FFRend Help

Contents

Getting Started

Introduction

FFRend (Freeframe Renderer) is a renderer for Freeframe plugins. It allows you to chain any number of plugins together, automate their parameters using oscillators, and record the output to an AVI file. The input is a video (AVI/MPG), still image (BMP/JPG/GIF), or source plugin.

Most VJ softwares support Freeframe, and can record their output, so what makes FFRend different? FFRend is optimized for content generation, whereas VJ softwares are typically optimized for live performance. The key difference is that FFRend never drops frames, even when your project is too CPU-intensive to be rendered in real time.

It's also possible to perform with FFRend, though you may have to lower the resolution, simplify your plugin chain, or use a more powerful PC, in order to get an acceptable frame rate. All parameters and oscillator settings can be controlled via MIDI.

FFRend supports plugin authoring, which means you can export a FFRend project as a Freeframe plugin. The exported plugin is called a metaplugin, because it uses other plugins as components. A metaplugin can be used in any Freeframe-compatible host application, and behaves as if you were running the equivalent project in FFRend.

FFRend is free, open-source software for Windows 2000/XP. It includes comprehensive help, an installer, and a simple demo project. If you want to render complex effects at high resolution, using chains of automated Freeframe plugins, FFRend is your friend.

License

Requirements

Minimum:

Pentium III 833MHz
256MB RAM
Windows 2000 SP4
DirectX 8.1
640 x 480 / 16-bit color

Recommended:

AMD Athlon 64 4000+ San Diego 1GHz FSB
Foxconn NF4K8MC-EKRS Socket 939 Micro ATX
2GB OCZ 184-Pin DDR SDRAM
ATI Radeon X800XL 256MB GDDR3 PCI Express x16
Western Digital SE WD2000JD 200GB 7200 RPM Serial ATA150
ASPIRE X-QPACK-AL/420 MicroATX Case 420W Power Supply
Microsoft Explorer 1.0 optical trackball
Windows XP SP2
DirectX 9
1024 x 768 / 32-bit color

Installing

Installing for the first time:

FFRend is distributed as a .zip file. Unzip the distribution file, using WinZip or an equivalent program, and then double-click on FFRend.msi to launch the installer. The installer is about as simple as an installer can be: just keep hitting "Next."

Note that by default, FFRend is installed for the current user only. This means additional users must manually create desktop and start menu shortcuts to FFRend. If this is inconvenient, you may prefer to install FFRend for all users, using the following procedure:

1. If FFRend is already installed, uninstall it.
2. Unzip the distribution file into a folder, if you haven't already.
3. In that folder, double-click AllUsers.bat to launch the installer, and then proceed as usual.

Upgrading a previous installation:

1. Unzip the distribution file to a folder, if you haven't already.
2. In that folder, double-click Upgrade.bat to launch the installer.
3. In the installer, select the "Repair" option.

Note that double-clicking FFRend.msi won't work: you will get the message "Another version of this product is already installed".

Uninstalling

To uninstall FFRend, use Add/Remove Programs in the Control Panel, or double-click on FFRend.msi and select the "Remove" option.

Demo project

FFRend's binary distribution includes a simple demonstration project, called DemoBall.ffp. You'll find it in the same folder as the application (typically C:\Program Files\FFRend). DemoBall loads a Freeframe plugin (FFSrcPlug.dll, also included) which displays a bouncing ball. The plugin has a single parameter, the ball speed. This parameter is automated, using a ramp down waveform, so that the ball gradually slows down and then suddenly speeds up again.

Plugins

Plugin chains

A Freeframe plugin processes video frames. FFRend allows you to load multiple plugins at once. The plugins form a *chain*, i.e. each plugin takes its input from the previous plugin's output. By default, video frames flow through the chain from left to right. In this case, the order in which the plugins appear is significant, because it determines the order in which they process video; reordering them can have a dramatic effect on the output.

It's also possible to create an explicit routing that overrides the default left-to-right signal flow. This is essential if you're using multi-input plugins; see Patch Bay.

Input video

The plugin chain requires an *input*, i.e. a source of video frames to process. The input to the plugin chain comes from a video file (AVI/MPG), a still image (BMP/JPG/GIF), or a special type of Freeframe plugin, called a *source* plugin. Source plugins differ from effect plugins, in that they don't take any input, they only create output.

Note that FFRend needs AviSynth to play MPEG files. If you try to play an MPEG without AviSynth, FFRend displays an error message asking you to verify that AviSynth is correctly installed. AviSynth is free software; you can download it from http://avisynth.org.

To use an input video or image file, choose *File/Video/Open*, or drag the file from Windows Explorer and drop it onto FFRend's main window. To use a source plugin, load it into the plugin chain, in the same way as an effect plugin.

Note that you must have a video or image file open, or have a source plugin in your plugin chain, otherwise you won't get any output. If you have both a video/image *and* a source plugin, the source plugin wins: the video/image remains hidden unless you bypass (or delete) the source plugin.

Plugin slots

A *slot* is a container into which a Freeframe plugin can be loaded. Plugin slots are represented by tabs along the top of FFRend's main window. To insert a slot, use *Edit/Insert*, or the `Ins key`. To delete a slot, use *Edit/Delete*, or the `Del` key. Plugins can also be inserted and deleted using the plugin context menu, which is displayed when you right-click on a tab.

Only one plugin slot can be viewed at a time. This slot is referred to as the *selected plugin*. To select a slot, left-click on its tab.

A plugin slot can be cut or copied to the clipboard, and then pasted to a different project, or elsewhere within the same project, using the standard Edit commands, or the plugin context menu.

Loading plugins

Before a plugin can process frames, it must first be *loaded* into a slot. Once a plugin is loaded, it displays its parameters and allows you edit or automate them.

The simplest way to load a plugin is via *Plugin/Load* or `Ctrl+L`. The command displays a file open dialog; browse for the desired plugin, and press OK. The plugin is loaded into the selected slot. Note that if the selected slot already contained a plugin, the new plugin replaces it; any changes you made to the previous plugin's parameters are lost.

To load a plugin into a specific slot, you can also right-click on the slot's tab, and select *Load* from the plugin context menu.

Plugins can also be loaded via drag and drop from Windows Explorer. This method allows you to load multiple plugins at once. Note that with this method, existing plugins are *not* replaced; empty plugin slots are inserted automatically. If you drop on a slot tab, insertion occurs at that position. If you drop elsewhere, insertion occurs at the selected slot.

To unload a plugin, use *Plugin/Unload*, or choose *Unload* from the plugin context menu.

Reordering plugins

Unless you create an explicit routing, the order in which plugins appear in the plugin chain determines the order in which they process video. The plugins can be reordered at any time, without disrupting their states, via drag and drop. To move a plugin, left-click on its tab, and while holding down the left mouse button, drag the cursor horizontally. The cursor changes, to indicate that you're in drag mode. To drop the plugin, position the cursor over the desired tab and release the left mouse button. The plugin is moved to the new location, and the output is affected immediately.

Monitoring plugins

The Monitor control bar normally displays the output of the entire plugin chain (i.e. the same image as the output window), but it can be configured to display the output of a specific plugin instead. This feature lets you examine any intermediate stage in your plugin chain, in a nondestructive way, without affecting FFRend's output. It's mostly useful for debugging, but it can also be used for previewing.

Before you try to monitor anything, make sure the Monitor bar is visible. To show or hide the Monitor bar, use *View/Monitor* or `Shift+N`. To monitor a plugin, select it, and then use *Plugin/Monitor* or `F8`. A monitor icon is displayed in the plugin's tab, to remind you that the monitor source was changed. To return to monitoring FFRend's output, chose *Plugin/Monitor* again. You can also control the monitor source via the Plugin or Patch Bay context menus, or the Monitor bar's context menu.

Bypass and solo

A plugin can be temporarily *bypassed*, i.e. disabled so that it has no effect on the output. The bypass state is a toggle: to disable or reenable the selected plugin, use *Plugin/Bypass*, or choose *Bypass* from the plugin context menu.

It's also possible to *solo* a plugin, i.e. bypass all plugins except the selected plugin. This can be useful for isolating problems. Solo is also a toggle: to enter solo mode, use *Plugin/Solo*, or choose *Solo* from the

plugin context menu. Choose solo a second time to exit solo mode; this restores the bypass settings that were in effect when solo mode was entered.

Note that Bypass works in solo mode; this allows you to solo a plugin, and then add the other plugins back into the chain one at a time, by un-bypassing them.

Patch Bay

As of version 1.3, FFRend supports multi-input plugins. FFRend's default signal flow (linear from left to right) is adequate for single-input plugins, but for multi-input plugins, it's necessary to create explicit connections. This is typically done using the Patch Bay control bar. To show or hide the patch bay, use *View/Patch Bay* or `Shift+P`.

The patch bay includes a panel for each plugin, and each panel contains one or more source drop-lists, depending on how many inputs the plugin has. The drop-list begins with a default option, followed by a list of the plugins in your project, each of which is a potential input source. A check appears next to the currently connected source. To connect the input to a source, select the source in the drop-list. To break the connection and restore the default signal flow, select <default>.

Connections can also be made using the *Plugin/Input* popup menus, or the plugin context menu (displayed when you right-click on a plugin tab), but the patch bay is more convenient, mainly because it allows you to see all of your connections at once. The patch bay also lets you quickly change the order of the plugins, by dragging them within the patch bay. Each patch bay row also has an Enable checkbox; unchecking it bypasses the corresponding plugin.

The overall layout of connections between plugins is known as the *routing*. Routing doesn't have to be entirely explicit: it's fine to have a mix of explicit and default routing. Default routing requires less setup, and can be changed by simply reordering the plugins. One strategy is to only use explicit routing when necessary. Note that if you replace a plugin with a different one via *Plugin/Load*, FFRend preserves the existing routing as much as possible; this allows you to audition plugins without having to redo your connections.

It's possible to connect a plugin to itself, resulting in feedback. As with analog feedback, the output may saturate to white or black, or converge on a steady state, or may not get started without a seed, e.g. a video clip. Automating some of the effect parameters may help prevent a steady state from developing. More complex feedback can be generated by using longer signal loops involving more effects, and by using a mixer plugin to adjust the amount of feedback. For example, in the following setup, the signal flows from Chromium to Mixer, and then from Mixer to PanSpinZoom and back into Mixer; the more Mixer favors input B, the greater the amount of feedback.

Plugin	Input	Source	Comments
Chromium		<default>	video clip or whatever
PanSpinZoom		Mixer	feedback loop
Mixer	A	Chromium	routed over PanSpinZoom
	B	PanSpinZoom	more B = more feedback

Obtaining plugins

Freeframe plugins are available from various sources on the net, including the Freeframe web site, http://freeframe.sourceforge.net/. Paradoxically, not all Freeframe plugins are free, but some of the best ones are. The author recommends the following freeware plugins:

Pete Warden's plugins
: Pete's excellent package includes nearly fifty high-quality effects, including kaleidoscope, tile, fish eye, many color effects, and a variety of blurs. See http://www.petewarden.com/. Note that four of Pete's effects (Mixer, RadialBlur, SpiralBlur and TimeBlur) have a buffer overrun bug; FFRend detects these buggy plugins and refuses to load them. Unofficial patched versions are available at http://ffrend.sourceforge.net/download.html.

WhorldFF
: Whorld Freeframe is a source plugin version of the author's Whorld geometric visualizer. For general information about Whorld, see http://whorld.org. Note that unlike most plugins, WhorldFF depends on external patch files, which must reside in a specific location on your hard drive; for details, see http://whorld.org/WhorldFFReadMe.html.

Chris Korda's plugins
: This package includes a clip player, a boolean mixer, and a wave generator; to read more about these plugins and download them, see http://ffrend.sourceforge.net/download.html.

Big Fug's plugins
: An interesting collection of plugins from Alex May, featuring chroma and luma keying effects, Sobel, MotionMatte, Thermal, etc. See http://www.frame-runner.com/about/index.php.

Resolume
: Resolume's VJ software isn't free, but you can download a trial version which includes their Freeframe plugins. They have delays, luma effects, and some useful source plugins, e.g. color gradients, shape generators, etc. See http://www.resolume.com/. Note that a few of these plugins don't work with FFRend.

Parameters

Parameters

A Freeframe plugin can have any number of adjustable settings which affect its behavior; these settings are called *parameters*. Freeframe parameters are normalized, i.e. they have a fixed range (from 0 to 1). When a plugin is loaded, FFRend determines what parameters it has, and then automatically creates an appropriate user interface for displaying, editing or automating them.

Editing parameters

FFRend can display the parameters of only one plugin at a time, referred to as the selected plugin. Only the selected plugin's parameters can be edited; to edit the parameters of a different plugin, you must select it first, by left-clicking its tab.

For each of the selected plugin's parameters, FFRend displays a row in its main window. Each parameter row includes the following columns:

Parameter
: This is the name of the parameter, supplied by the plugin itself.

Slider
: This slider represents the parameter's current value. Moving the slider changes the parameter. The parameter can also be edited numerically, using the edit box to the right of the slider; see below. Note that the slider is also used to display and edit the parameter's modulation range.

Value
: This edit box also contains the parameter's current value. It allows the parameter to be edited numerically. The parameter can also be changed via the slider; see above.

The remaining columns are used for automating parameters. Note that a plugin may not have any parameters, in which case no parameter rows will be displayed when that plugin is selected.

Automating parameters

FFRend provides a way of making a parameter change by itself. This is referred to as *automating* the parameter. While a parameter is being automated, its slider and value are updated continuously, so that you can watch the parameter change.

An automated parameter is controlled by a low-frequency oscillator (LFO), using a technique known as *modulation*. Since each parameter has its own oscillator, multiple parameters can be modulated at once, each in a different way. An oscillator generates an endlessly repeating pattern, determined by its waveform, amplitude, and frequency. The waveform and frequency are specified as explained below. The amplitude defaults to the maximum range of the parameter (0 to 1), but it can also be limited, using a modulation range.

Enable
: This checkbox enables or disables modulation. Note that for modulation to actually occur, the frequency must also be non-zero; see below. The Enable checkbox allows you to temporarily stop an automation without losing your frequency setting. This is useful for manual overrides.

Waveform

This drop-list allows you to select an oscillator waveform, which determines the shape of the modulation. The available waveforms include Triangle, Sine, Ramp Up, Ramp Down, Square, Pulse, Random, and Random Ramp.

Frequency

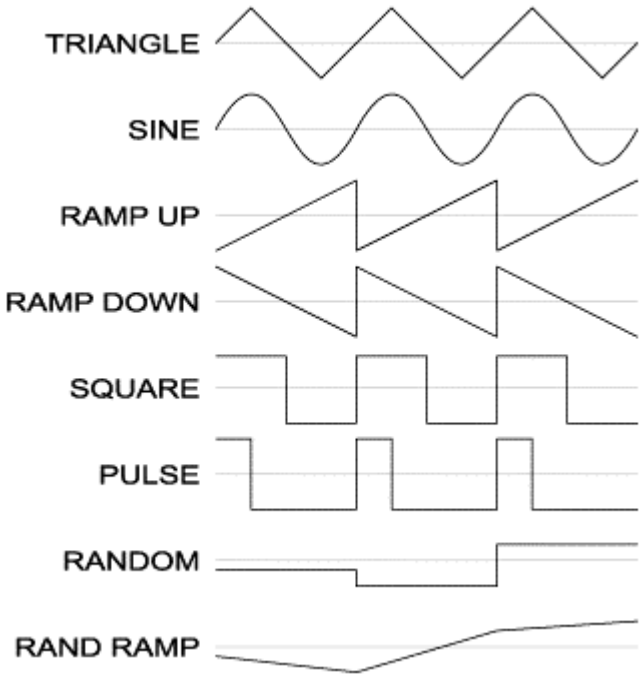
This edit control allows you to set the modulation frequency, in Hertz. If the frequency is zero, no modulation occurs. The upper limit is determined by the Nyquist theorem, and depends on the frame rate. Put simply, frequencies greater than half the frame rate won't work, and as the frequency approaches half the frame rate, the modulation becomes less accurate. Note that modulation frequency is affected by the master speed control.

Pulse Width

If the Pulse waveform is selected, this edit control allows you to set the pulse width; otherwise it has no effect. A pulse wave resembles a square wave, except that a square wave has a fixed 50/50 ratio of low to high, whereas a pulse wave has a variable low/high ratio, referred to as its *pulse width* or *duty cycle*. Pulse width ranges from 0 (all low) to 1 (all high); if it's 0.5, the pulse wave is identical to a square wave.

An automation can be resynchronized by left-clicking in the parameter's slider track. The slider thumb jumps to the cursor position, and automation continues from that point, though not necessarily in the same direction. To synchronize all of your automations at once, use *Edit/Sync Oscillators*.

Oscillator waveforms



Modulation ranges

A Freeframe parameter ranges from 0 to 1, and when a parameter is automated, its value normally traverses this entire range. Sometimes it's desirable to limit the value to a smaller range, called a *modulation range*. The range determines the *amplitude* of the modulation.

A parameter's slider has an unusually wide, open track, which is used to display the modulation range. The range is represented by a dark blue rectangle located within the slider track. During automation, the slider thumb (the part that moves) is confined to this rectangle.

A modulation range can be created in various ways. One method uses the slider's current position to set the range. Move the slider thumb to where the range should start, and choose *Edit/Modulation Range/Set Start*, or press `Ctrl+Home`. Now move the slider thumb to where the range should end, and choose *Edit/Modulation Range/Set End*, or press `Ctrl+End`. To remove an existing range, choose *Edit/Modulation Range/Remove*.

It's also possible to create or edit a modulation range numerically, using a dialog. To show the dialog, use *Edit/Modulation Range/Edit*. The dialog displays the current range, and allows you to edit it. To remove an existing range, set either the start or the end to −1.

The commands described above are also accessible via context menu. Note that to display the context menu, you must right-click on the parameter's name, NOT on its slider; right-clicking on the slider is reserved for additional shortcuts, which are explained below.

Shortcuts for creating a new range:

- To draw a new range, right-click in the slider track, at the point where the range should start, and drag the cursor horizontally, while keeping the right mouse button pressed. When the range has the desired length, release the right mouse button.
- If you right-click in the slider track without dragging, a range is drawn between the slider's current position and wherever you clicked.

Shortcuts for editing an existing range:

- To extend or trim an existing range, right-click near either end of it.
- To remove a range, right-click within the slider, but outside the slider's track.
- You can also drag an existing range: with the Ctrl key held down, right-click the range, and move the cursor horizontally, while keeping the right button and Ctrl held down. The cursor changes to a double-headed arrow to indicate the "drag range" state.

Projects

Projects

The entire plugin chain, including all plugin parameters and their automations, can be saved in a single document, called a *project*. To save your work as a project, use *File/Save* or *File/Save As*. To open an existing project, use *File/Open*. To start a new project, use *File/New*.

A project doesn't actually contain plugins; instead it contains *links* to plugins. Consequently, if plugins are moved or renamed, broken links can occur; see missing files. The project also includes a link to the input video if applicable. In general all settings are stored in the project, except for those found in the options dialog, which are stored in the registry.

Master speed

The Master toolbar allows you to affect the speed of all your automations at once. The toolbar contains a scaling percentage that's applied to your oscillator frequencies. If the scaling percentage is 100 (the default), the automations are unaffected; at 200 they're double speed, and at 50 they're half speed. To display or hide the toolbar, use *View/Master* or `Shift+M`. To change the master speed, set the desired percentage, via either the slider or the edit box.

If the CPU is overloaded while you're developing a project, the actual frame rate may be less than the ideal, which means your automations may seem slower than they actually are. You might not discover this problem until you record the project and watch the output video. If this situation occurs, you can use master speed to slow down your automations proportionally, instead of editing them one by one.

File Browser

The File Browser control bar allows you to navigate your files without using the File Open dialog or Windows Explorer. To show or hide the file browser, use *View/File Browser* or `Shift+F`. The browser has separate tabs for your Projects, Plugins and Clips, and remembers the current folder and view settings for each tab. To open a file, double-click its name, or select its name and press Enter, or drag the name onto the main window. The Plugins tab allows multiple selections, but the other two don't.

Note that if you drag plugins onto the plugin tabs, or onto the patch bay, they're inserted into the plugin chain at the drop point, i.e. it matters which plugin tab or patch bay row the cursor is over when you release the left mouse button. If you drag plugins elsewhere within the main window, they're inserted at the selected plugin.

The file browser supports five views: Icons, Small Icons, List, Details, and Thumbnails. To change the view, use the browser's context menu, which is displayed when you right-click anywhere within the browser *except* on a file name. Right-clicking on a file name brings up the file's context menu instead.

Which view you should use depends on how you dock the browser to the main window. In the default layout, it's docked vertically on the right, but if you prefer to dock the browser horizontally along the bottom, the List view lets you see more files at once. In the Details view, you can sort the files in various ways, using the Explorer-style clickable column headers. In the other views, use the context menu's Arrange option to sort the files. The Details view also supports resizing and reordering the columns.

The Thumbnails view is useful for selecting clips visually. Note that video clips will have thumbnails under XP but not under W2K. Thumbnails are created on demand whenever you browse a folder in Thumbnails view. By default, thumbnails are cached on disk, so they only need to be created once, though you can disable this feature via the Options dialog. Creating thumbnails can be CPU-intensive, so if you're performing live, create your thumbnails beforehand if possible.

FFRend's docking behavior is typical for Windows applications. If the bar is docked, double-clicking on its gripper (the two parallel lines) floats it. If the bar is floating, double-clicking on its caption docks it again. Bars can be resized while docked or floating. To move a bar to a new docking position, drag it by its gripper. To move a floating bar and *prevent* it from docking, drag it by its gripper while holding down the `Ctrl` key.

Missing Files

A project contains *links* to plugins. If a project's plugins are moved or renamed, broken links can occur. Broken links must be repaired before the project can be opened. If you try to open a project with broken links, FFRend displays the Missing Files dialog, which gives you the following options:

Search & Proceed
: Searches ALL folders of ALL hard disks for the missing plugins. This could take a long time, so only use this option when you have no idea where the missing plugins are. If you know where they are, even approximately, it's much faster to use the "Open Dialog" option instead. When the search is complete, the project is opened. Plugins that aren't found are removed from the project; see the warning below.

Proceed
: Opens the project immediately, regardless of broken links. The missing plugins are removed from the project; see the warning below.

Open Dialog
: Displays the Replace Files dialog, which lists the names of the missing plugins, and allows you to browse for them individually, or search for them in specific folders. If you want more control over the repair process, or if you know where the missing plugins are, even approximately, use this option instead of "Search & Proceed".

Cancel
: Cancels opening the project.

Warning: In all of the above cases except "Cancel", it's possible to open the project with plugins still missing. If you save the project in this situation, the missing plugins are *permanently deleted* from the project. To avoid doing this accidentally, save the project under a new name (using *File/Save As*) before proceeding.

Replace Files

If "Open Dialog" is selected in the Missing Files dialog, the Replace Files dialog is displayed. It lists the name and current status of each missing plugin, and gives you the following options:

Browse
: Allows broken links to be repaired individually. This is useful if a plugin has been renamed rather than moved. To repair a link, select it in the list (left-click its name), and press Browse. A file dialog is displayed. Locate the plugin, select it, and press Open. The link's status is updated from "Missing" to "Replaced".

Search Folder
: Allows you to search for missing plugins in a specific folder. The search is recursive, i.e. subfolders of the specified folder are also searched. Use this option when you know where the

missing plugins are, even approximately. When you press the button, a folder dialog is displayed; select a folder, and press "OK" to begin searching. As links are repaired, their status changes from "Missing" to "Replaced". The search can be repeated in different folders, until all the missing plugins have been found.

Search All

Searches for missing plugins in ALL folders of ALL hard disks. This can take a long time, so you should only do this if you have no idea where the missing plugins are.

OK

Ends the dialog and opens the project. Plugins that are still missing are removed from the project; see the warning below.

Cancel

Ends the dialog and cancels opening the project.

Warning: If plugins are still missing when you press "OK", they are removed from the project. If you save the project in this situation, the missing plugins are *permanently deleted* from the project. To avoid doing this accidentally, save the project under a new name (using *File/Save As*) before proceeding.

Recording

Recording

FFRend can record the output of your plugin chain to an AVI file. The AVI can be compressed or uncompressed. If it's uncompressed, there's no 2 GB limit; the AVI is limited is only by available disk space.

To create a recording, choose *View/Record* or press `Ctrl+R`. A file dialog is displayed; select a folder and filename for your recording, and press OK to continue. Next, the Record dialog is displayed. This allows you to specify the recording's length, frame size, and frame rate. Again, press OK to continue. Finally, the Video Compression dialog is displayed. This allows you to select and configure a video compressor. Press OK to begin recording.

You may find it convenient to start a recording while paused, to avoid missing the first few frames of your project. If you're trying to make an exact duplicate of a previous recording, and your project makes any use of randomness, you may also want to restart FFRend before each recording, so that the random seed is always freshly initialized.

Note that you have the option to defer the recording, instead of starting it immediately. To do this, check the Record dialog's "Don't run this job now; add it to job control instead" checkbox. You can queue any number of jobs, and then run them all at once; see Job Control.

Record dialog

Input frame size and frame rate
: These read-only fields display the frame size and frame rate of the current plugin chain. The recording can have a different frame size or frame rate; see below.

Output frame size
: This allows you to set the frame size of the recording. If the output frame size differs from the input frame size, the output frames are resized as needed. To use the input frame size, check "Use input frame size." Otherwise, select a standard frame size from the Frame Size drop-list, or for other frame sizes, select "Custom" and enter the width and height in pixels.

Output frame rate
: This allows you to set the frame rate of the recording. To use the input frame rate, check "Use input frame rate." Otherwise, enter a frame rate, in frames per second.

Output color depth
: This drop list allows you to select the color depth of the recording, in bits per pixel. The default is 24-bit (True Color).

Duration
: This allows you to specify the length of the recording, using one of the following options:

Custom	Lets you enter the desired length in the edit box. The Time/Frames radio buttons determine the input format. If Time is selected, enter a time in hh:mm:ss format; if Frames is selected, enter a frame count.
Unlimited	Records indefinitely, i.e. until you stop the recording manually.
Use AVI length	Sets the duration to the exact length of the input AVI file, and also automatically rewinds the AVI file at the start of the recording. Note that you must have an AVI file open, otherwise this option is disabled.

Don't run this job now; add it to job control instead
: Check this option if you want to defer the recording instead of starting it immediately. To manage deferred recordings, use the Job Control dialog.

Job Control

When you set up a recording, you can choose to *defer* it instead of starting it immediately. Deferred recordings are called *jobs*, and they're stored in a *job queue*. You can queue any number of jobs, and run them all at once while you're doing something else. This is known as *batch processing*.

The job queue is managed via the Job Control dialog; to show the dialog, use *File/Job Control*, or press `F4`. The dialog's layout should look very familiar to users of VirtualDub. The dialog contains a list, each row of which corresponds to a queued job. The columns are described below.

Name	The job name.	
Source	The source file, i.e. the name of the project from which the job was created.	
Dest	The destination file, i.e. the name of the output video file.	
Start	When the job started.	
End	When the job ended.	
Status	Waiting	The job is ready to run.
	Postponed	The job is not ready to run.
	In Progress	The job is currently running.
	Aborted	The job was canceled by the user.
	Failed	The job stopped due to an error.
	Done	The job completed successfully.

To start batch processing, press the Start button. To cancel processing, press the Abort button. To cancel the current job, but continue processing, press the Skip button. Jobs are processed in the order in which they appear in the queue, and only jobs with a status of Waiting are processed.

Editing the queue

The job queue can be edited at any time, even during processing. The only restriction is that you can't delete or change the status of a job while it's in progress. The editing commands are as follows:

To move a job
: Simply drag the job to the desired position, or select the job and use the Move Up/Down buttons.

To delete a job
: Select the job and press the Delete button, or right-click the job and choose *Delete* from the context menu.

To prevent a job from running
: Select the job, and press the Postpone button, or right-click the job and choose *Postpone* from the context menu. The status is changed to Postponed, unless it already was Postponed, in which case it's changed to Waiting.

To make a job ready to run
: Double-click the job. The status is changed to Waiting, unless it already was Waiting, in which case it's changed to Postponed. If the status was Failed, the error message is displayed, and the error is then cleared. To view the error *without* clearing it, right-click the job and choose *View Error* from the context menu.

To change the status of multiple jobs at once, use these commands from the Job Control dialog's *Edit* menu:

All Waiting => Postponed
: Postpones all jobs with Waiting status

All Postponed => Waiting
: Makes all postponed jobs ready to run

All Done => Waiting
: Makes all completed jobs ready to run again

All Failed => Postponed
: Makes all failed jobs ready to run again

To delete all completed jobs, Use *Edit/Delete done jobs*. To delete ALL jobs, use *Edit/Clear list*. The job queue can be saved to a file, or restored from a previously saved file, via *File/Save job list* and *File/Load job list*. To shutdown your computer after all jobs are completed, select *Options/Shutdown when finished*.

Job independence

Once a job is queued, it's completely independent of the project from which it was created. This is possible because the queued job contains a *copy* of the project data. Any subsequent changes made to the project file have *no effect* on the queued job. The project file can even be renamed or deleted. Note however that the job still depends on the project's plugins and input video. Be careful not to rename or delete plugins or clips referenced by a queued job, otherwise the job will fail.

A queued job also contains a snapshot of all application settings that could affect the job's behavior, including those found in the Options dialog and Record dialog. A job runs with the settings that were in effect when the job was queued, NOT the current settings. For example, if the frame size was 640 x 480 when a job was queued, that job *always* runs at 640 x 480, regardless of the current frame size.

Exporting bitmaps

Any frame of FFRend's output can be exported as a bitmap file. The bitmap's size will match the current plugin frame size. Only 24-bit color is supported. To export a bitmap, choose *File/Export* or press `Ctrl+E`, select a folder and filename for the bitmap, and press OK.

Export list

Sometimes you'll make a low-resolution recording, and then discover that certain frames would make nice still images, if only they had higher resolution. Why not just re-record the entire project at higher resolution? Because this might require a very large amount of disk space, especially since for making still images, you would prefer to use uncompressed video.

Instead of re-recording the entire project, you can export a list of specific frames. To do this, you must first create a frame list. This is simply a text file, containing the frame numbers of the still images you're interested in, as integers, one per line. Now set the desired resolution, enter pause mode, reload your project, and choose *File/Export List*. A file dialog is displayed; select your frame list, and press OK. FFRend regenerates the project, saving the specified frames as bitmaps, and throwing the rest away.

Note that certain plugins may not behave exactly the same way at different resolutions; you may need to compensate their parameters. Also note that if your project uses randomness, you may not be able to repeat the exact same sequence of frames unless you restart FFRend each time you load your project; see also random seed.

Pause

FFRend's processing can be paused at any time, using *Window/Pause*, or by pressing the `Space` bar. To resume processing, use *Window/Pause* or the `Space` bar again. While in pause mode, you can step forward frame by frame, via *Window/Step*, or `Shift+Space`.

MIDI

MIDI setup

Before you can control FFRend with MIDI, you must first select a MIDI device. Then you must *assign* MIDI messages to the things you want to control, using the MIDI Setup dialog. To show or hide the dialog, use *View/MIDI Setup* or `Shift+I`. MIDI assignments can be created via editing, or learned.

Things to which MIDI can be assigned are called *targets*. The most common MIDI targets are plugin parameters, or oscillator settings, such as modulation frequency. The list of available targets varies, depending on which plugins are loaded.

The MIDI Setup dialog contains tabbed pages of rows, much like the main window. There's a page for each plugin, and its rows correspond to the plugin's parameters. Each row lets you assign MIDI to a parameter's *properties*, which include the parameter itself, and its oscillator settings. Use the drop-list in the upper left corner of the dialog to select which property you're assigning to. If you just want to control parameters, not their oscillators, leave the drop-list set to "Parameter".

Put another way, the parameter pages form a three-dimensional matrix of MIDI targets, in which the axes are plugins, parameters, and properties. The pages correspond to plugins, the rows correspond to parameters, and the drop-list selects a property.

The dialog also includes two additional pages: the Plugin page, and the Misc page. The Plugin page is for properties that occur once per plugin, e.g. Bypass; the page contains a row for each plugin. The Misc page is for system-wide properties, e.g. Master Speed.

MIDI editing

MIDI assignments can be created and edited using the controls in each row of the MIDI setup dialog. The controls are described below. To make assignments this way, you must know which messages your device sends, and on which channels. MIDI assignments can also be learned, in which case you don't need to know these details.

Note that a given MIDI message can only be assigned to one target at a time. When you assign a message, if that message is already assigned to a different target, the previous target automatically (and silently) "loses" the message. Specifically, the previous target has its Event type set to "OFF".

Range
: This determines how much effect a MIDI message has on its target. The message is interpreted as an unsigned value, and *normalized*, so that it ranges from 0 to 1. The normalized value is multiplied by Range, and the result is the target's new value. The default range is 1.

  Range lets you make a trade-off between the *precision* and *magnitude* of a MIDI controller: making Range smaller allows finer adjustment, but reduces the controller's effect. Range can be negative, in which case the controller's effect is inverted.

Event
: This is the type of MIDI message assigned to the target. The possible values are as follows:

OFF	No message is assigned to the target.
CTRL	A continuous controller message is assigned to the target; the controller edit box specifies the controller number.
NOTE	A note is assigned to the target. In this case the controller edit box displays and understands MIDI note numbers, e.g. F#4. Pressing the note *toggles* the target, i.e. flips it back and forth between 0 and 1. Releasing the note has no effect. Notes are useful for controlling switches.
PITCH	The pitch bend message is assigned to the target. In this case the controller edit box is ignored.

Channel
: This is the channel on which the target's MIDI message is expected to arrive, ranging from 1 to 16.

Controller
: For controller messages, this is the controller number, from 0 to 127; be aware that controllers above 120 are normally reserved for channel mode messages. For note messages, this is the note number, from C0 to G10. For pitch bend messages, this value is ignored.

Value
: This is the actual data byte from the target's most recently received message, which can be helpful when debugging MIDI problems. It's displayed as a read-only value from 0 to 127.

Learn mode

Assigning MIDI messages via editing requires you to know which messages your device sends, and on which channels. FFRend can also "learn" the MIDI assignments, in which case you don't need to worry about those details. To use "learn" mode, do the following:

1. Check the "Learn" box at the top of the MIDI Setup dialog.
2. Select the target you want to control, by left-clicking its row. It's easiest to click on the row's name, but anywhere within the row will do. The row changes color, to green; this helps you keep track of which target is being learned.
3. On your MIDI device, twiddle the controller that you want to assign to that target. To assign a note to the target, press and release the note. The target's event, channel, and controller/note number should "snap" to the correct value. Be careful not to accidentally touch controllers or notes that you've already assigned to other targets, otherwise those assignments will have to be redone.
4. Repeat steps 2 and 3 for each target you want to control. To switch to a different page, left-click its tab. To change the property, use the drop-list in the upper-left corner of the dialog.
5. Uncheck the "Learn" box.

If you close the MIDI setup dialog with the "Learn" box checked, it will still be checked when you reopen the dialog, but no target will be selected. This helps prevent you from accidentally trashing your assignments.

Options

Plugin frame size

This option allows you to specify the size of the video frame that's passed along your plugin chain. This size effectively determines the rendering *resolution*. Use the drop list to choose a standard frame size; for other sizes, choose "Custom", and then enter the desired width and height in pixels.

The plugin frame size can differ from the input video frame size, in which case the input video is resized to fit the plugin frame. Note that this may cause aliasing (a type of distortion), which you can avoid by matching the plugin frame size to the input video whenever possible.

The plugin frame size can also differ from the output window size, e.g. you could render at 320 x 240 but view the output at 1024 x 768. The output image is generated post-rendering, so the size of the output window doesn't affect rendering quality.

Changing the plugin frame size reinitializes any currently loaded plugins. This may cause glitches in the output, so changing the frame size during a performance or a recording is not recommended.

As the resolution increases, more CPU time is required to render each frame. Large frame sizes may cause FFRend to fall behind and no longer be in real time. When you're recording a finished project, staying in real time usually isn't a concern. During development, however, staying in real time is preferable, because otherwise it's hard to judge the speed of your automations.

As a result, it can make sense to develop your projects at a low resolution, and then record them at a higher resolution. One problem with this approach is that your plugins may not behave exactly the same way at different resolutions. For example a plugin could have a parameter that's specified in pixels. Such a parameter would have to be compensated for changes in frame size.

While FFRend doesn't impose any upper limit on frame size, unfortunately the same can't be said for Freeframe plugins. Many plugins behave unexpectedly above a certain frame size. The limits vary, and can only be determined by trial and error. Most plugins can handle at least 1024 x 768, but above that, expect surprises.

Frame rate

This option specifies the ideal rate at which video frames should be processed and displayed, in frames per second (FPS). The actual frame rate can differ from the ideal. Both the ideal and actual frame rates are displayed in the status bar.

The actual frame rate is an approximation, and may fluctuate slightly around the ideal, but if it's *consistently* wrong, FFRend is not in real time. Note that FFRend NEVER drops frames, no matter how incorrect the actual frame rate is. Frame rate errors can be divided into two types: timer-related, and load-related.

Timer-related errors are linked to the way timing is done in Windows, which makes some frame rates easier to approximate than others. Timer-related errors are identified by the fact that they occur even when the CPU is idle. They can often be reduced by using a multimedia timer.

Load-related errors occur when the time required to process a frame exceeds the time available. The time available is a function of the frame rate, e.g. at 25 FPS, each frame must be processed in 1/25 of a second or less. If reducing the frame size, or simplifying your plugin chain helps, the problem is load-related. For more information, see performance.

Random seed

FFRend uses random numbers for some of its oscillator waveforms. This option allows you to control the seed value that's used to generate random numbers. You can specify a particular value, or you can use the system time, by checking the "Use time" checkbox, in which case no two runs of the program will generate the same random numbers.

FFRend's random numbers aren't truly random, in the way that tossing a coin is random; they're actually *pseudorandom*. Pseudorandom numbers form a deterministic sequence, which repeats eventually, but appears random enough to be useful. A pseudorandom sequence evolves from an initial value, called the random *seed*.

If the random seed is a fixed value, every time the program is run, it generates the exact same sequence of pseudorandom numbers. This can be good or bad, depending on the situation. If you want true randomness, the solution is to have the program use a different seed each time it runs. In practice this is accomplished by using the system time as the seed. This works because the system time is unlikely to ever be the same for any two runs of the program.

In some cases true randomness may not be what you want however. For example, suppose you create a project that does something you like, and you want it to do the exact same thing again in the future, e.g. so that you can make a recording of it. If your project makes any use of randomness, and you were using the system time as a random seed, you're out of luck. Your project will never do exactly the same thing again. This might be a good reason to *not* use the time as a seed.

Multimedia timer

The standard Win32 timer provided by Windows is unable to accurately achieve many common frame rates. For example under XP, if the frame rate is set to 25 FPS, you'll actually get 21.33 FPS, which is a considerable difference. This may not matter if you're recording at high resolution, because you're probably not in real time anyway, due to CPU overload. It's more likely to be a problem when you're previewing your work at low resolution. You want a correct frame rate when you're previewing, because otherwise you won't get an accurate sense of how fast your parameter automations are.

To make the frame rate more accurate, check the "Use multimedia timer" option. Note that this is a trade-off, because a multimedia timer uses significantly more CPU time.

Achievable frame rates		
	Win32 timer	**Multimedia timer**
2000	100, 50, 33.33, 25, 20, 16.67, etc.	100, 90.91, 83.33, 76.92, 71.43, 66.67, 62.5, 58.82, 55.56, 52.63, 50, 47.62, 45.45, 43.48, 41.67, 40, 38.46, 37.03, 35.71, 34.48, 33.33, 32.26, 31.25, 30.3, 29.41, 28.57, 27.78, 27.03, 26.32, 25.64, 25, 24.39, 23.80, etc.
XP	64, 32, 21.33, 16, 12.8, etc.	

A frame rate is normally expressed as a *frequency*, i.e. 25 FPS is equivalent to 25 Hz. A Windows timer is specified via its *period*, which is the inverse of the frequency. A timer has a property called *granularity*, which you can think of as its coarseness. Only timer periods that happen to be multiples of the granularity are supported. Other timer periods are rounded to the nearest multiple of the granularity. This won't make much difference if the granularity is fine compared to the period you want: e.g. a granularity of 10 ms might be acceptable if you're trying to wait a minute, or an hour. Unfortunately the granularity of a Win32 timer is coarse compared to most common frame rates, and worse still, it varies by Windows version.

Under Windows 2000, the timer granularity is 10 ms. If you ask for 25 FPS, you're in luck, because the period is 1/25 = 0.04 or 40 ms, which just happens to be a multiple of the timer granularity. On the other hand, if you ask for 30 FPS, the period is 1/30 = 0.033 or 33 ms, which is definitely not a multiple of 10 ms. The period therefore gets rounded up to the nearest multiple of 10, i.e. 40 ms, so you still get 25 FPS, even though you asked for 30 FPS. Not good.

You might wonder why 33 was rounded up, instead of down, since 33 is closer to 30 than 40. It turns out that timer periods are always rounded up in Windows 2000, whereas in XP they're rounded up or down, depending on which granularity is nearest.

Under XP, the timer granularity is 15.625 ms. If you ask for 25 FPS, the period (40 ms) gets rounded up to the nearest multiple of 15.625 ms, i.e. 46.875 ms (15.625 * 3), giving you a frame rate of 1/.046875 = 21.33 FPS. If you ask for 30 FPS, the period (33 ms) gets rounded down to 31.25 ms (15.625 * 2), giving you a frame rate of 1/.03125 = 32 FPS, which still isn't too helpful.

In summary, with a Win32 timer, 30 FPS is not achievable under Windows 2000, and neither 25 nor 30 FPS are achievable under XP. In contrast, a multimedia timer has one millisecond granularity under both 2000 and XP, which allows reasonable approximations of most common frame rates, though at the cost of increased overhead.

MIDI device

This option allows you to select a MIDI device. The list of devices varies, depending on what MIDI hardware is installed. If your PC doesn't have a MIDI device, the only option is "No MIDI input", which disables FFRend's MIDI support.

The default value is "No MIDI input", which means that in order to use MIDI, you must select a MIDI device at least once. You'll also need to assign specific MIDI messages to FFRend's functions, using the MIDI Setup dialog.

If you're using a USB MIDI interface, always connect it to the PC *before* launching FFRend. If you launch FFRend while the interface is disconnected, your device selection will be lost, and connecting the interface won't make it appear in the device list: you must first exit FFRend, and then connect the interface, re-launch FFRend, and reselect the device.

Undo levels

This option allows you to set the number of undo levels. For unlimited undo, check Unlimited; otherwise, uncheck it, and enter the desired number of levels in the edit box. To disable undo, set the number of levels to zero. By default, undo is unlimited. The undo history is cleared whenever you start a new project or opening an existing one. Note that unlimited undo can consume a significant amount of memory over time.

Save warning

If the current project has been modified, opening a different project normally causes the application to display a Save Changes warning dialog. In some cases this behavior may be undesirable, e.g. during a live performance. To suppress the warning, uncheck the Save Warning checkbox.

Global plugin

The global plugin option allows you to apply a post-process to all of your existing projects *without* editing the projects. This can be useful during a live performance, e.g. you might want to adjust the brightness/contrast globally, or display your projects through a matte. When you open an existing project, the global plugin is automatically appended to the plugin chain. The global plugin is NOT included when you save the project.

To set a global plugin, open the Options dialog, go to the Global plugin group, and press the Browse button. A file dialog is displayed. Select a plugin, and press OK. The selected plugin is added to the drop list. Press OK again to save your changes and close the Options dialog. Now open an existing project, and the global plugin should appear at the end of the plugin chain. Note that nothing happens until you open a project; selecting a global plugin has no effect on the current project.

The drop list contains the most recently used global plugins, so that you can select one of them quickly without browsing. To disable the global plugin, select <none> in the drop list. Again, note that you must open a project before global plugin changes take effect.

If you edit the global plugin's parameters, the edits persist until you select a different global plugin, or exit the application; the global plugin's state is not affected by opening projects. The same is true if you bypass the global plugin, or automate its parameters.

It is possible to apply multiple effects at once, even though there's only one global plugin. To do this, encapsulate the desired effects in a metaplugin, and then select the metaplugin as the global plugin.

Thumbnails

The File Browser supports a Thumbnails view for selecting clips visually. The thumbnails can be created in various sizes. To change the thumbnail size, select a different size in the "Thumbnail Size" drop list.

Thumbnails are created on demand whenever you browse a folder in Thumbnails view. By default, thumbnails are cached on disk, so they only need to be created once; to disable this feature, uncheck the "Cache thumbnails" checkbox.

FFRend stores thumbnails in the folder to which they apply, in a database file. The database files are named according to a specific convention, e.g. for 96 x 72 in 32-bit color, the database name is ckThumbs96x72x32.db.

Metaplugins

Metaplugins

FFRend supports plugin authoring, which means you can export a FFRend project as a Freeframe plugin. The exported plugin is called a *metaplugin*, because it uses other plugins as components. A metaplugin can be used in any Freeframe-compatible host application, and behaves as if you were running the equivalent project in FFRend.

A metaplugin preserves all project attributes, including parameter automations and signal routing. The only significant exception is MIDI assignments; these are ignored, to avoid interfering with the host's MIDI implementation.

A metaplugin can expose parameters to the host; such parameters are called metaparameters. A metaparameter can directly control a parameter in a component plugin, or it can control other targets, such as a modulator property, or a plugin's bypass switch.

A metaplugin normally contains links to its component plugins, but it's also possible to embed the components within the metaplugin's DLL. Embedding makes it easier to distribute the metaplugin, by avoiding dependence on external files. Since embedding could potentially encourage piracy, only *copyleft* plugins can be embedded.

It's possible to import a metaplugin back into FFRend as a project. This allows you to edit and re-export a metaplugin, even if you don't have the project file from which the metaplugin was originally created.

Metaplugins can be nested, i.e. a metaplugin can use other metaplugins. The depth of nesting is unlimited.

Exporting metaplugins

To export the current project as a metaplugin, choose *File/Metaplugin/Export*. Select a destination folder and filename, and press OK. The Metaplugin Properties dialog is displayed. Customize the properties as needed, and then press OK to export the metaplugin. The Metaplugin Properties dialog can also be shown via *File/Metaplugin/Properties*. The properties are described below.

Name
: This is the name of the plugin, which will be displayed in the host application. It's limited to 16 characters, and defaults to the metaplugin's file name.

Unique ID
: Theoretically every Freeframe plugin is supposed to have a unique four-character ID, but in practice this hasn't worked out too well, since there's no centralized coordination. As far as I know, only Resolume pays any attention to the ID. You should at least try to give your *own* metaplugins unique IDs, especially if you plan to distribute them.

Description
: This should contain a one-line description of what the metaplugin does. It defaults to "Metaplugin", but you can change the default, by pressing the Default button. FFRend doesn't limit the length, but most hosts probably will, so be reasonable. Don't put your name here; it belongs in the Author/License field (see below).

Author/License
: This should contain your name and copyright/copyleft notice, e.g. "Copyleft 2007 Chris Korda". It defaults to "Copyleft", but you can change the default, by pressing the Default button. Note that if you want to allow your metaplugin to be embedded within other metaplugins, your

Author/License information MUST contain the string "copyleft" (case doesn't matter). Using copyleft makes it easy for others to distribute metaplugins that contain your metaplugin, thereby encouraging *derived works*.

Version (Major/Minor)
: Note that this is the *plugin* version number, not the API version number. You can and should change this version number when you revise existing metaplugins.

Type (Effect/Source)
: Whether the metaplugin is an effect or a source is up to you. This means *you* have to decide whether your metaplugin processes its input frames (effect) or completely overwrites them (source). If your metaplugin pays *any* attention to its input, it should be considered an effect. Just because your metaplugin contains a source plugin, doesn't mean the metaplugin is automatically a source plugin too. For example you could have a metaplugin that mixes a source plugin with the input frames: that's still an effect. If the input frames *never* affect the metaplugin's behavior, the metaplugin is a source plugin, otherwise it's an effect.

Embed Plugins
: Check this box to embed the component plugins within the metaplugin DLL. Note that only *copyleft* plugins can be embedded. For details, see embedding plugins.

Edit Inputs
: This feature is not supported yet.

Defaults
: You can change the defaults for Unique ID, Description, and Author/License, using the Defaults button. The defaults are saved in the registry. Note that unless Author/License contains the string "copyleft" (case-insensitive), it will not be possible to embed your metaplugin within other metaplugins.

Note that FFRend automatically decides which video modes the metaplugin supports, as follows: the metaplugin will support a given video mode only if that mode is supported by ALL of the metaplugin's component plugins.

Importing metaplugins

It's possible to import a metaplugin back into FFRend as a project. This allows you to edit and re-export a metaplugin, even if you don't have the project file from which it was originally created. To import a metaplugin, use *File/Metaplugin/Import.* Select a filename, and press OK.

Note that if you only want to know which plugins a metaplugin uses, it's not necessary to import the metaplugin; it's easier to use the file browser. Find the metaplugin in the file browser's Plugins pane, right-click the metaplugin's name, and select Properties. The properties dialog lists the component plugins, and tells you whether they're linked or embedded.

Metaparameters

Like any other Freeframe plugin, a metaplugin can expose parameters to the host application; such parameters are called *metaparameters.* A metaparameter can directly control a parameter in a component plugin, or it can control other targets, such as a modulator property, a plugin's bypass switch, or a global property (e.g. Master Speed).

Metaparameters allow you to design your metaplugin's *interface*. Part of creating a metaplugin is deciding which things to expose. Since hosts often severely limit the number of parameters a Freeframe plugin can have, it's a good idea to order your metaparameters, so that the most essential ones come first. FFRend allows a Freeframe plugin to have unlimited parameters, so if you're only going to use the metaplugin in FFRend, you can expose as many metaparameters as you want, but you still may find it helpful to be selective.

Creating a metaparameter

To create a metaparameter, first show the Metaparams control bar (*View/Metaparams* or `Shift+E`), otherwise you won't be able to see what you're doing. Select the plugin, by right-clicking on its tab, and now in the main view (beneath the plugin tabs, where the plugin automations are), right-click on the NAME of the parameter. You'll see the following context menu:

> Mod Range >
> Metaparam >

Select Metaparam, and you'll see a popup menu like this:

> Parameter
> Mod Enable
> Mod Waveform
> Mod Frequency
> Mod PW
> Plugin Bypass

To expose the parameter itself, select Parameter. You can select one of its modulation properties instead, or to expose the plugin's bypass switch, select Plugin Bypass. The metaparameter magically appears as a row in the Metaparams control bar, with the slider and edit box already set to the target's current value. If you move the slider, or type a new value in the edit box, you'll see the target property change in sync.

Name and range

The metaparameter's name is generated automatically, but if you don't like the name, right-click it and select Properties, to show the Metaparameter Properties dialog. Now you can enter a custom name. You can also use this dialog to map the Freeframe parameter range [0..1] to whatever range you want. This is especially useful when the target is a modulation frequency. The range can also be inverted, e.g. to turn a "bypass" control into an "enable" control, use a range of [1..0] instead of [0..1].

Alternate method

Metaparameters can also be created via the Metaparameter Properties dialog, though the method described above is more convenient. This alternate method is the ONLY way to create a metaparameter for Master Speed. Right-click in the Metaparams control bar and select Insert. Now right-click the Metaparameter row you just created, and select Properties. In the properties dialog, go to the Plugin combo box, select Misc Properties, and you'll find Master Speed. You can use the combo boxes to select any target in FFRend.

Reordering

The order of the rows in the Metaparms control bar determines the order in which the metaparameters will appear in the host. The order can be changed by dragging rows. To drag a row, position the cursor over the metaparameter name, press and hold down the left mouse button, drag the row to the desired location, and release the left mouse button.

Metaplugin links

A metaplugin won't function unless all of its component plugins are available. There are two possible scenarios: the metaplugin can contain *links* to its components, or the plugins can be embedded within the metaplugin DLL. This topic deals with the first scenario, a linked metaplugin.

Since a link is an absolute path to a component DLL, links can break, e.g. if you move your plugins around, or rename folders. This is very likely to happen if you distribute a linked metaplugin, because the recipients probably won't have organized their hard drives in the same way as you.

When a linked metaplugin is loaded, it checks for broken links. If components aren't where they're supposed to be, the metaplugin searches for them, using a specific strategy. A linked metaplugin looks for its component plugins in the following places, in order:

1. The absolute path, as specified in project data.
2. The folder that the metaplugin DLL was loaded from.
3. The profile folder Application Data\FFRend\Plugins.
4. The METAFFREND_PATH environment variable path(s).

Option #1 is the most efficient. Option #2 is useful with hosts that require all Freeframe plugins to reside in a specific folder. Regarding option #3, note that Application Data\FFRend\Plugins is also used for unpacking embedded plugins. The environment variable is discussed below.

Error handling

If one or more components aren't found in any of these places, the metaplugin disables itself. A disabled metaplugin has no effect on its input, and doesn't expose any parameters to the host. The metaplugin also writes an error message to a log file indicating which plugins are missing. The log file (Application Data\FFRend\MetaFFRend.log) is a text file, and can be viewed with Notepad or any text editor.

Broken links can be repaired by importing the metaplugin as a project. The import displays the missing files dialog, which allows you to search your hard drive for the missing plugins. If the plugins are found, you can save the repaired links by re-exporting the metaplugin.

Environment variable

The advantage of the METAFFREND_PATH environment variable is that it allows you to direct the metaplugin to any folder(s) you like. So for example if you keep your plugins in a folder hierarchy separated by author, you can leave them right where they are, instead of copying them to App Data. Multiple paths MUST be separated by semicolons.

If you've never set an environment variable, consult your Windows documentation, but generally the following should work: On the desktop, right-click My Computer, select Properties, select the Advanced tab, click Environment Variables, and in User Variables, select New. For Variable Name, enter METAFFREND_PATH, and for Variable Value, enter the search path(s), separated by semicolons if there are more than one. Don't use quotes, and be careful to avoid leading or trailing spaces. Press OK a few times and you're done. You can use other variables as examples, but don't change them or stuff will break.

Embedding plugins

By default, a metaplugin contains links to its component plugins, but it's also possible to *embed* the components within the metaplugin DLL. To enable embedding, check the "Embed Plugins" checkbox in the Metaplugin Properties dialog.

The advantage of embedding is that the metaplugin doesn't depend on any external files. This is especially useful if the metaplugin is being distributed to other users, because it avoids the whole problem of broken links. The user doesn't have to worry about having all the necessary component plugins in the correct locations on their hard drive; they just load the metaplugin into their host application, and it works.

The disadvantage of embedding is that it can significantly increase the size of the metaplugin DLL. On the other hand, the plugins are stored in a compressed format, which helps keep the size down.

Embedding is only allowed if ALL of the metaplugin's components are *copyleft*. A plugin is considered copyleft if the string "copyleft" (case-insensitive) appears in its Author/License field. The only exception to this rule is for Pete Warden's plugins. Embedding is restricted to prevent inadvertent distribution of commercial plugins. Note that this restriction also applies to metaplugins used as components; see nesting metaplugins.

How it works

When you load an embedded metaplugin into a host for the first time, the metaplugin *unpacks* its component plugins into a special folder in your profile: Application Data\FFRend\Plugins. If a given component is already present in this folder, it isn't unpacked. Since components aren't deleted, a metaplugin only needs to unpack itself once; the next time you run it, all of its components will already be present. This means the slight delay associated with unpacking only occurs the first time you use the metaplugin.

A metaplugin may not even need to unpack itself the first time you use it, e.g. if you've previously used a different metaplugin that shares the same components. Note that an unpacked component plugin is an exact binary copy of the original; even the file times are preserved.

Nesting metaplugins

A metaplugin can use other metaplugins as components. This is possible because from the host's point of view, a metaplugin is simply a Freeframe plugin, like any other. No special procedure is required; nesting occurs naturally when you include one or more metaplugins in a project, and then export that project as a metaplugin.

A metaplugin can use a combination of metaplugins and ordinary plugins, e.g. you could make a metaplugin that uses a mixer plugin to automatically crossfade between two metaplugins.

There's no limit on how deeply metaplugins can be nested. Each level of nesting results in some additional overhead, but the amount is insignificant compared to the overhead of the ordinary plugins that are actually doing the work.

Nesting lends itself to an object-oriented approach, in which simpler metaplugins are used as building blocks in higher-level metaplugins. This encourages iterative composition, and distributed creation. For example, if I make a metaplugin that does something you like, and send it to you, you can add value to it, by using it inside one of your own metaplugins.

Nesting can also be used to create modulations more complex than FFRend's built-in oscillator waveforms. For example, suppose a metaplugin exposes a parameter's modulation frequency as a metaparameter. If the metaplugin is nested within another metaplugin, the top-level metaplugin can modulate the frequency. The same technique applies to pulse width.

Nested metaplugins can be embedded, but only if all the plugins at each level are *copyleft*, i.e. have the string "copyleft" in their Author/License information (see exporting metaplugins). For example, if I send you a copyleft embedded metaplugin, you can embed it inside one of your metaplugins; if your metaplugin is also copyleft, it can be embedded by someone else, and so forth. When the top-level metaplugin is loaded into a host, it unpacks itself recursively.

Loose Ends

Full screen

FFRend's output can be displayed full-screen in one of two ways: ordinary full-screen, and Exclusive. In ordinary full-screen mode, the output window fills the entire display, but FFRend's main window and toolbars remain visible. In Exclusive mode, the output window fills the entire display, and all other windows are hidden. The main advantage of Exclusive mode is that it eliminates tearing.

To enable or disable full-screen mode, use *Window/Full Screen* or `F11`. To enable or disable Exclusive mode, use *Window/Exclusive* or `Ctrl+F11`.

Note that while in Exclusive mode, you should avoid switching to a different application (e.g. via the Windows key, `Alt+Tab`, or `Ctrl+Esc`), otherwise FFRend will unexpectedly exit Exclusive mode. You should also avoid starting up the Task Manager via `Ctrl+Alt+Del`, as this may cause FFRend to crash.

Dual-monitor

FFRend supports a dual-monitor setup, in which its main window and output are displayed on two different monitors. In the usual case, the main window is on the primary monitor (the built-in display on laptops), and the output is on the secondary monitor, but the reverse also works.

To make FFRend use a secondary monitor, you must first extend your Windows desktop onto the secondary monitor. This is typically done using Control Panel/Display/Settings, or in some cases via software that came with your graphics card. Once you've extended the desktop, you can drag FFRend's output window onto the secondary monitor. Now when you enable either full screen or Exclusive mode, the output window fills the secondary monitor.

Note that while in Exclusive mode, you should avoid switching to a different application, otherwise FFRend will unexpectedly exit Exclusive mode. It helps to maximize FFRend's main window, as this prevents you from accidentally clicking on other application windows or the desktop.

Tearing

Tearing is a distracting visual artifact. It typically looks like horizontal lines running up and down the image. Tearing occurs if the display is refreshed while the CPU is in the middle of writing a new frame to the graphics hardware. At that moment, both the old and new frames are partially visible, and if they differ (as is likely), the eye perceives a discontinuity (a tear) in the image.

Tearing can be only prevented by using Exclusive mode; this allows DirectDraw to synchronize access to the graphics hardware. In normal windowed and full-screen modes, tearing is unavoidable.

Performance

FFRend attempts to maintain a constant frame rate, but if the CPU is overloaded, the frame rate will drop, sometimes drastically. CPU overload typically results from a large frame size, a long plugin chain, or especially CPU-intensive plugins. It isn't necessarily a concern when you're recording a finished project,

though it can cause FFRend's user interface to become unresponsive. It's more likely to be a problem when the perceived frame rate matters, e.g. during development, or during a live show.

The simplest way to improve performance is by reducing the frame size. You may find it convenient to develop your projects at a low resolution, and then increase the resolution for the final recording, but be aware that some plugins won't behave exactly the same way at different resolutions.

If you're creating complex effects at high resolution, it may not be possible to stay in real time, except by using a more powerful computer. FFRend is compute-bound, i.e. its performance is limited primarily by CPU power, rather than graphics power, memory size, or disk speed.

Frame-dropping

FFRend solves the frame-dropping problem which occurs in some VJ softwares. To understand why frame-dropping is a problem, you need a little background about Microsoft DirectShow.

DirectShow is a component of Windows that deals with playing video and audio. Windows VJ softwares often use DirectShow to display video. DirectShow is a "streaming" technology, which means it tries to keep the video in real time at any cost. If the video falls behind (e.g. due to CPU-intensive effects, high resolution, or limited disk bandwidth), DirectShow *drops* (i.e. omits) as many frames as necessary to catch up. The advantage of DirectShow is that the application doesn't have to worry about keeping its video and audio synchronized.

The problem is that DirectShow-based VJ softwares typically record the same output that's being displayed. This means that if dropped frames occur in the displayed output, the recording will also contain dropped frames. The result is that in certain situations--e.g. if you're using many CPU-intensive effects, or a high output resolution--it can be impossible to make a clean recording. This is the very frustrating problem that FFRend was designed to solve.

Since FFRend doesn't use DirectShow, it isn't restricted to a fixed frame rate. Instead, it allows the frame rate to vary as needed. If the CPU is overloaded, the frame rate slows down, but frames are NEVER dropped, either in the displayed output or in the recording. Of course not using DirectShow has a down side too: FFRend can't play MPEGs unless AviSynth is installed. Sometimes you can't have everything.

Shortcuts

Ctrl+C	Copy	Copy the selection and put it on the Clipboard
Ctrl+E	Export Bitmap	Export current frame as a bitmap
Shift+E	Show Metaparms	Show or hide the Metaparameters control bar
Shift+F	Show File Browser	Show or hide the File Browser control bar
Shift+I	Show MIDI Setup	Show or hide the MIDI Setup dialog
Ctrl+L	Load Plugin	Load the selected plugin
Shift+M	Show Master	Show or hide the Master control bar
Ctrl+N	New	Create a new project
Shift+N	Show Monitor	Show or hide the Monitor control bar
Ctrl+O	Open	Open an existing project
Shift+O	Show Options	Show or hide the Options dialog

Shift+P	Show Patch Bay	Show or hide the Patch Bay control bar
Ctrl+R	Record	Record the output to an AVI file
Ctrl+S	Save	Save the active project
Shift+U	Show Output	Show or hide the output window
Ctrl+V	Paste	Insert Clipboard contents
Delete	Delete Plugin	Delete the selected plugin
Shift+Delete	Cut	Cut the selection and put it on the Clipboard
Ctrl+End	Set Range End	Set end of modulation range from current value
Shift+End	Goto Range End	Move to end of modulation range
F1	Help	List Help topics
F10	Toggle Solo	Enable or disable solo mode
F11	Full Screen	Display the output full screen
Ctrl+F11	Exclusive	Display the output in Exclusive mode
F4	Job Control	Show or hide job control dialog
F8	Monitor Plugin	Monitor the selected plugin
F9	Bypass Plugin	Bypass the selected plugin
Ctrl+Home	Set Range Start	Set start of modulation range from current value
Shift+Home	Goto Range Start	Move to start of modulation range
Insert	Insert Plugin	Insert an empty plugin
Ctrl+Insert	Copy	Copy the selection and put it on the Clipboard
Shift+Insert	Paste	Insert Clipboard contents
Space	Pause	Pause the output
Shift+Space	Step	Single-step the output
Ctrl+X	Cut	Cut the selection and put it on the Clipboard
Ctrl+Y	Redo	Redo the previously undone action
Ctrl+Z	Undo	Undo the last action