SMEDGE

What's New in Smedge

Smedge 2010

Table of Contents

Smedge 2010 Update 2

New MachStudioPro Virtual Module

Smedge now supports rendering using MachStudioPro, version 1.4, on multiple machines at the same time. MachStudioPro is a GPU lighting and rendering system available from http://www.studiogpu.com.

New Distribution Modes: “Sample” and “Reverse”

Instead of just sending out frames in increasing order, Smedge can now send out in forward, reverse, or the new “Sample” mode. In Sample mode, a sampling of frames throughout the sequence is sent out first. This can give a better representation of the time a sequence will take, and lets users quickly spot check the entire sequence without waiting for the whole thing to finish. Sample mode is the new default mode, but it can be configured in the Configure Master dialog, and it can be overridden for a Job in the Advanced Info tab.

Smedge components now use the FileServer to share log files

Output and operational log files are now saved locally on each machine by default. They are made available to the system using the built-in file server system, and displayed using the SmedgeGui Output window. Not only are captured output logs saved this way, but operational logs from each Smedge component can also be viewed, including the History.log file from each engine.

Herald can now load and save actions to a file

You can create and save sets of actions and pass them around using a file. Herald can load and save the actions.

SmedgeGui and Herald now warn on closing

To ensure that you don't accidentally close Smedge while it's working, there is now a warning dialog on closing the SmedgeGui and the Herald. SmedgeGui also now gives you options for leaving the Master and Engine running when the GUI closes.

New Aegis component

Aegis shows buttons for immediate Engine control. It is a complete GUI Shell component, but actually does its work by interacting with the Engine command line shell component. It presents buttons for enabling the local engine, disabling it deferred (so any currently running work can finish) or disabling it immediately (stopping and re-queuing any currently running work).

Improved Conspectus Display

Conspectus now sorts the Engines alphabetically, and shows the enabled/disabled state of each engine using the font color.

New parameter $(CurrentProcessID)

Gets replaced with the numeric process ID number assigned by the OS for the currently running process.

Output Window automatically scrolls much smarter

Before, the output window would often scroll by itself, especially when lots of output was being added. Now, the window stays where you scroll it unless you have scrolled to the very end of the output, in which case it will follow (or tail) the output. It also now selects by lines only, making is faster to select and copy sections of output. It uses custom keyboard shortcuts and also includes a custom context menu for more operations.

Additional command line controls

SmedgeGui includes new controls to determine if it should start the Master and Engine components on start up. Use -StartMaster to control the Master and -StartEngine to control the Engine. If you do not supply a parameter, or if the parameter you supply evaluates to a true value, then the appropriate component will be started. If you supply a parameter that evaluates to false, then the component will not be started.

You can also now control the log cleanup parameter by command line. Use the -LogCleanup (period) (size) command line flag to control the log cleanup process. The period is the number of seconds to elapse between checks (defaults to 3600 seconds or 1 hour). The size is the maximum size in megabytes to allow logs to grow (defaults to 1).

Finally, you can set the amount of time that the Engine will keep local captured output files. By default, it will keep any captured output logs for 30 days. You can use the -DeleteOutputAfter (days) flag to SmedgeEngine to specify a different number of days for output to be stored locally. Note that this only applies to work logs stored in the local log folder. If you override the output folder for a Job or for an Engine, these folders are not automatically cleaned, and output there will persist indefinitely.

Clients can bind to a specific listening interface

You can specify which network interface client components (everything other than the Master) use to listen for communication from the Master. This is useful if you have multiple network interfaces installed on a machine and want to ensure that the Smedge communication only uses one specific interface. Use the Configure Connection dialog to configure this.

Canceled work was calling the successful finished event instead of the unsuccessful event

Canceled work is no longer considered “successful.” Instead, the unsuccessful event handler will be called after a canceled work unit.

SmedgeGui sorts by Job status correctly now

It was sorting all “finished” jobs as the same status. Now it correctly differentiates between different finished status codes, including if you manually cancel a Job or let it finish normally.

Submit Job window replace functionality was not working on event commands

Search and replace now works correctly on the event command strings.

Configure Engine dialog was not transferring the Engine enabled status correctly

If you had selected multiple Engines and tried to enable them all at the same time, the change is now transferred to all of the Engines correctly.

Maxwell image generation during merging is now optional

Instead, it generates the image only on the final merge, increasing performance and decreasing disk usage during operation.

Fixed possible problems with Nuke script translation

There were some conditions that could cause the Nuke script translation system to fail, leading to problems trying to render Nuke scripts on a different platform than was used to create the script.

Fixed possible problems with Max local MXP file generation

It was possible for the file to ignore some of the requested folders when generating the MXP file and using the copy locally system.

XSI Virtual Module detects filenames correctly and finds its executable

It should detect the image filenames more consistently now, especially with mental ray rendering. It will also try to find the correct executable automatically when first run. It will generally find the latest version, but you can still adjust the path to the executable manually as needed.

Jobs cannot access $(ImageFile)

It was not returning the correct value, making it impossible to use this in an event handler.

Engine Shell's disable true command was not working correctly

It would disable the engine and kill the work, but not exit correctly.

Deprecated buggy $(HideSubrange) system

When enabled, the old $(HideSubrange) system would cause the system to fail to re-queue failed and aborted frames, and the Job progress would always stay at 0 percent (even though work was being done and not repeated). The Hide Subrange system is now deprecated and disabled by default. A new parameter $(ActualSubRange) serves the same purpose, returning an empty string if the $(SceneNameFormat) has been set, but allowing normal operation that requires the real underlying sub-range to be known to work correctly.

If you have used the $(HideSubrange) functionality in any virtual modules, you should replace it using the $(ActualSubRange) parameter for the areas where you need the sub-range hidden, and expect that the $(SubRange) will always return the valid sub-range of the work unit, even for work where the $(SceneNameFormat) has been set. This way you can get the sub-range even for work that uses a scene file per frame.

Optimized performance, stability, and resiliance

Smedge memory and CPU usage should be vastly reduced on all platforms, increasing system performance and scalability. Several possible crash and hang points have been fixed, making this the most stable Smedge ever. And some possible transient error conditions are now handled better to avoid possible problems related to busy networks or file servers.

Fixed work being incorrectly marked as lost after an Engine disconnect

The Master would incorrectly mark any work that was running on the engine as “Restarted” even though the work had not been restarted, and it would be terminated, leaving it permanently unfinished, but considered as finished by the Master. Now it correctly recovers the work already on an Engine.

Fixed event handler functionality

Work Parameter Changed event now sets the parameter that changed before calling the event handler, and canceled work calls the Work Finished Unsuccessful event instead of the Successful event.

Improved Service/daemon installation

When you install the Master or Engine as a service or daemon on a machine, the GUI now also creates a machine option that disables the automatic starting of the component. This can reduce overhead at startup and possible issues with multiple instances. Update 1 also fixes a related bug on Windows with the system that ensures that only a single instance of a component is run on a machine. Together, these changes should simplify the process of installing services on a machine. It will also try to restore the original value for these options when you remove the service.

Engine Restrictions are displayed in the Engine list and InfoPanels

When an Engine is restricted, it shows in the Engine's status, including the number of available CPUs during the restricted period. It also uses new Engine list colors for Restricted and Restricted and Active Engines. The Engine InfoPanel also shows the restrictions that are set on an Engine.

Floating Job History window is now a frame

Like the output window, it is separated from the main GUI window on Windows and Linux (Macs already worked this way), and shows an icon in its title. Also fixed the inconsistent display of the icon in the output window.

Menu commands to browse User and Machine folders

Added more commands to find the User and Machine preference folders in the System > Smedge Files submenu. These commands browse to the root folder where each component app has its own folder to store preferences.

Default Creator String can be the User or the Machine name

There is a new Master configuration option that allows you to specify if you want the default “Creator” string to be the name of the machine instead of the name of the logged on user. You can also set it to use both, in a User@Machine format.

New Fryrender Module

Fryrender works similarly to how the Maxwell Light Simulator works, including allowing merging of renders of the same image from multiple machines, and the ability to stop and merge all work even before the requested number of iterations have been completed. It supports the Fryrender 1.0 syntax, and will support the modified syntax of Fry 1.5.

Maya Module finds the path to the fcheck executable by default

The Maya Module can now find the path to fcheck to use in the SmedgeGui options for viewing frames without having to configure it manually. It now also exports the MAYA_LOCATION environment variable so that fcheck works correctly on Macs. Also fixed a problem consistently finding the Maya Render executable on Windows.

Nuke Module can now translate paths inside of the Nuke script

The Nuke Module now includes a new parameter “TranslateScriptPaths.” This enables a system that copies the Nuke script locally to each Engine, then parses through the file to translate any paths it finds, using the Smedge path translation system. You can disable the system as an Engine option for Nuke Products, or for any given Job using the Advanced Info parameter.

Better output processing for mental ray Standalone Module

The mental ray Standalone Module handles processing output from the mental ray renderer better by ignoring the thread information at the beginning of each line of output. This allows it to more easily find image filename and error strings as reported by mental ray during the render process.

New “heartbeat” logging system to help diagnose problems if Smedge components fail

There is a new system that can periodically dump a log of where every thread of operation inside of a Smedge component application is currently running. This is important if you find that a Smedge component is gets hung up and ceases to respond correctly. To enable this system, add the -Heartbeat command line flag to the component you are starting. If you add this to SmedgeGui, and SmedgeGui is set to start the Engine and Master, those components will also inherit the flag. You can adjust the period of the logging using an optional number of seconds after the flag. If you do not add a number, it will log every 15 seconds. This system adds minimal overhead to Smedge processing: the snapshot itself is very quick and the data was already being maintained by the system, and the logging happens in its own thread to minimize the effects on any other thread. The debug build available in the API distribution logs by default, even if the -Heartbeat flag is not present, but the system can be disabled by setting a value of 0 to the heartbeat log period.

Product Limits were not being saved correctly

This bug was introduced as part of the redundant Master system. The Master now correctly saves the Product limits.

Fixed possible unresponsive Master

Certain failures in the Master's communication system could cause the Master process to be running but completely stop all communication. The only resolution was to stop and restart the process. Now the Master checks for this condition, and can try to restart the communication systems as needed. Note that if the conditions that caused the failure of the communication system itself is not addressed, the Master may still not work correctly. These conditions are usually hardware or OS related, and are quite uncommon and often temporary.

Fixed issues with Engine Restrictions not being correctly saved

The restrictions would be set in memory and saved to the Engine's options on disk, but if the Master was restarted, the Engine defaults would override any custom Engine restriction times you had set up. The Engine now correctly preserves customized engine restriction times. Also fixed an issue with a “One per Engine” Job not always being started correctly if an Engine was in a restricted period but still allowed work to be processed.

Fixed possible memory allocation system failure

Sometimes the internal memory allocation system reported an error about an invalid handle, and failed to get some basic required systems running correctly. Now the system uses a better system to deal with the allocator object handles which should eliminate this issue.

Version releases are now done by year

As you clearly notice from the name change, Smedge is now released annually, with the year as the version number instead of the complicated and confusing version system that had been used before. Smedge 2010 is the first version to make use of this system, and the plan is to release annual updates, making it clear how old your Smedge is and what the annual support contract gets you.

Automatic Redundant Master

This is a major change in the basic operation of Smedge. You no longer need to worry about specifying which machine is going to be the "Master" and ensuring that no other machine starts the SmedgeMaster process. Now, instead, the Master process can be started on as many machines as you like. The Master will keep the extra redundant "Mirror" Masters synchronized, so that any machine can take over as Master at any time, seamlessly. The GUI now simply (by default) tries to start both the Master and the Engine, making it as simple to get a machine online as just double clicking the big red S icon. Combined with the new ability to find the installed executable for many products, this vastly simplifies installation, especially on small networks. But because the components are still all separate processes, stability is not impacted, and you can still configure Smedge in a completely custom Master/Engine system just like before. This combines the best aspects of Smedge 2 and Smedge 3 to make Smedge 2010 the ultimate tool for managing renders.

See the new simplified Installation Guide for more information.

Engine Defaults can be saved on the Master

This allows you to override the factory defaults for engines the very first time they connect to the system. This is another incredible time saving tool, especially for configuring new nodes that may be added to a system. Before, the first time a node connected it would use the hard-coded defaults for the Engine settings and Product options, and you would have to manually configure that Engine at least one time. While Smedge provides many systems for simplifying the manual configuration of an Engine, it could still be annoying and could cause work failures on the machine until it was correctly configured.

Now, the first time the Engine connects, it gets the defaults from the Master, instead of using the hard coded defaults in the program files. This allows you to specify a system-wide default Engine configuration that is used immediately. The result is that you may not ever have to configure new Engine nodes, since the defaults will contain your required configuration. Adding a new node can become as simple as plugging it into the network and starting Smedge on it.

Conspectus

Conspectus is a new graphical shell that displays the name, CPU usage graph and memory usage graph for every Engine on your system. It provides a convenient place for you to quickly monitor your entire farm, to see which machines are working, and if they are working efficiently. The source code for Conspectus is also included in the API so you can see how to create a graphical UI shell component.

Smedge is bundled into a single application icon on OS-X

Installation on a Mac is now as simple as dragging the entire application bundle by its icon into whatever folder you want. It uses the Mac's "bundle" system to hide the complexity of the system. All of the components and modules are still there, just hidden by default. To see them, you can "View Package Contents" from the Finder, or use the Browse... commands in SmedgeGui.

Simplified Service/Daemon installation

The process of getting Smedge running as an automatic background process (known as a Service or a daemon) has been simplified and largely automated. Each platform has a set of scripts that automate the installation and removal of these kinds of processes, and the scripts can also be accessed directly from the SmedgeGui System menu.

Automatic detection of the latest installed executable path

When Smedge is first started, it can find the path to the executable for many of the products it supports automatically. This simplifies installation by eliminating the need to configure the path to the executable, or adding that path to the system PATH environment variable. Once it has been found or customized, it does not get changed automatically.

Products can be hidden in SmedgeGui

If you only use a handful of the Products that Smedge supports, you can hide the ones you don't use to streamline the interface. This is available in the SmedgeGui options dialog box, and is stored as a user preferences for the GUI. (You can use Smedge's intelligent INI loading to create a machine default, see the section on Smedge INI files in the *Administrator Manual* for more information.) The products will not display in any list in SmedgeGui, but the operation of the Engine and Master will not be affected. As before, you can always remove Modules or product definitions from the system if you want to completely exclude Products from the system.

Commands to explore the Log folder and Application and Utility folders

SmedgeGui now has commands in the System menu to browse the contents of the log folder, the application folder and the utilities folder on Windows and OS-X. This makes it easier to find the logs, the supporting shell components and the utility scripts and tools included in the Smedge distribution, even when the folders may be buried in deep folder hierarchies or hidden by the operating system.

SmedgeMaster can be bound to a specific network address

This makes it possible to configure the Master to use a specific network interface on a machine that may have multiple interfaces installed, giving you more customization options.

Master detection system isolates versions

Smedge 2010 uses the same ports as Smedge 3 used, but it ignores Master location requests from prior versions, so that old versions of Smedge that may be running on other machines don't connect and start causing problems because of communication differences.

Log files are now written in the User folder

This eliminates any permission issues on the log files that could stop Smedge components from starting properly.

Master and Engine can run without writable machine folder

Smedge tries to create the “machine” folder in a common shared location. Components used to fail to start if this folder could not be created (usually because of permission issues). Now all component processes can run properly even if this folder cannot be created or written to. Some data will not be saved between sessions in this case, but as long as the processes are running, they will run normally.

System command to dump all logs

This will dump a more detailed log with lots of debugging information on every connected node to help track down installation and configuration problems. The command is available in the SmedgeGui System menu, and will dump every log on every connected node into that node's log folder.

Configure Master command line Shell can now initiate a system-wide shutdown

Now you can request a system-wide shutdown programmatically using the Configure Master command line shell component.

SmedgeGui now has a toolbar

Common operations have toolbar buttons

SmedgeGui can hide Products

SmedgeGui can hide Products that you never use. The settings to do this are accessible in the SmedgeGui options. The hidden products are simply hidden from display for the current user. They are still available for other processing specifically for doing work by SmedgeEngine, and are not hidden for any other users by doing this.

Advanced Info tab on the Submit Job window shows if any settings have been customized

Like the other tabs, now the Advanced Info tab also shows a star in the tab name if any of the settings have been modified from their defaults. This makes it easier to notice when repeating jobs or opening a new window with the last job's settings, reducing the chances of submitting a job with incorrect settings.

Work runs at low priority by default

By default, Smedge now starts processes at a lower process priority. This helps keep the interface and other applications on a machine running smoothly while rendering. On workstations, the advantage is that the machine stays much more usable while rendering. On render boxes, the difference is minimal, since there are few other process running anyway. You can still configure the processes to run at either normal process priority, which may give a slight performance boost to the render process itself, or at idle priority, an even lower process priority.

Work that fails to start is correctly reported back to the Master in all conditions

There were certain types of work failures during the start of work processing that could result in the Engine aborting the work start request without reporting that it was doing so to the Master. This lead to “zombie” work units that could be hard to get rid of. Now it always reports back to the Master no matter what the result of the request to start the work is.

Herald can now respond to Jobs being paused or resumed

This gives Herald the ability to perform any action in response specifically to a Job being paused or resumed. All of the herald actions and all of the possible filters can be applied to these events. The events are only called if the Job's status is actually modified, so that pausing a Job that has already been paused doesn't cause the event to be triggered twice.

Herald can now respond to work assigned and work parameter changed events

The work assigned event is triggered when a work unit has been assigned to an Engine, but before the Engine has confirmed that it will start the work. The work parameter changed event is triggered when the work detects image file names and error text messages, or other changes to work unit data. Herald can now respond with any action to these events.

Interrupting work will try to interrupt pre-work event commands

If the work has not actually started yet because the Engine is still processing event commands associated with that work, interrupting the work will try to abort the event command.

ProcessJob Error detection can now look for error trigger strings anywhere in a line of output

Before it would only look for the error start string at the beginning of a line of output. Now, it can optionally look anywhere in the line of output. This can lead to more false positives if your error start strings are not specific enough, but can also simplify searching, and can allow for handling errors where the error start string is after some non-constant header data, as with mental ray, which outputs thread information before the error indication.

Smedge can delete files that fail the file size check after work

This is a new optional system that can be used to remove files that are detected but that fail the image file check tests. For example if a renderer creates a stub image at the beginning of the process, but fails before writing the final image, now the stub images can be removed. This is useful for products that support the concept of not overwriting existing frames, and can use this as part of their distribution system, such as After Effects.

New Parameter Command: CopyLocally

This new parameter command will attempt to copy a file from its current path to a local temporary folder on the Engine. It will return the temporary path to that file, even if the file or folder itself is not copied or does not yet even exist. If the source file does exist and either has not been copied, or is newer than the copy, it will be copied immediately upon the execution of the command. If the command is accessed as part of a parameter for a Job, all files copied will be deleted when the Job finishes. Otherwise the copied files will remain in the temp folder. The folder copied to is $TEMP/smedge3/LocalCopies/*job-id*.

New Job Events: WorkPostExecuteSuccessfulEvt and WorkPostExecuteUnsuccessfulEvt

Synchronously executed immediately after the WorkPostExecuteEvt, for successful and unsuccessful work units, respectively. This gives you a chance to have a synchronous event command executed only for successful or unsuccessful work, in addition to the universal synchronous post-work event. Note that these are not events in the API, but are simply commands that are called by the post-execute system in the Job itself.

Output server is displayed in the history

The log file and output server for work units are displayed in the Job history, so you can verify that things are working correctly.

Smedge makes system and work information available to the work environment

Smedge can set environment variables that the spawned processes can make use of.

SMEDGE_ENGINE	= The machine name
SMEDGE_ENGINE_ID	= The engine ID
SMEDGE_LOG_FOLDER	= the Smedge log folder for the engine
SMEDGE_PROGRAM_FILES	= The base of the Smedge 3 binary folder hierarchy
SMEDGE_COMMAND_LINE	= The command line generated for this work unit
SMEDGE_JOB_ID	= the ID of the parent job
SMEDGE_JOB_NAME	= the name of the parent job
SMEDGE_WORK_ID	= the ID of the work unit
SMEDGE_WORK_NAME	= the name of the work unit
SMEDGE_WORK_PARAMETERS	= A user customizable list of work parameters

New Restriction: Job Priority

When restricted, this stops non-administrator users from modifying the priority of any job, even their own. Essentially, this makes all jobs equal priority. Administrators can still change priorities for any job, allowing prioritization when needed, just out of the hands of the end users.

Job command line Shell improvements

The Job command line Shell component has had several improvements. It now supports a new option “Command” to programmatically send a Job command. It includes a new option “Status” to set a Job's status value to any customizable value. It also supports a new command “History” to get the entire work history for one or more Jobs. The “Preempt” command can now specify any custom status to use for stopping work.

The Submit Job window shows the Product type in the title bar

This makes it easier to remember what kind of Job you are configuring, especially for very similar product types.

All space characters are removed from captured output log files

This was a user requested feature to keep the generated filenames a little more computer friendly. Space characters are replaced with underscores.

All dynamic Products can override defaults and flags

All Modules that support dynamic products defined in INI files can now include the ability to have the defaults and the flags for Job parameters overridden by the customized Products. See the Dynamic Products section in the *Administrator Manual* for more information.

AfterEffects can start more than 2 instances of the AERender process

After Effects has several limitations based on its design, but one cause of crashing AERender processes on 64 bit Windows was due to how Smedge was monitoring the output from the renderer. Changing the process output capture process has resulted in allowing any number of AERender processes to be started simultaneously, at least when Smedge is running as a normal application. (When running as a service on Windows, there are still some limitations, because services don't have a GUI console by default, and AE is still primarily a GUI based application, even when command line rendering.)

AfterEffects now correctly detects the rendered image filenames on OS-X

OS-X After Effects reports the rendered image files using the Mac style path, with the mounted volume as the first entry, and using the colon character as a directory delimiter. Smedge uses Unix style paths on Mac (since it is Unix underneath) and so the files could not be properly detected to verify. Now it converts the Mac style path string into the correct Unix style path on the system to do validation and other image file name processing.

Max copies the scene and folder hierarchy locally

3D Studio Max has a confirmed issue when rendering with too many nodes at the same time that causes the scene file to fail to read correctly. To work around this Smedge uses the CopyLocally system to copy the scene file, and any files and folders inside the scene's parent folder to the local temporary folder for the Job. It then generates a local path file to help Max find the textures on the local folder hierarchy.

See the 3D Studio Max sections in the *Administrator Manual* for more information.

Maxwell Pyramid Merging and Maxwell version 2.0 support

The process of merging Maxwell's MXI files has been streamlined and improved to minimize network bandwidth, and optimally copy the files between machines. Maxwell Jobs can now be stopped before all requested repetitions may have been sent out, and still have the MXI files merged correctly. The new system handles failures more gracefully as well, and now supports the updated syntax for Maxwell version 2.0, as well as still supporting the previous versions, and correctly supports more of the extra command line parameters for both newer and older versions.

See the Maxwell section in the *Administrator Manual* for more information.

Maxwell percent done would be wrong for sequence rendering

It was calculating the wrong number of expected work units, which cause the percent done to calculate incorrectly.

mental ray for Maya can override the -mem flag

Allows you to customize the memory available to mental ray for caching data. You could add the flag manually before, but now it displays nicely in the list of custom parameters on the “Render Overrides” tab.

mental ray verbosity can be turned off completely for Maya rendering

Maya now supports setting the mental ray verbosity to none or errors only in order to reduce the copious output mental ray can produce when it's used in Maya rendering.

VRay for Maya

Maya module now includes support for the VRay for Maya plug in renderer.

Maya module ignores errors about renderLayerManager

Changes to Maya 2009 and later cause renders to sometimes report an error about the renderLayerManager. The error has no impact on rendering, so Smedge now ignores this error.

Maya 2010 works on OS-X

Maya 2010 on OS-X requires setting the MAYA_LOCATION environment variable. The Maya module now sets this automatically based on the path to the executable, so you don't have to set it before starting Smedge.

Nuke improvements

Nuke now is a customizable dynamic product, that can use an INI file to allow multiple versions to be used simultaneously. It also now includes support for specifying the write node to render, and -c, -f, -l, and -p command line flags, as well as a space that can be used to customize the render command line. It also now correctly detects the rendered filenames, and does a better job of finding error output that could sometimes have been lost on earlier versions of Smedge.

See the Nuke section in the *Administrator Manual* for more information.

Cinema 4D now includes output error detection

Errors reported by the C4D renderer in the process output are now detected and reported as errors using the standard Smedge output error system.

Sub parameters can now be set to be enquoted automatically

ParameterInfo::Parameters type parameters can include any other type of parameter as part of their definition. Now these “sub parameters” can be set to have quote marks added automatically if needed.

Messenger redesign

Many changes and improvements have been made to the Messenger system. There is now the ability to add a global handler that is called for every message received, giving you complete access to the entire message stream. The message allocation system has been integrated into the Message class itself, making it much simpler to create custom message types in Modules, without having to use the MessageFactory, which has been vastly simplified. Organization of the systems and optimization of the code has made the system more reliable and more efficient.

File Server System

Smedge now has a built in file server system allowing any node to share any arbitrary file to any other node. The “server” node must specifically share a file, programmatically. The system assigns a unique ID to this file, which can then be used by any other node to request that file to be transferred directly. The Master is only involved in translating the ID to the appropriate request. Once the request has been initiated, the transfer is directly between the server node and the client node. Failures, permission problems and unknown requests are all handled and cleaned up after.

Modules can send Messages from the Master

Two new API methods allow you to send messages from the Master to clients. These methods are only available on the Master, and will throw an exception if called from any other process:

```
void SSendMessageTo( UID clientID, MessageHandle message);
```

This method sends a message to a single client based on the client's ID. Engine's always use the same ID, based on the unique hardware ID, but every other client generates a random ID each time it starts up. A client can access its ID from its Messenger.

```
void SSendMessageToAll( MessageHandle message );
```

This method broadcasts a message to all clients.

Upgraded to Visual Studio 2008

Smedge is now built on Windows using Visual Studio 2008, and the API can be used to generate Modules and Shell components even using the freely available Visual Studio Express 2008. Part of this update includes .vsprop property sheet templates that can be used to correctly compile and

link with the Smedge libraries simply by including the property templates in your project. This vastly simplifies use of the Smedge API, and makes it available without having to purchase a compiler from Microsoft.

Upgraded to wxWidgets 2.8

This is a more up-to-date version of the cross-platform GUI library, which is faster and more stable on all platforms. Smedge uses wxWidgets built as a monolithic shared library. It does not use the Unicode build yet, because the other Smedge libraries are not yet fully Unicode capable (this is planned for Smedge 2011). On Linux, a few uncommon and unused dependencies have been removed.

Work Job types can be different than Parent Job types

This allows you to create customized Job subclasses for different types of work. The separate work types can have different parameters than the parent types, and will be hidden from the user when selecting products for creating Jobs.

Job::Copy can use any job types as source and target

Parameters in the source that are not available to the target will be ignored. Parameters in the target that are not available in the source will be untouched by the copy operation. The type is never copied, because that can only be set when a Job object is created anyway.

Job::IsReadyToSendWork() should be passed the Engine that is being checked

This system gives jobs a chance to delay sending work if the job has pending work, but that work is not ready to be sent yet. `Job::Dispatcher::IsReadyToSendWork()` takes both a Job and an Engine, and `Job::IsReadyToSendWork()` takes an Engine. That way, if the distributor cares about which engine is doing what, it has the information it needs to handle this.

CopyLocally system can be accessed programmatically

Besides using the parameter command “CopyLocally”, there is a new Job parameter “JobLocalFolder” that is substituted with the job's local folder. Additionally, you can get the Job local folder and use the Copy Locally system directly from the Job class API with the methods

```
Path Job::CopyLocally( Path source ) const;

Path Job::GetJobLocalFolder() const;
```

Linux and OS-X now log the command to start before searching for the executable

When starting a process on a Unix based OS, Smedge searches in the PATH and expands special characters in the name, and validates that the executable requested actually exists. Before, if the search for the executable failed, it would log the error about the missing executable without ever logging the command it was trying to start. This made the logs confusing and the problem hard to understand. Now, it logs the command it's trying to start before starting the search, so that it's more clear what is being searched for and what you need to do to fix the problem if the executable is not found.

Events no longer use modules and dynamic creation

The EventFactory has been removed completely, and the Event class functionality has been vastly simplified. All Events in the system are now simple stack objects that exist only for the duration of the event processing. The Event handler syntax did not change, because it was already using the an Event reference parameter, which works perfectly for stack based objects.

Centralized common version number and formatting

Updated to correctly show Smedge 2010 version system, there is now only a single version in the entire system, maintained in RLib.

Engine 'Type' is now 'Platform'

Type was confusing and “platform” is what it's called everywhere in the system, so now it's what it's called in the Engine

Alternate startup and cleanup signals

An additional method for being notified of the starting up or shutting down of a component is now available that uses the RLib Signal mechanism. The signals are triggered after all startup has finished and before any cleanup has commenced, so all library resources are guaranteed to be available. Using this system avoids having to use the LibStatic system, and can be used by any object, unlike the Application class virtual methods.

RLib supports Named Pipes

The “named pipe” form of inter-process communication has been added to the core cross-platform library, giving another way for Smedge components to communicate with other processes running on the system. Rlib's NamedPipes work like Unix domain sockets, even on Windows. They do not allow communication between separate machines, but can allow arbitrary communication between processes on a single machine that is more efficient than using the TCP/UDP sockets, and more customizable than the anonymous pipe implementation.

String integer conversions can convert using bases other than 10

You can now convert from string representation of hexadecimal or other integer base counting to a 32 bit integer or unsigned integer. String::Int and String::Uint take an optional base parameter. If not supplied, it will default to base 10. They do not look for leading characters in the string, so any leading 0 will be ignored as part of the conversion, and any leading 0x will cause the conversion to stop at the x, making the value 0. Note that the 64 bit conversions are not changed. String::Int64() currently will always assume base 10, and String::Uint64() will determine base from the value, if there is a leading 0 (for octal) or a leading (0x) for hexadecimal.

Other Optimizations

Messages use a dynamically sized buffer

Smedge no longer uses a fixed size message buffer to send messages. The fixed size buffer placed an upper limit on the size any individual message could be, which could cause problems for certain types of communication. This could be an issue, for example, when using a very long packet size to render many frames at once. When combined with the image filename detection system, the number of rendered image files could eventually hit this upper size limit, and cause communication problems.

Now there is no upper size limit. Any individual message can be as large as the working memory of the machine will allow. While there are still technical limits based on your hardware, OS and network, these limits are so large that you would experience problems with basic machine operation before you would hit the limit in Smedge communication, removing any practical limit to the size a packet of frames could be.

File::Copy speed optimized

Copying a file uses a larger buffer to optimize copy speed.

Process output window could fill very slowly

Especially as the amount of output grew longer. It now optimizes how it adds the output to the window to vastly increase performance.

Resolved IP addresses are now cached internally

To avoid repeatedly calling the system's name resolution system, the system now caches the resolved IP address and uses that, improving performance of TCP communication.

Engine connection overhead reduced

Engine options that are already at the default values are not broadcast through the system, reducing the Smedge network traffic overhead when Engines connect to the system.

Engine settings files are now group and world writable

Since the data is shared, it should be changeable by any user.

SmedgeGui font size on OS-X is smaller

It now looks much better and can fit a lot more data on the screen.

CheckFileSequence list flashes badly

It would flash the whole list every time it checked the files, which is every few seconds. Now only files that change are actually refreshed giving much better display performance.

Generic Script Jobs could crash the Master

Incorrect internal initialization could cause a memory access violation on the Master that could cause it to crash.

Possible hang in SmedgeMaster sending messages

A thread deadlock could cause the SmedgeMaster to hang sending messages, which could freeze the process under some circumstances.

Possible hang in SmedgeMaster on client close

A client closing its connection at the same time as the Master finished sending messages could cause a deadlock that would hang the Master.

The message stream could get corrupted

Under heavy load, a small amount of data could be accidentally discarded, which would corrupt the message to which the data belonged, and cause the messenger system to lose the message stream, causing communication failures. Though uncommon, the problem could often manifest as a consistent failure to maintain a connection with a wide variety of symptoms displayed as the cause of the failure, depending on the specific amount and type of data that was being processed when the problem occurred for a given user. This has been corrected by ensuring that the messenger only reads exactly the amount of data known to be required.

Master could lose a client connection without the client knowing

This could cause the client to become unresponsive to the system possibly until it was restarted even though the Master kept trying to force it to reconnect. Now the Master has a way to directly contact the client and force the reconnect when this situation occurs.

Linux daemon did not correctly change user and group when requested

The group was never changed and the user could cause problems writing to the logs which could cause the process to fail with permission issues. Now it correctly changes both group and user and avoids permission issues with logs.

Smedge files were created with incorrect permissions

Smedge files were being generated with 0600 permissions always, regardless of the umask setting for the process. Now the permissions are totally open, and correctly limited by the process umask.

Over average time kill was not working correctly

It was calculating the time delay in seconds instead of milliseconds, causing the delay before the kill to be 1,000 times longer than it should have been.

Some default Job values were not at the correct defaults

Some of the BoolOverride parameters are not at "Engine Default" by default, and the Idle timeout and minimum time values show 0 by default.

Job Files with invalid ID strings in the section could crash the reading application

The reading code would assume that the section heading could form a valid ID, and would crash if it could not for some reason. Now it will ignore any sections in the Job File where the ID cannot be determined from the section string.

Job Files did not restore the paused state of a job

All Jobs would be loaded in the queued state, even if the state was set to paused in the Job file. It now restores the paused state correctly.

Single instance limitation system was not correctly limiting to a single instance on Linux/OS-X

Partly due to permission issues and incorrect programming logic in the single instance detection system, it was possible to start multiple instances of the Master and Engine despite the system designed to stop that.

Smedge was only detecting up to 2GB RAM on Macs.

It was using an older 32 bit API to detect the installed RAM, which limited the result to 2GB. Now it uses the 64bit API to correctly get all of the installed RAM on the machine.

SmedgeGui could crash displaying Job History

History control could try to access invalid memory when detecting a work unit's current running time under some circumstances. Now it properly handles the situation and avoids the crash.

SmedgeGui could crash trying to select an Engine

The Engine InfoPanel could sometimes try to access an invalid memory area in a background thread as it tried to connect to the Engine for the status information stream. It now protects against this to avoid the crash.

SmedgeGui would freeze trying to set the path to the executable

Now it works correctly.

SmedgeGui could freeze when selecting a Maya scene file on Linux

The auto-detection system could hang the main thread if the Maya project could not be found.

SmedgeGui list controls could lose selection when any data was updated on OS-X

This could make it really hard to use, especially in heavy use.

Submit button on Submit Job window didn't set the Creator properly.

Submit and Close worked, but using Submit could cause the wrong creator to be assigned to the Job.

$(WaitForJobID) would only substitute a 0 or a 1

Now it substitutes the Job ID, or the NULL ID if there is no job ID to wait for.

Possible crash trying to remove whitespace at the end of a string

Could cause random crashes in Smedge components that used this functionality, generally the Engine and GUI.

Possible crash on Linux startup if `uname` process could not be started correctly

Fixed this possible crash

CPU usage was not working correctly on 64 bit Windows

Changed to a different CPU usage calculation system that should work on all Windows versions

SmedgeMaster was not using the correct virtual method to indicate a finished work unit

Now it correctly uses the virtual `Job::SetFinished` method.

SmedgeGui would try to connect to an Engine's status server twice

This could cause issues trying to display the status, and was a waste of bandwidth

Mac memory display was not consistent with the other platforms

Windows and Linux consider the virtual memory size to be the physical size plus the swap size. Mac considered the virtual memory size to be just the swap size. This is now changed to be consistent with the other platforms.

Workaround for Mac bug in pthread_cond_timedwait on 10.4

This API has a bug on OS-X 10.4 that can cause a thread to hang when polling the synchronization object. This system is used in the RLib thread synchronization classes, and could cause components to hang. Now it works around this bug by using a different OS-X specific method that does not have this bug.

SMTP header could be rejected by some mail servers

The SMTP connection now sends a fully qualified name during the connection, in order to satisfy picky mail servers. It also removes a double space in the header that could cause problems with some mail servers, even though the extra white space was valid according to the mail RFC specification.

Emails are sent with the wrong time

The time was being sent in UTC instead of local time, making it possible to look like emails were being sent from the future on some clients.

Reading a String from a Stream could incorrectly validate buffer

Some buffer underruns and overruns were not properly detected, which could cause random crashes.

Dequoting a Path could leave a trailing quote mark

Now it correctly trims both leading and trailing quotes in all circumstances.

RenderJob log message about clearing the image format strings could cause issues

The log message could try to access invalid memory that at best caused incorrect logging and at worst could lead to a crash caused by accessing invalid memory.

Possible crash at shutdown caused by Log cleanup

Uncommon thread cleanup ordering could cause a log to be left outstanding after the Log system had cleaned up after itself, which could cause an invalid memory access during the shutdown process and an abnormal termination.

Possible crash logging information about the ProcessingWorkMsg

Incorrect bounds checking could result in a memory access violation causing a crash. It was also missing some possible status strings for the work.

Stream::DisplayBuffer debugging method could crash with an access violation

It was not properly checking against a buffer overrun condition.

Seeking in a buffered stream was not working correctly

It's could cause read errors to seek before the current buffer position on a file, when it should have just rewound to the requested position and refilled the buffer.

Reading a string from a UDP connection could set the string length incorrectly

It would set the string length to the requested read length instead of the actual size of the string that was read.

Engine Information stream could get messed up

The reports of CPU and memory usage could sometimes get off the stream, and would cycle through very strange values.

Cutting the extension from a quoted path would not remove the quotes

This could result in the wrong extension being returned.