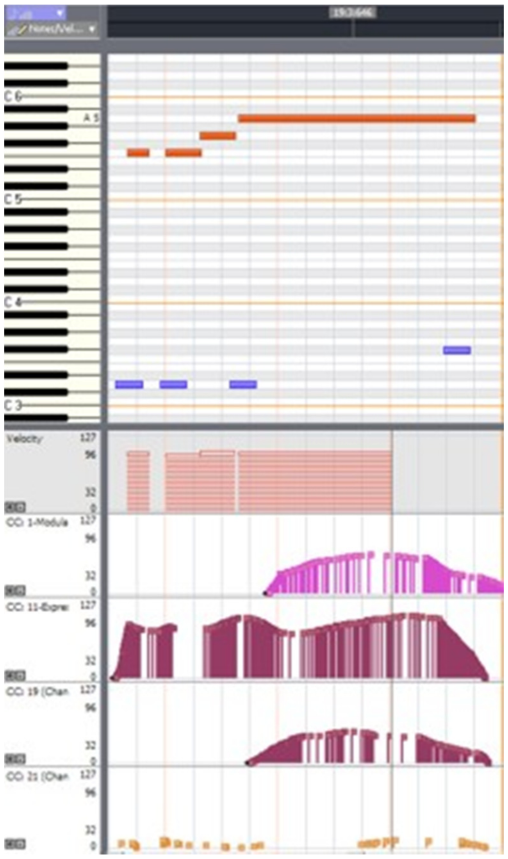
Thus far, I'd only been playing The Trumpet in a very basic fashion. I was using the keyboard for the notes themselves, as well as for velocity-controlled attributes like note attack strength and portamento time, while controlling dynamics with an expression pedal and controlling vibrato rate with the mod wheel, occasionally using the pitch wheel for bends. Even using just these basic controls, The Trumpet sounded good, and it was clear that the technology behind the instrument would bear out my hopes and expectations, making The Trumpet the new instrument to beat for software emulation of the instruments it covers.

Printing out the ReadMe document that serves as a quick reference to The Trumpet's use of MIDI controllers and keyswitches, it was time to dive into some of The Trumpet's more advanced capabilities. This is where The Trumpet really surprised me, and in a very good way. Using keyswitches is nothing new for modern sample-based instruments. Most instruments use these to change articulations, switch modes, and so on. However, most such use is akin to quickly switching between sample sets (e.g. straight strings versus tremolo strings, muted versus unmuted brass, pizzicato versus bowed strings, etc.). The Trumpet takes keyswitch control to a new level. See above for details on what The Trumpet's keyswitches do, but the important notion to come to grips with in terms of playing the instrument is that The Trumpet's keyswitches are more akin to playing techniques than sample set switches. The body of the sample generally stays the same, but note articulations and/or endings, and/or other special techniques, are invoked by the switches. Some are even have immediate effect while a note is in progress. Additionally, the notion that keyswitch note velocity affects the strength of the keyswitch's articulation dramatically expands the expressive potential.

Experimenting with the keyswitches was a lot of fun, and I found the layout became intuitive after awhile, making it possible to actually incorporate keyswitches in real-time playing. There was a catch, however. Not being an octopus, I am limited to two hands, so making extensive use of the keyswitches would come at the expense of controlling vibrato and pitch bend in real time. Whether that was a good tradeoff or not would depend on the musical context. Of course, for sequenced parts, it is reasonably irrelevant, other than for limiting how much can be tracked via real-time playing, since anything not tracked during real-time playing can be overdubbed or manually entered later. Adding the keyswitches to the other capabilities of The Trumpet took The Trumpet beyond just the notion of being the new product to beat for the instruments it emulates toward making it a significant advance in the state-of-the-art for software instruments emulating acoustic instruments.

At this point I was highly inspired to start putting The Trumpet, more specifically its cornet, to work in my Dixieland project. I decided to scrap the part I'd had in progress prior to that point and track a new part from scratch. I started by playing the part live to see how far that would take me. I decided not to try to track keyswitches live, preferring to use my left hand for controlling vibrato while playing, then adding keyswitch-based articulations later. While the part I'd tracked was already sounding pretty good, it was still a long ways from what I had in mind for this style of playing. Part of this was simply that I was new enough to The Trumpet that I hadn't yet gotten the degree of control over playing it to get exactly what I wanted, even not considering the keyswitches and extra continuous controllers. Of course, the other part was needing to add those parts to take the performance the rest of the way toward the level of expressiveness I was seeking.

My next pass at that point was to refine phrasing and fix any playing mistakes. The Trumpet uses note overlaps to distinguish between tonguing and slurring. This is a very intuitive way of approaching this, but not necessarily something a piano player like me gets right the first time. Thus, one task was to extend some notes and shorten others as needed to achieve the desired phrasing. I also found my control over the expression pedal for dynamics left a lot to be desired (see below for more on this), so I ended up manually redrawing many of the CC#11 curves in my sequence for finer control. While mod wheel controls vibrato intensity, it does not control vibrato speed. That is controlled via CC#19, so I manually added data for that. I also wanted to use hints of growl, more at some points than others, so I added CC#21 (continuous flutter intensity) data manually to achieve that.



It was finally time to start playing with key switches. I decided to place those on another track routed to the same MIDI channel in case I ever needed to transpose my tracks to another key (e.g. to suit another singer -- i.e. since the keyswitches would need to remain constant across key signatures). I wasn't 100% certain what I wanted in this area, so I went through the song a little at a time, experimenting with articulations and ending types to see what worked best in the context of the song. Of course, things got a lot quicker as the song progressed and I started learning what worked in various contexts, but part of the fun here was in the experimentation.

If you'd like to get an idea of what the resulting data looked like, have a gander at the screen clip at left, which is a snapshot of one phrase from SONAR 7's Piano Roll View. In the note pane, the red notes are the actual notes the instrument is playing, while the blue notes are keyswitches. The controller panes display, from top to bottom, note velocity (note attacks and portamento speed), CC#1 (vibrato depth), CC#11 (dynamics), CC#19 (vibrato speed), and CC#21 (continuous flutter intensity). Of course, to paraphrase a well-known saying, looking at music is like dancing about architecture. Thus, if you'd like to hear the result, you can check out "Fixin' the Hole in the Wall" (words and music by Michael J. Parker and Rick Paul, © 2008 Michael J. Parker-BMI/Closet Cowboy Music-ASCAP). You can draw your own conclusions, but, to my ears, this was a dramatic improvement over what I'd been likely to achieve with the instrument I'd initially been using on the project.

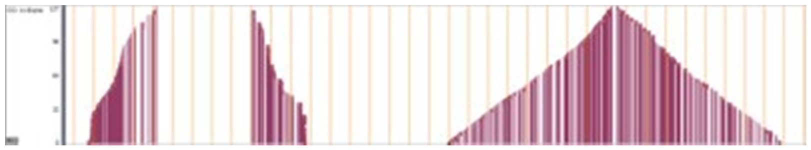
Mind you, while incorporating The Trumpet into this project went very quickly, and I was extremely happy with the results, it wasn't all smooth sailing. One early problem I noticed was that anytime I stopped SONAR's transport in the middle of a note being played by The Trumpet, that note hung until such time as I retriggered the same note to make it cut off. Also, if I restarted playback with it still in that state, until I got to the point in playback where the same note was retriggered, the rest of the part The Trumpet was playing was highly dicey. (This is because The Trumpet is a monophonic instrument, and it thought the hung note was still being held, thus generating many transitions to and from that note as other notes came and went.) It turns out this problem was due to an instrument level configuration setting in each of the instruments in The Trumpet and SONAR's behavior on stopping its transport. SONAR apparently sends out a MIDI All Notes Off command, but does not send individual Note Off messages for notes in progress. The instrument-level settings have an option to respond to All Notes Off, and setting this cleared the problem. Note that it must be done each time an instrument is loaded. (Sample Modeling intends to make this setting the default setting in a future patch update.)

Another issue I encountered is that, while The Trumpet was working fine while I was working at the ASIO latency settings I use for tracking softsynths (typically in the 4-7 ms with the E-MU 1820M audio interface on a 44.1 kHz project), it was a different matter after I finished tracking and set my ASIO latency to the setting I usually use for editing and mixing (50 ms at 44.1 kHz). At that latency I noted a number of artifacts in the sound. About the best I can describe them is that they reminded me of the kinds of clicks and pops I recall hearing when playing vinyl records. Not only did these glitches exist on playback, but they also showed up in rendered audio, with visually noticeable glitches in the waveform. It turns out that, at least on some systems, it is extremely important that the audio card latency be set quite low when using The Trumpet, even when using the Fast Bounce method of rendering audio. The good news is that there is an obvious workaround for this problem -- i.e. keeping the audio latency low while working with The Trumpet at the MIDI level. The bad news is this is not always practical, especially for users with slower systems. It may force compromises such as archiving or freezing other tracks and/or turning off plug-ins while working to finalize the parts being played by The Trumpet, which may slow down workflow if the decisions you'll make in trumpet parts might have been different had fuller context been available. Even on my reasonably modern system (based on the Intel Core 2 Duo E6600), I quite often have more going on in a mix than my CPU can handle at low latency. It is not clear to me whether this issue is an inherent issue with Kontakt 2, perhaps in the scripting area, or whether the issue resides in Sample Modeling's scripts or some other aspect of The Trumpet's instrument implementation. (NOTE: Giorgio Tommasini indicates this is a general Kontakt 2 issue.) I do hope that it can be resolved in the future, though, as the workaround may not be convenient and may sometimes be impractical.

While The Trumpet is capable of great things through a combination of playing and programming, how much fun it is to play also begs the question as to what kind of results it can yield purely through playing the instrument live. In fact, Sample Modeling touts the potential for playing the instrument live, so the desire to use it this way is not just wishful thinking on my part. To look into this possibility, I revisited an old "for fun" project I'd put together a number of years ago in Cakewalk's Project5. The project was an instrumental rock-flavored version of Schubert's "Ave Maria" with ACID-ized drum loops, three softsynths for backing instruments, and a solo trumpet for the melody. My main goal at the start of the project was to replace the GPO trumpet I'd used in the original version of the recording with The Trumpet. I'd also intended to keep the project in Project5 for simplicity. The latter goal would not be achieved.

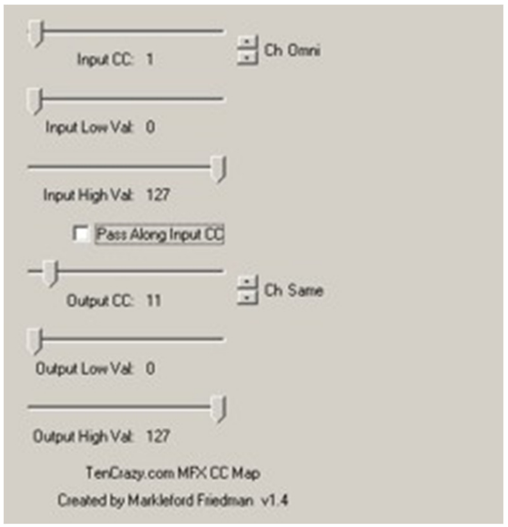
The big issue I encountered trying to keep the recording in Project5 was that the above-mentioned ASIO latency considerations of The Trumpet were too low for what Project5 (V2.5) could reliably play back in real-time on this particular project. It glitched noticeably in spots, and these same glitches carried over into the rendered audio. Project5 isn't as flexible as SONAR for dealing with performance considerations, so I manually moved the key project data across to SONAR to continue working on the recording there. After making minor adjustments to the backing tracks then freezing them to free up CPU power to deal with The Trumpet at low latency, it was on to tracking a new trumpet part with The Trumpet.

By this point, I'd been getting more used to playing The Trumpet in real time. It truly is a pleasurable instrument to play, and my initial attempts at recording my live performance were a significant step forward from what I'd previously achieved prior to using The Trumpet. At the same time, though, various individuals who listened to my early results commented on a lack of dynamics. One even went so far as to ask why I wasn't using the expression pedal (I most definitely was). A good deal of testing and analysis ensued. The short of it is I learned that, somewhere between my foot, my expression pedal (a Roland EV-5), my keyboards (mainly the Alesis QuadraSynth Plus Piano, but I also tried using the Roland Rhodes MK-80 at one point), and the computer-based side of my system (i.e. E-MU 1820M MIDI interface coming into SONAR 7 Producer Edition), the response I was getting out of the expression pedal was far from linear. The diagram below shows what I mean:



The CC#11 data at left shows the actual response of my expression pedal while attempting to make a linear sweep. The CC#11 data at right shows a hand-drawn linear curve over approximately the same range of time. Note how the extreme lower extremities of the expression pedal's sweep appear to completely cut off, then ramp up extremely steeply to a point around a data value of 30. After that, they change curve shape until around 60, then change again through around 100. After this, the response becomes reasonably linear up to peak value, but then stays at peak value for awhile more with no place to go at the higher extremities of the pedal's range. This behavior results in data values' being largely clustered around five bands of data, and very coarse dynamic differences result.

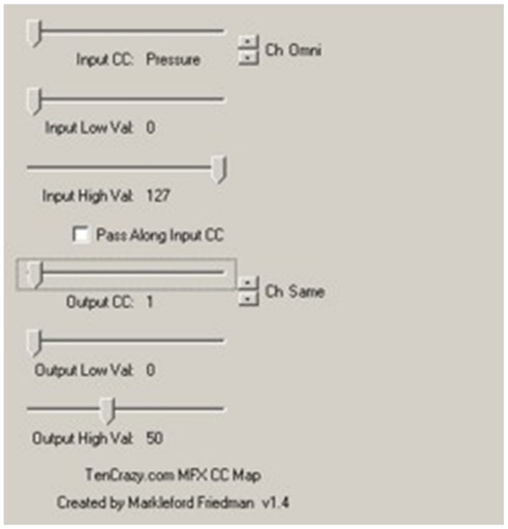
While taking this behavior into account allowed me to somewhat improve my results by paying more attention to expression pedal movement, short of obtaining equipment that behaved differently, there was a limit to how far I could go in this area, at least within the motor skills of my feet. However, I was not ready to resign myself to those results because I felt strongly that I could achieve better results with my existing hardware, if only I could shift my controller use to allow using a controller better suited for fine control to control dynamics.



Input CC: 1
Ch Omni
Input Low Val: 0
Input High Val: 127
Pass Along Input CC
Output CC: 11
Ch Same
Output Low Val: 0
Output High Val: 127
TenCrazy.com MFX CC Map
Created by Markleford Friedman v1.4

The Trumpet makes it easy to map another controller to CC#11, and my mod wheel was an excellent candidate for getting much more linear control. However, I also did not want to give up control over vibrato depth, and the Trumpet gave me no capabilities for mapping another controller to CC#1 or, for that matter remapping any other controllers beyond the remapping of an arbitrary controller to CC#11. However, in real life, we aren't limited to using products in isolation, so I looked around for a MIDI plug-in that could remap continuous controller information, and came across MFX CC Map from TenCrazy.com. I started out by remapping my mod wheel (CC#1) to send CC#11, making sure to have the plug-in remove the original CC#1 data from the MIDI event stream after converting it to CC#11. This got the mod wheel nicely controlling The Trumpet's dynamics.

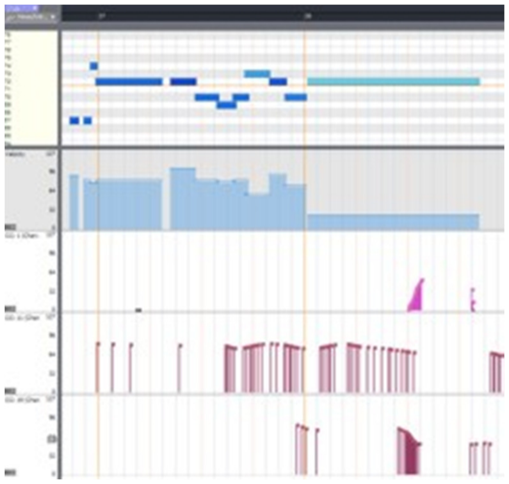
One thought I'd had at this point was to simply reverse the mod wheel and expression pedal use, using the mod wheel for dynamics and the pedal for vibrato depth. I'd had my expression pedal sending CC#11 at this point, so using it as is would just confuse things by interleaving the expression pedal data with the remapped data from the mod wheel. The solution was to configure my keyboard to send another CC value for pedal data. Once I'd done that, I could remap that CC value, which needed to be one that wasn't already being used elsewhere (I chose CC#2 since I don't have a breath controller), to CC#1 with a second instance of MFX CC Map placed after the first instance. (Note that placing it before the first instance would have made a mess of things by merging the remapped info from the pedal with the mod wheel data.) While that worked, I found it wasn't the most intuitive way to control vibrato rate. This was due partly to the same response curve issues I'd observed in using my expression pedal for dynamics, which made it tough to be subtle, and partly just to my natural inclination to want to use the pedal for volume. While I might have eventually gotten past the latter, I had another idea.



Input CC: Pressure
Ch Omni
Input Low Val: 0
Input High Val: 127
☐ Pass Along Input CC
Output CC: 1
Ch Same
Output Low Val: 0
Output High Val: 50
TenCrazy.com MFX CC Map
Created by Markleford Friedman v1.4

For me, the most intuitive control for triggering vibrato is aftertouch. While I'm a keyboard player, the notion of wiggling a finger on a stringed instrument to get vibrato simply feels sensible, and varying pressure rapidly on a finger as a note progresses feels similarly intuitive. MFX CC Map can also remap aftertouch data to a CC value, so sticking another instance of the plug-in after the one used to map CC#1 to CC#11, this time mapping aftertouch (which MFX CC Map calls "Pressure") to CC#1, would do the trick. However, my keyboard's aftertouch response is a bit dicey, almost always having some part of it that includes full-scale values (i.e. 127), with very coarse response in general. With The Trumpet, that would quickly get into exaggerated vibrato depth and shake territory. Luckily, MFX CC Map also provides the ability to limit the range of output data, so I limited the high data value to 50, and this resulted in a sensible range of data for vibrato depth.

While I could have stopped at this point, seeing as I now had the basic CC values needed to play The Trumpet live covered, it seemed a pity to just let my expression pedal sit idle. Additionally, while simply varying vibrato depth is where many instruments leave off in providing vibrato control, acoustic instrument players usually vary vibrato rate alongside vibrato depth. Thus, I used my keyboard controller's setup capabilities to map the expression pedal to CC#19 (vibrato rate on The Trumpet), and I was ready to go. In fact, the expression pedal proved quite a reasonable way to control vibrato rate. This was partly because precise control over the rate isn't mandatory for obtaining believable results and partly because my inclination toward using the pedal like a volume pedal coincidentally produced motions at least somewhat similar to how I would intentionally vary vibrato rate.



At this point I was off and running, with the finer control I needed over dynamics thanks to the combination of a more linear response in my keyboard's mod wheel and the finer motor skills of my fingers versus feet. The screen shot at right gives an idea of what the post-MFX filter (i.e. after applying the MFX in SONAR) remapped data looks like in the context of a phrase of the song. If you'd like to listen to the results, you can check out "Ave Maria (Schubert)" on my SoundClick page.

It's important to note that the choice of MIDI controls to use for controlling the various attributes of The Trumpet, or any other MIDI instrument for that matter, can be a highly personal preference. It can be affected by a number of variables. These include motor skills, various attributes of the specific equipment being used (e.g. response curve, physical resistance, etc.), and context (e.g. in some songs it might be more important to control growl level than vibrato, while the reverse would be true for other songs). If you add further considerations, such as the potential for using a breath controller, it is very easy to see where different users could have markedly different preferences in this area. Thus, one wish list item I have for a future edition of The Trumpet is the ability to easily map MIDI controls to the various controls available in The Trumpet. While it is not that difficult to do outside The Trumpet using third party plug-ins (and MFX CC Map is freeware/donationware), it would seem much more convenient if the user were to be able to do this in The Trumpet itself. This could be done similarly to how other controllers can be remapped to CC#11 in the current product.

It is also worth noting that, even in its stock configuration, The Trumpet is a pleasure to play. The point in reconfiguring things is to optimize the playability to achieve the greatest level of realism and expressiveness in light of one's limitations and preferences. It is also worth noting that your particular combination of expression pedal and other equipment, motor skills, and personal preferences may well make The Trumpet's default configuration the optimal one for you.

Closing Notes

The Trumpet may be the first product of this collaboration between Peter Siedlaczek and Giorgio Tommasini, but it goes well beyond the status of just being a solid offering. In fact, it raises the bar for acoustic instrument emulation yet again. If The Trumpet is representative of what we can expect from Sample Modeling in the future, I suspect I won't be the only one eagerly looking forward to their next offering. A few obvious candidates would seem to be tackling the trombone and saxophone families. (I think saxophones may be a bit akin to the Holy Grail of instrument modeling challenges.)

While the current product is not without a few issues, thankfully all of these have at least some form of workaround. Given how responsive Messrs. Tommasini and Siedlaczek have been in their user forums, I am confident that issues that do arise will at least be given due consideration, and likely addressed as rapidly as is practical.

The bottom line, though, is that anyone who needs a software-based emulation of a trumpet and/or related instruments that is as realistic as possible will definitely want to check out The Trumpet.

**Rick Paul is a songwriter living in Southern California. You can contact him at http://www.RickPaul.info.*