SMEDGE

What's New in Smedge

Smedge 2014

Table of Contents

Introduction

Welcome to Smedge 2014!

The latest version of Smedge provides a whole new level of performance and reliability, and gives you more control over your rendering workflow than ever before. Production tested on networks over 1,000 nodes, scalability is no problem.

Smedge 2014 does not communicate with prior versions of Smedge. We strongly recommend that you update your whole network at one time to avoid strange any problems with old versions left running.

Because some of the low-level API elements that make the system operate have been modified, we also recommend that you back up any old data before upgrading. While the data can be upgraded automatically, if you ever want to go back to the old version of Smedge, the upgraded data will not be successfully read by the old versions.

As always, if you have any questions about new features or our future development plans, we encourage you to contact us.

Thanks, and see you in Smedge!

System Changes

Smedge now works on up to 3 nodes with no license installed

In the spirit of the original Smedge from the late '90s, Smedge will now run on up to 3 computers with no license installed. If you have only a couple machines you want to hook together, or if you are a student or just learning about animation, or if you want to try it out without dealing with a demo license, just start it up! Gain the power of rendering on up to 3 machines without any worry about licenses, support, expiration dates or anything. Go ahead, take those three hundred dollars and spend them on something much more fun than render farm management software, like a nice pair of sunglasses or a trip to Six Flags!

Manual control over licensing

Continuing on the licensing changes, Smedge now gives you complete control over which nodes are licensed without having to start and stop processes directly on the machines. The GUI now includes commands in the Engine menu to force nodes to release licenses, or to request nodes to check out licenses if they are available. These administrator mode commands allow you to ensure that if you have more machines than you have licenses, that the machines you want to be rendering are. The Engine command line shell also includes these new commands, allowing you to integrate license control to other scripted operations as well.

Additionally, a license bug introduced in Smedge 2012 has been fixed so that the automatic license distribution now works correctly when a licensed node goes offline.

Dedicate multiple master machines

Previously, if you wanted to manually configure the connection to the Master machine, Smedge would only allow you to specify a single machine name. Now you can specify multiple machine names or addresses, allowing you to dedicate several redundant master machines to ensure optimum system availability. If you supply multiple names, the machine will try connecting to each one in order over and over until it finally connects successfully. You can also optionally add the special host name: `*` (a single asterisks character) which indicates to the client to use the automatic location system. Note that the automatic system can only find a Master on the same subnet as the local machine. You can also submit multiple Master hosts with the -Master common client command line flag.

Partial Job waiting

You can now configure the Job waiting system to wait for either the entire job to finish, as before, or only part of the job to finish. For example, if a render job that can run in parallel is waiting on an export job that has to run on a single machine, as soon as the export job reported a frame as finished, the render job could send off a work unit for that frame to render. This way your other machines can start rendering as soon as the one

translate machine has completed any portion of its work, better utilizing your machine time. Partial job waiting will work as long as the jobs have Distributors that are compatible with each other. If the distributors don't understand each other, the relationship between them will automatically fall back to regular full job waiting, where the job waiting won't send any work until the waited for job has completed or canceled all work.

Improved machine power saving features

The power saving features in Smedge have been improved to work more smoothly and efficiently than before. The system is much smarter about waking up only those machines that can actually be used for work, and only waking just enough machines for the work that is available. There are also several bugs related to the power saving system that have been addressed.

Set creator limits with regular expressions

With the low level String API changes come the addition of regular expressions, which can now be used with the Master's job limiting feature related to the creator. You can use regular expression syntax to define patterns for matching the creator string and determining the work limit allowed for a job based on that pattern. Smedge uses Perl style regular expression syntax as defined by the Boost library. You can find full syntax details here:

http://www.boost.org/doc/libs/1_54_0/libs/regex/doc/html/boost_regex/syntax/perl_syntax.html

Modernized development platforms and runtime libraries

As time moves forward, computer languages and compilers evolve and improve. To take advantage of some of those improvements, Smedge is now built on more modern operating systems and compilers than before. The downside is that some older generation hardware and software is no longer fully supported. On Windows, Smedge no longer supports Windows 2000. On Mac, it now requires at least OS-X 10.5 and no longer runs on PowerPC Macs. On Linux, we now develop on CentOS 6, so older kernel versions may not run any more.

Besides changing the platform requirements and runtime libraries, the biggest manifestation of this change is the new communication system. This new system is vastly more efficient and reliable than the old communication system, and has been proven to scale way beyond anything that was possible before. Development versions are currently running on networks of 850 nodes with no problems, and testing has shown it to be capable of scaling well into the multiple thousands of nodes with very little increase in the overhead of the system itself. Finally, this will also improve the reliability of the system even under the heaviest loads.

Note that some of these older platforms may be able to still support the modern code, but require special builds. If you have a need to run on such ancient hardware or operating systems, please contact support@uberware.net and we will see what we can do to accommodate you. Also, on Windows, the compiler requires a different run-time library than before. If you install from the MSI installer, this is handled for you automatically, but if you run it directly from the archive or copy the program files around, you will need to ensure that these run time libraries are installed on your system.

At this point, the Mac and Linux builds are 64 bit native applications, but the Windows build is still a 32 bit program. It will still run fine on 64 bit versions of Windows, and can control 64 bit rendering applications, however, its own run-time library will require the 32 bit (x86) version of the Microsoft Visual Studio redistributables. Again, you can contact support if you have any questions or run into any problems.

Improved 'Round-Robin' style distribution

The 'round-robin' distribution algorithm has been improved to better ensure that all jobs in your system get a more even amount of work time from the available Engines. You can enable round-robin distribution in the Configure Master dialog box.

Work was not always correctly interrupted

It was possible that an attempt to interrupt work would fail, leaving the work in a state that would not allow it to be interrupted. This has now been addressed so that if, for any reason, the request to interrupt the work does not succeed, the work can still be interrupted later.

Renders that generate huge volumes of output no longer slow down

The mechanism by which Smedge monitors the standard output and standard error streams for child processes has been improved so that the monitoring of the data send on these streams no longer impacts the performance of the child process itself. This means that renderers that generate copious amounts of text reporting about their status (for example mental ray) now run at full speed all the time, even as Smedge processes and monitors that output for error messages or rendered image filenames. You can now feel free to leave the verbosity of your renderers up high though of course if the verbosity of the output slows the rendering process itself, that is out of the control of Smedge, so the options to decrease or disable the renderer verbosity still exists for those renderers that support such configuration.

Correctly handles child process output streams that don't end with a carriage return

Related to the above, if a child process sends a message on one of the output streams that doesn't happen to end with a carriage return (new line or line feed), that no longer causes problems for Smedge monitoring of the output. Before that could lead to a deadlock in the child process and a halt to rendering.

Engine Pool order would get changed with more than 9 pools

Because the sorting of INI file data is handled using alphabetical ordering, the strictly numerical sorted differently than expected. The system now uses a padded number to ensure that you can correctly order as many pools as you need.

Font synchronization did not always work correctly on Windows

Windows doesn't handle fonts terribly well, so the font synchronization feature of Smedge has been modified to deal with the operating system shortcomings better. The problems mostly manifested when trying to synchronize a large number of fonts at the same time, and were caused during the font removal stage. In order to avoid the problem, Smedge no longer tries to remove the fonts it synchronizes. Note that this can lead to a large number of fonts on the rendering systems, so you may want to monitor the font usage of the machines and periodically clean out unused fonts yourself if you find that it is impacting performance on the nodes. We are still searching for a better approach but for now this should improve the stability and performance of the system.

System bug fixes

Some of the other general bug fixes for the system operation:

- Global stagger start was not always correctly respected
- Linux daemon status scripts show the correct status for the daemon processes
- CentOS was not displaying the correct operating system information
- Fixed strange behavior when machines shutdown or reboot
- Fixed several possible crashes, hangs, infinite loops and memory leaks

New list and history look

The main window GUI interface has been revamped for drastically improved look and performance. The view lists, the history list and the output window are all now double buffered to eliminate all flickering and have vastly improved display performance. The overall look has been improved to make it easier to see items in the lists, find the sort column and respond more quickly to user input. The colors are customized to fit in better with the look of the effects and animation software packages that Smedge supports, and progress is now shown with a progress bar, since everyone loves progress bars.

The history performance is nearly instantaneous, and now includes better statistical analysis of the overall job, as well as of individual runs for the work units. The look between Windows, Linux and Mac has been made totally consistent. The individual elements of the work history now display times that are more practical. And the operation of displaying the output associated with any given run works more consistently and reliably. Automatic scrolling also works better now and works on all of the lists as well as the output window.

Job Graph

There is now a job time graph display available in the GUI. This graph is built from the new history API data, and can show you the total or finished time for all work, along with the shortest, longest, and average times. The graph also uses colors to help show differences in job data and to differentiate active work from completed work (when in the total time display mode). You can see the graph by selecting a job or a work and looking at the appropriate tab in the Info Pane. You can also open a graph in a new window from the Job menu, the toolbar button, or from new menu commands in the History display for that job.

Automatic Engine Mode

The GUI, by default, starts the Master and Engine components with it so make the machine able to do work and able to control the system. However, all of these components may be excessive for a machine that is just to be used as a rendering engine only. While you always have the manual systems for running just the Engine component to improve performance by reducing Smedge overhead, the new automatic system can do it for you with the GUI.

After a certain amount of idle time (time during which no operations are performed in the GUI itself), it will shut down the GUI communication system and leave only the Engine component running and connected on the system. This greatly reduces the Smedge overhead, as the engine component only requires updates for events related to work running on that engine, not for the entire network. The additional memory and CPU performance are then available to your actual render processes, and Smedge performance is improved across the whole network.

The amount of idle time is customizable at both the system level, in the Administrator Options dialog box, and at the local level in the SmedgeGui options. In both cases, you can disable the system altogether, or set the idle timeout to a number of minutes of your choosing. You can also enter Engine Mode manually through the command in the System menu. If the GUI shuts down in Engine mode, then it will restart in Engine mode as well. Finally, if you run Smedge on a machine with a name that sounds like it is a render node (e.g. a machine called “render-010” or “blade5f40” or “node001”, then Smedge will automatically assume that you will want to run in Engine mode the first time that it is started, though of course you can still manually set it to normal operation.

Custom view presets

You can save a set of customized views as a view preset. The Customize Views tab includes new controls to create or select a preset collection of view tabs that you have saved. Selecting a preset will remove all current views and rebuild the views as defined in the preset. You can also now specify a default preset collection of views that is applied to all GUIs by default. After creating the preset in the Customize Views tab, you can select it from the Administrator Options dialog box.

Job list shows Job failure count

The Job list control now has another possible column that shows the count of failed work units for the job. The jobs are also colored differently if they have had failures, making it easier to find jobs that are having or have had problems. The error column is one of the default columns for the Job list.

Immediate command execution improvements

The window for immediate command execution is now a modeless floating window that indicates the engines on which the command will be executed. After executing the command, the window remains open, in case you want to execute the command again. Additionally, the commands you execute are now correctly remembered between sessions, so you have access to the history of commands that you have run to easily find and run them again. And since it's modeless, you can open multiple instances of this window with the same or different Engines selected, making it easier to run commands on larger numbers of heterogeneous machines. Windows Engines now also correctly execute batch files.

Engine power saving settings were not correctly transferred to all selected Engines

There was a bug in the Configure Engine window that did not correctly transfer the power settings when you were changing those settings on multiple Engines at the same time. This is now resolved, so you can reliably set up your Engines more efficiently.

Output windows were not placed correctly

The output window, used for showing the output from renders, as well as the history files and dispatch reports, was incorrectly placing itself, and would eventually start placing itself near the edge of the screen, even when no windows were open. This is now fixed, so the windows now open more reasonably, and there was also a memory leak with extra instances of these window remaining open but hidden forever that is now fixed.

GUI correctly shuts down locally started system component processes

Previously, if the GUI was set not to shut down the Master and Engine when it closed, those components would then continue to run until the user terminated them manually or logged out. Now, if you request to shut down the components, the GUI correctly tires to send the request, even if the component wss not started by this instance of the GUI.

New common command line options:

`-LogWithPID`	Add this flag to have the Smedge log files saved in a sub-folder with the process ID of the component process. This is useful when you have a command that may be executed several times on the same machine, even simultaneously (for example as event handler commands) to ensure that each instance of the process can independently save its log without interfering with any other instance of the process. This makes it easier to find and fix problems with these commands. Note that these logs are not cleaned up, so you should ensure that you clean up the file eventually to avoid consuming excessive amounts of disk space over time.
`-SendLogDump` *name|pid*	Use this to request a specific component process running on a machine to initiate a log dump. You can specify the component by the name supplied to that component with the `-ListenForShutdown` command line flag, or by its process ID number (as reported by the operating system). The component you are using to send the request does not have to be the same component you are requesting to dump logs from, and all components, even the graphical ones, will respond to the request by dumping logs. The process you start with this command line option will immediately terminate after sending the dump request to the other process (regardless of whether the other component is running or receives the request).
`-SendShutdown` *name|pid*	Use this to request a specific component process running on a machine to terminate itself. You can specify the component by the name supplied to that component with the `-ListenForShutdown` command line flag, or by its process ID number (as reported by the operating system). The component you are using to send the request does not have to be the same component you are requesting to terminate, and all components, even the graphical ones, will respond to the request by terminating. The process you start with this command line option will immediately terminate after sending the dump request to the other process (regardless of whether the other component is running or receives the request).

Job command line shell History display improved

With the big changes in the Job History API, the Job command line shell can now display a lot more information about the history of a Job, including the Job statistics (average work time, total elapsed time, etc...) that are in the GUI history list. The default Job history formatting now emulates the look of the GUI with a full breakdown for each run of each work unit, and specific details about the history elements for that run, organized into sections. The new history display will also give you the output log path and the ID for downloading the output log for each run with the Smedge file server system.

You can still get the old style raw dump of history elements as well, with a new command line option -Dump added after the History command. This dump of the raw elements is as before, except that the elements will be sorted by the work unit and sub-sorted by the run, and now they also include the ID of the work unit for each run. Note that it will always show the header information for the raw dump now, as that command line option has been removed.

Engine command line shell can use regular expressions

You can now use a regular expression syntax with the Engine command line shell to make Engine modifications. Use the -regex flag to start your expressions, and then have one or more expressions that will be matched against the engine name, ID and note.

Fixed crash in Job shell getting product info

The Job command line shell was crashing when trying to display the product information from the installed modules because it was trying to apply job filters from the features added in Smedge 2012 to filter the list of jobs.

Support for the Indigo renderer

Added support for the Indigo renderer (a repeat + merge type renderer)

Support for the Thea renderer

Added support for the Thea renderer (a repeat + merge type renderer)

Maya module includes a product specification for the Arnold for Maya plugin

There are new product specifications in the Maya.ini file for the Arnold for Maya plugin renderer. The two added products support rendering with Maya for Arnold with complete command line customization for both sequence rendering and single frame rendering. Note that you can also use the default

Improvements to After Effects submit script

The script that integrates with After Effects to submit jobs directly from inside the AE interface can now add extra command line parameters to the Submit component itself. You can use this if you need to customize how jobs are submitted without having to modify the script file itself. Some other small bugs in the script have been addressed as well.

Ability to disable Nuke verbose output

The Nuke module now has an advanced Job parameter and Engine option to disable the Nuke verbose output. You can configure this as an Engine option and override the Engine's default for specific Jobs.

Modo module supports render passes

The Modo module now can support rendering specific render passes. A new Job parameter allows you to specify the passes to render and the module will send the appropriate command to Modo to make it work correctly.

Generic Script jobs with no range correctly update when Engines are added or removed

Generic Script jobs with no Range set will run the script one time on each Engine in the pool. Before, however, once the Job was submitted, changes to the Pool members, or the addition or removal of Engines from the system were not respected by existing Jobs. Now Jobs that are submitted, even if they are partially complete, will respect these changes and will update correctly and run on all specified Engines.

Added Cinema4D error message

Added another error detection reported by Cinema4D: Project not found. (There are now about 5 different versions of the “not found” message supported by the Module... let's hope that Maxon doesn't add any new different ways to report the same error.)

All virtual modules enquote file and directory sub-parameters

All existing Virtual Modules that have file or directory type sub-parameters for a parameter (most often in the “Render Overrides” parameter) will now correctly enquote those values if they contain spaces. This should allow you to use paths with spaces in them for all of these parameters without having to manually force quote marks around them.

Maya single frame rendering could set the slice range incorrectly

The correct slice positions, including the requested slice padding, are now calculated for each slice, making this work more reliably.

Fry Module seed value is now correctly updated

The seed value used to ensure that each repetition of a frame is unique is correctly updated as a job progresses and as new jobs are submitted.

Houdini module handles the -V verbosity flag correctly

The Houdini module now correctly handles the verbosity flag so you can configure the verbosity of the output from the renderer.

XSI Module detects images and errors for Arnold plug-in renderer

The XSI virtual module was modified to allow it to detect images and errors from the Arnold plug-in renderer.

New parameter command to format times into human readable strings

If a parameter is a time value, you can use the new commands Format, FormatDuration, FormatSpan, or UnixEpoch to modify the value into something more useful:

`Format`*`:string`*	If the value is a time value from the system, it formats that time using the standard strftime formatting, with the addition of using %x for the milliseconds.
`FormatDuration`	Formats a time in milliseconds as a duration in minutes and seconds to days and hours, similar to how durations are shown in the GUI
`FormatSpan`*`:string`*	Formats a time in milliseconds as a span: `%%` - Insert % sign `%D` - Total days in this span `%d` - Total days in this span (default `%2d`) `%H` - Total hours in this span `%h` - Hours in the current day (default `%02d`) `%M` - Total minutes in this span `%m` - Minutes in the current hour (default `%02d`) `%S` - Total seconds in this span `%s` - Seconds in the current minute (default `%02d`) `%X` - Total milliseconds in this span `%x` - Milliseconds in the current second (default `%03d`) You can insert an optional width and pad character between the % and the specifier: `%`[[0]*width*]*char*
`UnixEpoch`	Converts the library time representation to a Unix time_t type value (seconds since 1-1-1970)

Master process can send messages to itself

Modules that run code in the Master (for example Distributors) can now simulating sending a message to the Master like a client, to make it easier for certain types of processing.

All distributors now can report their status

The status of the distributor allows you to see what work has been started and finished, and look for errors in distribution or reasons for why distribution may not be taking place.

Sequence Distributor will distribute a single work unit for jobs with an empty range

In the past, if you left the “Range” field blank for a ranged type job that uses the SequenceDistributor to distribute work (as most ranged render types do), the job would be forced to be “complete” as soon as it was submitted. Now, instead, such a job will get a single work unit sent out for the job. The Range and SubRange fields will be empty, so the command line for all render jobs that this applies to will now only include the range command line parameters if the range is not empty.

Renumbering adjustment is not set if no renumbering is happening

The $(RenumberStart) parameter was always being set, even if there was no renumbering set up for the job. For the most part, this did not cause any problems, but it has been corrected so that if no renumbering is set up for the job, the value will be blank, and can be excluded from a formatted command line.

Repeat/Merge Distributor

There is a new system distributor for handling the repeat + merge type renderers, like Maxwell. Because of the details of making this work, there is also a customized Job class to go with the distributor that you must use for jobs that use this distributor.

New String class

The low level string class as been replaced with a wrapper around the standard C++ template library string class.

Changed Log to only accept single string object

The system logging functions no longer do formatting of parameters. This lead to random crashing based on random user input in production. Instead the Log API takes a single String object. If you want to format that string, manually use the Sformat() API to create the log string.

Removed FixedAllocator

The FixedAllocator system has been removed as it no longer provided any performance improvement with the updated system libraries and only created more complexity and hid possible causes of crashing.

Job method to set the Note and also log the note string

There is a new Job class method to set the “Note” field for the job and also log the message into the History.log file

Component control preserves -Master and -LogCleanup flags

The Components API preserves the flags that configure operation that are passed to the calling process.

PostExecuteEvt was using the WorkFinishedEvt command string

The Job object was using the WorkFinishedEvt string for the PostExecuteEvt event command.

Smedge app startup events supply a reference to the CommandLine

The SmedgeAppStartupEvt and SmedgeClientAppStartupEvt events now contain a reference to the CommandLine object that is controlling startup, so that any changes to that object are passed along to the continued startup event sequence.

Scoped locking objects allow access to underlying lock object

The scoped thread locking objects now include public access to the underlying synchronization primitive, to give more flexibility in their use.

Options are now typed exported variables instead of defined macros

The options are no loner macros, and instead are now exported typed variables in the SmedgeLib dynamic library.

Improved handling of embedded spaces

Fixed issues with how spaces were parsed and processed when embedded in a value with an escape sequence or properly quoted. The changes include fixes to the StringTokenizer, IniFile, CommandLine and Process classes, wherever spaces could be an inter-process issue.

RenderJob correctly respects report images setting

It was reporting before even if requested not to.

Thread exceptions were not always correctly caught by the library

Certain exceptions were not caught in the correct place, resulting in incorrect behavior during the stack unwind and termination.