



SAGE™ (Scalable Adaptive Graphics Environment)



EVL Electronic Visualization Laboratory, University of Illinois at Chicago

Dana Plepys, Luc Renambot

September 2013

SAGE™ is a trademark of the Board of Trustees of the University of Illinois.

SAGE Overview: What is SAGE?	3
SAGE Features and Tiled Display Wall Configurations	5
Basic SAGE	5
SAGE Pointer.....	5
Intermediate SAGE	5
Intermediate SAGE Features	5
Advanced SAGE.....	6
Advanced SAGE Features	6
SAGE Systems Guide	8
Minimum Systems Requirements	8
SAGE Downloads	9
SAGE Binaries	9
SAGE Source.....	9
SAGE Installation (Linux Distribution)	10
Configuration.....	15
Starting SAGE	21
SAGE Users Guide	22
Basic SAGE	22
Intermediate SAGE	29
SAGE Structure	30
SAGE for Developers	35
SAGE Structure	35
Libraries	35
Dependencies	35
Folder Structure	36
Files.....	36
SAGE Code Structure	37
Initialization	37
Pixel Streaming.....	38
Widgets	39
Application Details	40
Advanced Topics	43
Troubleshooting/Technical Information.....	44
Hardware	44
OpenGL Hardware Driver	44
Video Capture Hardware Driver.....	44
Network Driver	44
Software	44
NFS Setup	44
SSH keys	44
Linux Notes	45
FAQs (Frequently Asked Questions)	48
Compiling SAGE	48
Running SAGE.....	48
Running SAGE Applications	50
Other Questions	50
Appendices	53
Audio.....	53
Room Management and Control.....	53
UltraGrid Software	53

Scalable Adaptive Graphics Environment

1. SAGE Overview: What is SAGE?

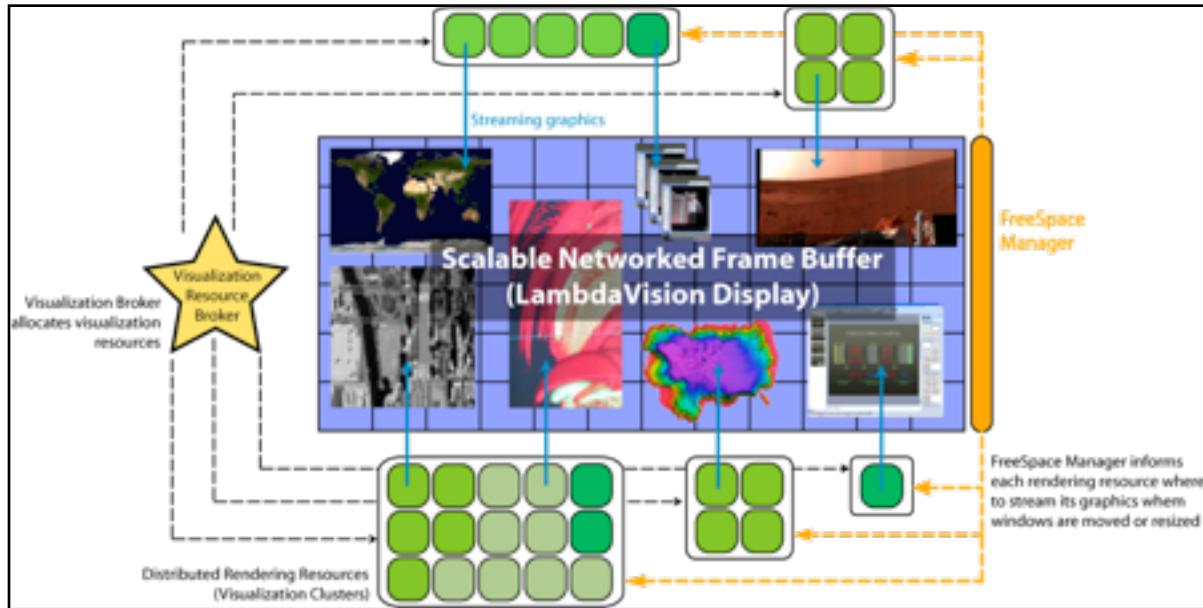
SAGE (Scalable Adaptive Graphics Environment) is cross-platform, open-source middleware that enables users to have a common operating environment, or framework, to access, display and share a variety of content – whether digital cinema animations, high-resolution images, high-definition video-teleconferencing, presentation slides, documents, spreadsheets or laptop screens – in a variety of resolutions and formats, from multiple sources, to one or more tiled display walls.



SAGE graphics streaming architecture supports the sharing of high-resolution media and data between visualization environments or among groups of collaborators at local or remote sites. Its network-centric architecture allows collaborators to share visualizations (such as remote desktop, video streams, 2D maps, and 3D renderings) on local or remote systems. Applications stream pixels over high-speed networks to displays with potentially hundreds of megapixels of contiguous display resolution.

Scalable Adaptive Graphics Environment

SAGE is software for organizing images, animations, and information on scalable display walls (OptIPortals) to help people deal with problems of scale and complexity in their data. It specializes in streaming images from remote rendering servers / supercomputers. SAGE development originally began with an NSF ITR grant (OptIPuter) in 2002 with continued support via NSF STCI grant (OCI-0943558) through 2013.



SAGE installations can vary from basic, to intermediate and advanced configurations to support a range of collaborative session requirements. The basic and intermediate configurations focus on local collaboration, with enhanced high-speed desktop sharing and whiteboard functionality, augmenting the intermediate installation. The advanced configuration supports highly interactive, multi-site remote, collaborative work, and enables multi-user interaction using a variety of input devices: 3D mouse, touch-screen and motion sensing input devices such as the Microsoft Kinect.



This classification is purely descriptive, and in no way represents limitations imposed by SAGE or its configuration, but commonly observed scenarios.

2. SAGE Features and Tiled Display Wall Configurations

2.1. Basic SAGE

A basic SAGE installation only supports local collaborative sessions by allowing multiple participants to share their desktop and assets to a local tiled display or to remote sites. The basic system does not require high-speed network for streaming applications. Desktop content is shared to and controlled on the local display through use of the SAGE Pointer application.

2.1.1. SAGE Pointer

The SAGE Pointer application extends the user's desktop interaction space to cover the whole tiled display. Using his mouse, the user can control a personalized cursor, manipulate windows, share the desktop, drag and drop local assets (PDF files, video files, Office files and images) from the desktop to the tiled display. The SAGE Pointer application can be run via a wireless network connection, and is the most commonly used modality for group collaboration (each user controls a unique cursor). Finally, the SAGE Pointer application allows the user to initiate a VNC connection between the user's laptop and the local tiled display.

2.2. Intermediate SAGE

In addition to the functionality provided by a basic SAGE installation, an intermediate configuration supports improved streaming of the desktop via the SAGE qShare application. It enables desktop applications to be shown on the local tiled display, for instance interactive whiteboard software, video editing suite or 3D modeling software.

Through desktop sharing, third party applications such as Skype can be incorporated into a SAGE session for enhanced communication between participants. It is important to note that third party applications are not integral to SAGE, but are launched as standalone applications that are shared via SAGE.

2.2.1. Intermediate SAGE Features

qShare

qShare allows for high-performance screen capture and sharing of computer desktop displays at near real-time frame rates (15 to 20 fps) to SAGE. qShare requires a network connection of at least 1Gbps between a standard desktop system (HD resolution) and computer/s running SAGE. A 10Gbps connection allows high-resolution desktops (i.e. 2560x1600 pixels) to be shared at interactive frame rates.

Whiteboard

A Wacom Cintiq display (integration of an LCD with pressure-sensitive pen technology <<http://www.wacom.com/en/creative/products/pen-displays/cintiq>>) can be used into a SAGE configuration to provide whiteboard functionality for local collaboration. Whiteboard functionality can enhance SAGE local collaborative sessions by providing a system for sketching in support of highly active user interaction, by leveraging the high frame rate of qShare (compared to VNC). This does not exclude the use of the SAGE Pointer concurrently.

Skype / Other Collaboration Tools

Incorporating communication tools for distance collaboration into a SAGE session is accomplished by near real-time sharing of the desktop via qShare. Ancillary equipment, such as a webcam, microphone and speaker set-up is needed.

2.3. Advanced SAGE

An advanced SAGE configuration provides functionality for highly interactive local and remote collaborative sessions by sharing and streaming media and data between multiple SAGE-enabled sites. Integration of video conferencing provides for enhanced communication between distributed participants, which is an integral component of networked collaboration. An additional benefit is real-time frame rates provided by the SAGE Capture hardware solution.

Advanced configurations may also incorporate a variety of user interaction alternatives, such as a multi-touch, head tracking, and motion sensing input devices for 3D hand/body tracking. Using EVL's Omicron software development kit (SDK) <<http://code.google.com/p/omicron-sdk/>>, input data from a number interface devices – such as the SAGE Pointer or commodity game controllers are streamed to the SAGE application for enhanced interactive functionality. Omicron is a user interaction library used in conjunction with SAGE; it is currently under development.

2.3.1. Advanced SAGE Features

SAGE Capture

SAGE Capture incorporates video HD capture hardware (HDMI or SDI) and 10Gbps network connectivity to capture real-time video conferencing or streamed desktop sessions. The application is command-line binary called ‘decklinkcapture’ which can be configured through a list of parameters.

Streaming Audio Manager (SAM)

SAM (Streaming Audio Manager) developed by the Calit2/UCSD Sonic Arts Group is a robust audio system providing enhanced functionality and control over SAGE audio <<https://code.google.com/p/streaming-audio/>>. Just as SAGE enables streaming pixels to video displays from any machine, SAM enables streaming audio from any machine to a central audio (SAM) machine which then plays the audio using a variety of possible audio rendering modes. Control data is exchanged via SAM and clients using Open Sound Control (OSC). OSC is also used to communicate with third-party user interfaces.

In the basic mode (default), SAM mixes all incoming audio signals to a single signal that is played out of the audio interface on the central audio (SAM) machine. To take advantage of advanced modes, a SAM renderer is required <<https://code.google.com/p/streaming-audio/wiki/DevelopersGuide>>. Advanced audio rendering modules (spacialization, etc.) enables features such as window-tracked audio and applications in SAGE, which is especially effective for tracking media across large-scale tiled displays.

The SAGE Web Control Interface (see below) can be used to enable users to interactively control SAM sound parameters, such as audio on, off, master volume, mute. This is implemented using OSC messages between the various components.

SAGE Web Control Interface

The SAGE Web Control Interface is a customizable web-based medium for interactively configuring, launching, and controlling the SAGE display and associated technologies through an HTML5/Node/Javascript framework. It is a cross-platform, browser-based interface that works on any device connected to the Internet (e.g. tablets, mobile phones, laptops), enabling users to interactively control all aspects of their collaborative environment (processes, applications, displays, devices, etc.). The SAGE Web Control Interface runs on a web server typically installed on the SAGE control computer. It is built on top of Node.js jQuery and Javascript.

Multi-Touch Overlay

A SAGE tiled display can be configured to support multi-participant touch interaction using the PQ Labs Multi-Touch solution. A small server process connects to the PQLabs touch server (running on the Windows machine connected to the touch overlay hardware through USB), analyzes the touch information and gestures and translates them into the SAGE pointer protocol to behave like a pointer. It allows the user to perform the usual SAGE interaction commands (open application, navigate the file library, create partition, etc.). A few more gestures have been integrated such as ‘clear the wall’ or ‘close application’. Plans are underway to integrate this extraneous process with the Omicron server to leverage more interaction devices.

3. SAGE Systems Guide

There are two types of hardware configurations that can support SAGE - a single machine or a cluster based configuration. Though a single machine alternative may be attractive due to its apparent streamlined nature for overall systems administration and price, this alternative is far less scalable beyond a basic tiled environment (limited to ~4x4 tiled display). A GPU workstation will still have the capacity to do video capture and network at 10G, but at a cost almost equivalent to a small cluster. Bear in mind, the resolution of the tiled display, stereo capabilities, etc. all factor significantly into the choice of computer infrastructure supporting SAGE. There is seemingly greater flexibility and scalability of using a cluster based approach, especially considering rapidly changing display technologies.

It is highly recommended to consult the SAGE user community via the on-line forum at <<http://www.sagecommons.org/support/forum/>>. Currently, there are over 100 SAGE sites worldwide with a large variety of systems configurations. Advisement from this community regarding advantages and disadvantages of one configuration over another can be sought through forum postings and discussions.

Sample configurations valid at the time of writing this document (early 2013).

3.1. Minimum Systems Requirements

Basic SAGE

- 1 control computer (SAGE manager)
- 2 displays/computer (# display tiles to computer ratio)

Specifications/Computer:

- Dual Xeon E5-2690 Supermicro GPU server
- 64GB memory
- NVIDIA GTX680 graphics card (per machine)
- Dual 10Gbps network card (per machine)
- 160GB SSD
- 2TB hard drive

Networking:

- 10Gbps network switch

Audio:

- Asus Xonar DX PCI Express Sound Card
- 2 speakers (Dell AY410 Multimedia Speaker System) connected to control computer

Intermediate SAGE (Basic SAGE +)

Mac mini:

- 2.7GHz Dual-Core Intel Core i7
- 8GB 1333MHz DDR3 SDRAM - 2x4GB
- 256GB Solid State Drive
- Mini Display Port to DVI Adapter

qShare:

- Wacom Cintiq 24HD Interactive Pen Display (whiteboard) controlled by the MacMini

Networking:

- 1Gbps network connection to the 10Gbps switch running SAGE

Audio:

- 4 Channel audio mixer (Rolls MX44s Mini-Mix IV)
- Rolls PS27 12VDC AC adapter

Advanced SAGE (Intermediate SAGE +)

SAGE Capture:

- Blackmagic Design ‘Intensity Pro’ HDMI capture card (installed in control computer or additional PC). For more complex scenarios, the ‘Studio’ model supports SDI and HDMI, the ‘duo’ model supports two SDI video signals simultaneously, the ‘quad’ model supports four SDI inputs, and the ‘4K Extreme’ supports 1080p60, 2K and 4K signals. 10Gbps Ethernet card is needed for anything higher than 1080p30 signals. During the capture, the video is pushed to the SAGE wall, and the audio is streamed to SAM (optionally). This is not a video-conferencing system but a means to display local content (which can be the output of a conferencing system outputting HDMI or SDI).

Video Conferencing:

- LifeSize Passport system (720p30) with PTZ camera/micpod (any video-conference system that outputs HD video SDI or HDMI can be used; SAGE developers have experience with LifeSize and Polycom systems.)
- ‘UltraGrid’ software <<https://www.sitola.cz/igrid/index.php/UltraGrid>> is also provided with the SAGE. It supports various HD capture cards and provides a SAGE display module. UltraGrid supports audio/video streaming, compression, and robust streaming over wide-area networks.

Audio:

- Basic & Intermediate Audio System Requirements

Multi-touch System:

- PQ Labs Multi-Touch Solution G3 Basic (6 touch points/inch), Standard (12 touch points/inch), or Plus Overlay (32 touch points/inch)
- Dell XPS 8500 (Windows 7 Professional, 64 Bit) connected through USB to the touch hardware
- Mid-range CPU needed for data processing
- 8GB Dual Channel DDR3 1600MHz – 2 DIMMs memory

3.2. SAGE Downloads

3.2.1. SAGE Binaries

Binary packages are available for various 64bit Linux distributions <<http://www.sagecommons.org/resources/downloads/>>. As of February 2013, the following are available for version 3.4.1: openSuse 12.2 and 12.1, Ubuntu 12.04 LTS, Fedora 17 and 18.

3.2.2. SAGE Source

The source code is now available through EVL’s subversion (SVN) server. Access the server in read-only mode at: <<svn://iridium.evl.uic.edu/sage2012/trunk/>>

Code: `svn co <svn://iridium.evl.uic.edu/sage2012/trunk/ sage-dev>`

As of early November 2012, a few extra branches exist containing on-going work on the audio API and on stereo 3D support.

Compilation / Installation:

```
cd sage-dev  
.configure  
make (or in parallel: make -j 12)  
make build  
make install (by default in $HOME/sage, change the PREFIX variable in Makefile)
```

Run:

```
cd $SAGE_DIRECTORY/bin  
.sage  
press START/STOP
```

Configuration:

Configuration files are located in \$SAGE_DIRECTORY/sageConfig or ~/.sageConfig

- *fsManager.conf*: global settings (IP address of master, tiling file, ...)
- *stdtile.conf*: layout of your cluster and displays

Basic SAGE Bootable Distribution

A bootable and installable complete distribution to run SAGE is based on the latest openSuse 12.1 with all the dependencies necessary to run SAGE. Burn a DVD, copy the image, boot from the DVD, and install the OS on the hard drive (password: linux).

Download the ISO file (1.5GB): DVD (posted on: 11/11/2012): <<http://www.sagecommons.org/resources/downloads/>>

Download SAGEPointer (Windows, Mac OSX Lion 10.7.x, Mountain Lion 10.8.x or Snow Leopard 10.6.): <<http://www.sagecommons.org/resources/sage-pointer-and-ui/>>

SAGE Applications

Download qShare

Windows and Mac OS X binaries: <<http://www.sagecommons.org/resources/sage-pointer-and-ui>>

Mac binary (Snow Leopard and Lion): Qshare-mac (23.62MB) <<http://www.sagecommons.org/wp-content/plugins/download-monitor/download.php?id=40>>

3.3. SAGE Installation (Linux Distribution)

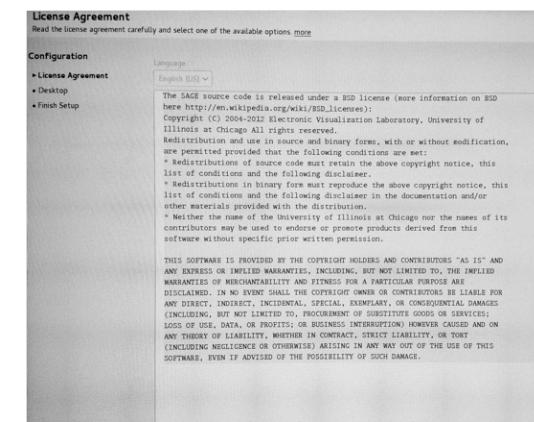
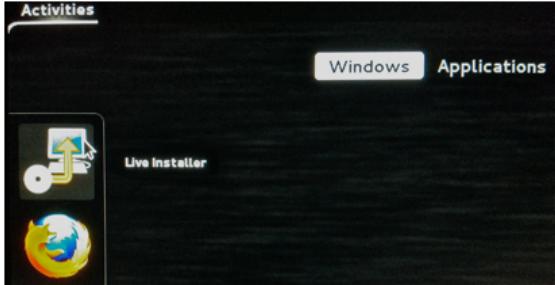
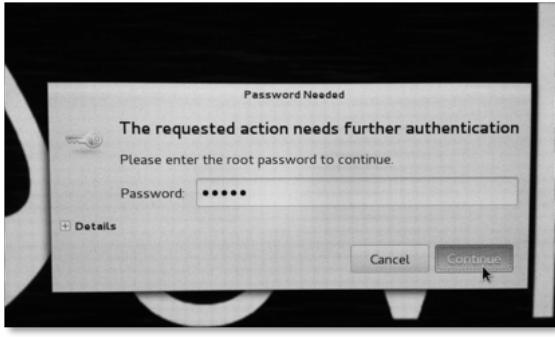
SAGE is installed on all computers comprising the cluster, beginning with the control computer and then subsequent display nodes. Linux based on openSuse is installed on all computer nodes.

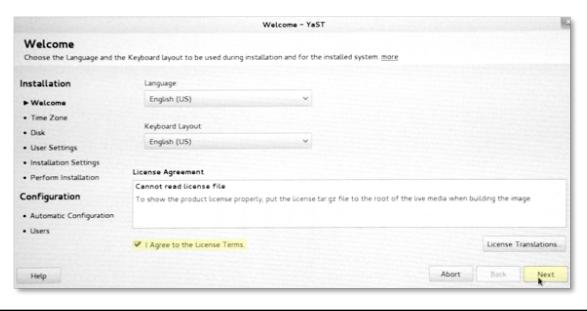
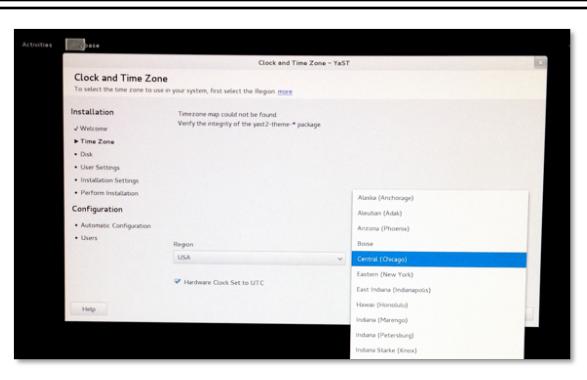
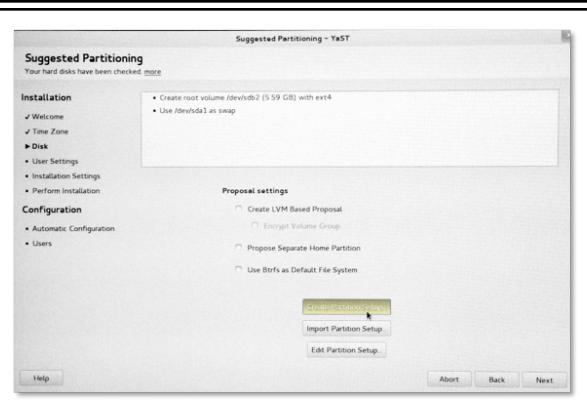
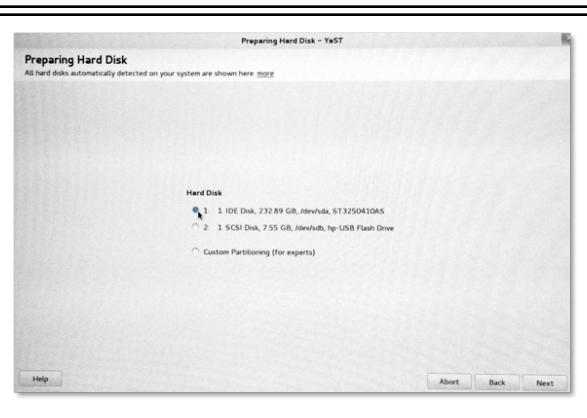
Notes:

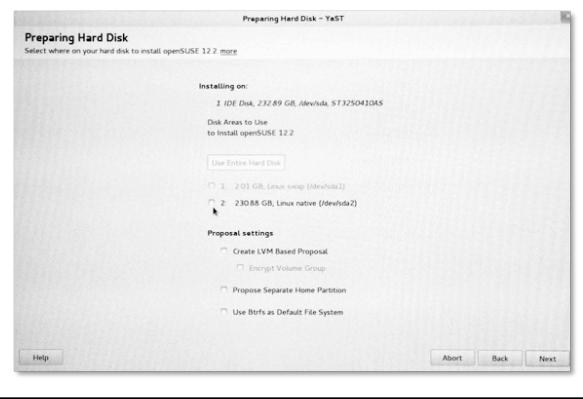
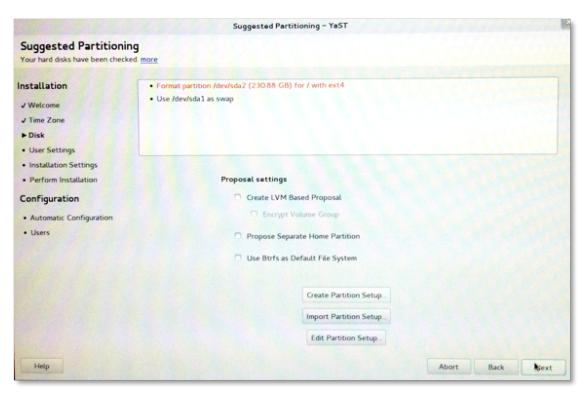
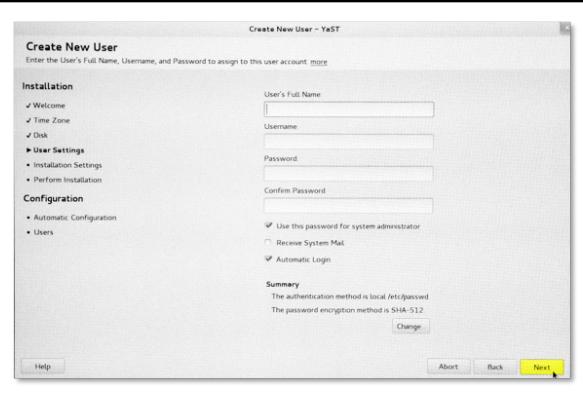
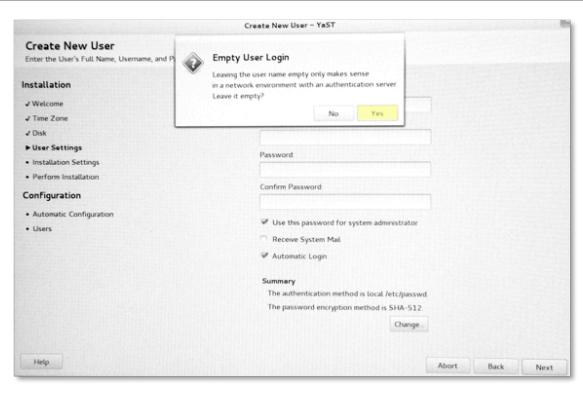
- The control computer exports the SAGE home directory via NFS to the display nodes.
- The control computer must be able to securely connect (via SSH) to all the SAGE nodes (display, application, control...). Refer to SSH documentation on how to setup SSH keys across machines for login without password.
- X11 server and OpenGL driver must be configured on all nodes.
- Cluster setup (networking, shared file system, account management, ...) is outside the scope of this document. Please refer to the openSuse documentation of your favorite Linux distribution. The SAGE developers do not support cluster configuration outside of SAGE specific issues.

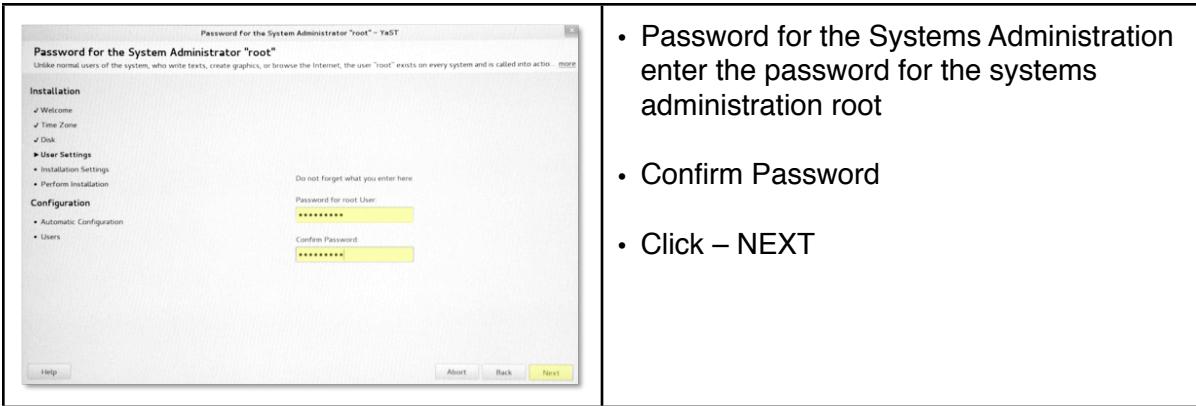
Control Computer Installation

A bootable Linux distribution includes all the SAGE software dependencies for running SAGE and writing applications. Download the ISO file onto a DVD or a USB drive and boot your PC from it. The current version is based on openSUSE 64bit version 12.2.

	<ul style="list-style-type: none"> • Insert SAGE installation media (USB or DVD) in the control computer • First step will automatically launch or manually select – SAGE Desktop suse 12.2 • Press – ENTER
	<ul style="list-style-type: none"> • Licensing Terms screen choose – Yes, I Agree to License Agreement • Click – NEXT
	<ul style="list-style-type: none"> • Activities menu (upper left corner) select – Live Installer
	<ul style="list-style-type: none"> • Enter password: linux • Click – Continue

	<ul style="list-style-type: none"> Welcome YaST check – I Agree to License Terms Click – NEXT
	<ul style="list-style-type: none"> YaST Time Zone select – your time zone from the Time Zone List Click – NEXT
	<ul style="list-style-type: none"> YaST Disk uncheck – Propose Separate Home Partition (site dependent) Select – Create Partition Set-up Click NEXT
	<ul style="list-style-type: none"> Select the disk to install SAGE Click – NEXT

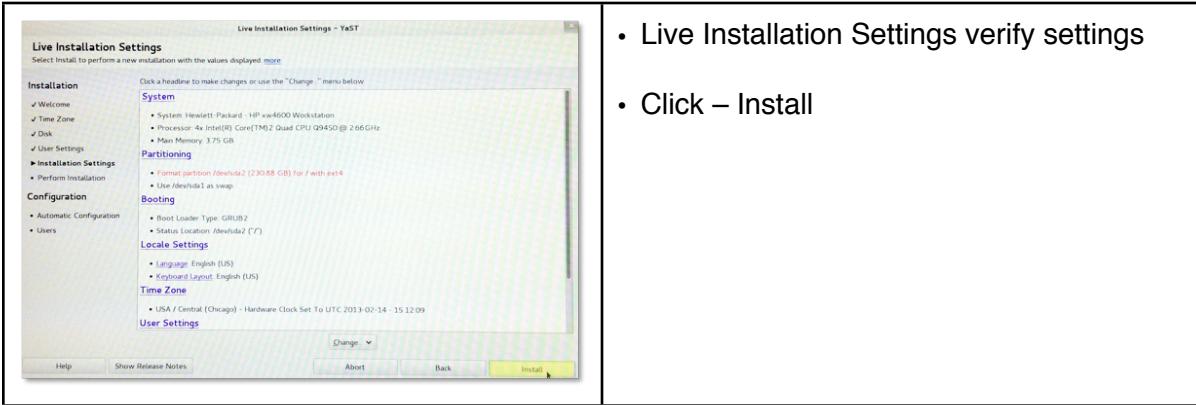
	<ul style="list-style-type: none"> • Preparing Hard Disk • select – the partition to install SAGE or select – Use Entire Hard Disk • Click – NEXT
	<ul style="list-style-type: none"> • Click – NEXT again to confirm Suggested Partition
	<ul style="list-style-type: none"> • Create New User screen click – NEXT
	<ul style="list-style-type: none"> • Empty User Login click – YES



- Password for the Systems Administration enter the password for the systems administration root

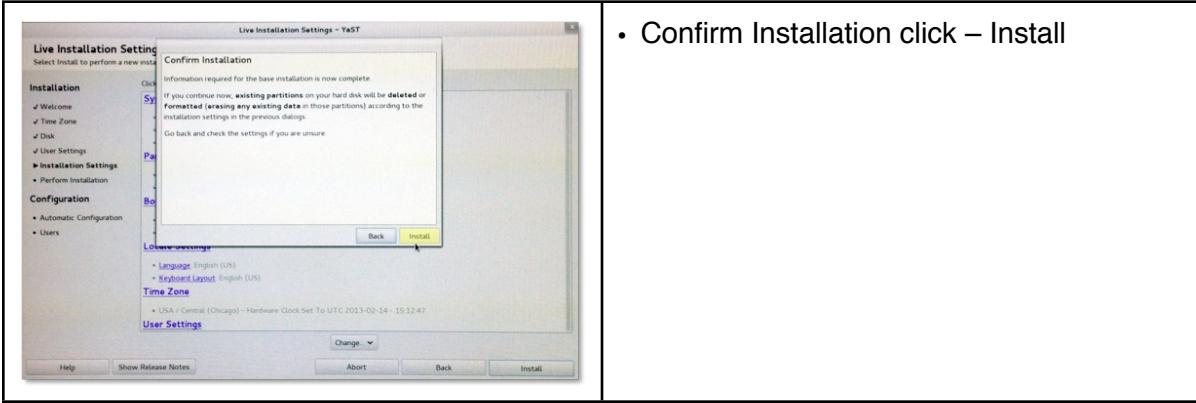
- Confirm Password

- Click – NEXT

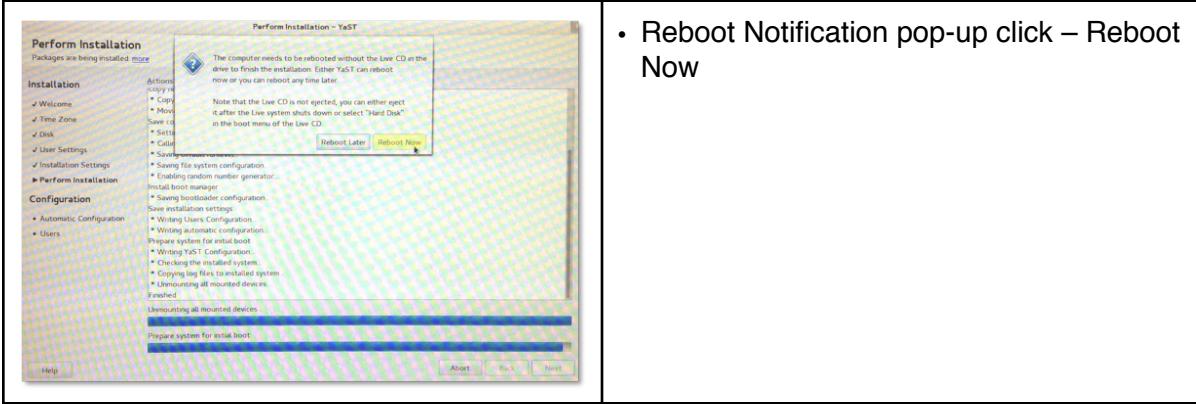


- Live Installation Settings verify settings

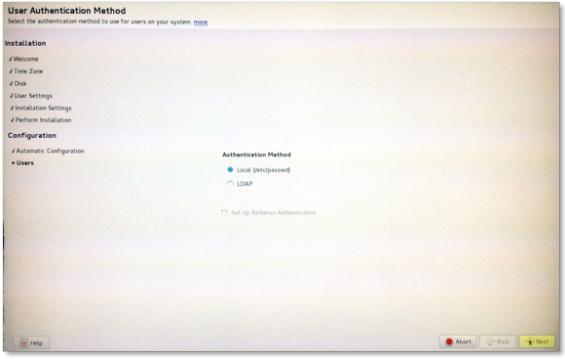
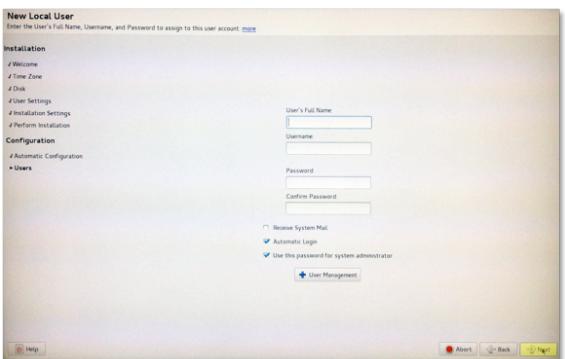
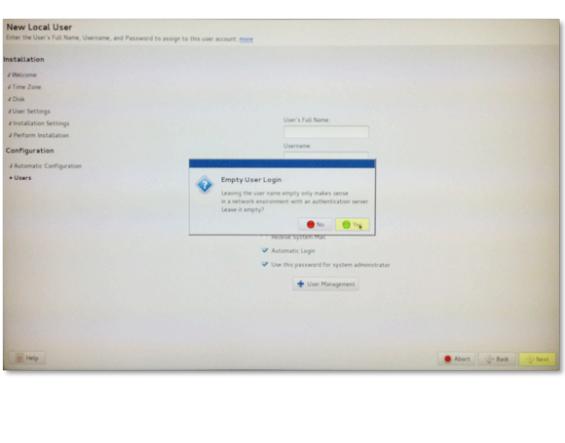
- Click – Install



- Confirm Installation click – Install



- Reboot Notification pop-up click – Reboot Now

	<ul style="list-style-type: none"> • Remove installation media • After rebooting your system openSUSE will launch Automatic Configuration • User Authentication Method click – NEXT
	<ul style="list-style-type: none"> • New Local User click – NEXT
	<ul style="list-style-type: none"> • Empty User Login pop-up click – NEXT • Click – YES • Click – NEXT again • Empty User Login pop-up repeats • Click – YES again • Empty User Login pop-up repeats • Click – YES again

3.4. Configuration

SAGE is designed for computer cluster installation. The initial SAGE installation and configuration is done on the cluster control computer for testing purposes, and then installed and configured on display nodes.

During the configuration process, the SAGE Streaming Audio Manager (SAM) will also need to be configured. Consult the SAM user's guide <<https://code.google.com/p/streaming-audio/wiki/UsersGuide>> where the config file parameters are fully described. The SAGE configure script is set to automatically build and install SAM on the head node. SAM can also be installed on a separate machine (linux or mac). Mplayer will need to be updated with the SAM machine IP address (see 6.4 Application Details; SAM audio module).

Control Computer Test Configuration

	<ul style="list-style-type: none"> Activities menu select – Applications
	<ul style="list-style-type: none"> Select – GNOME Terminal In the Terminal Window type – sage Press – ENTER SAGE Launcher Window – Press Start
	<ul style="list-style-type: none"> SAGE Connection Window – Press Quit
	<ul style="list-style-type: none"> Alt_Tab to Python SAGE icon

- SAGE Launcher Window – Press Stop
- Close SAGE Launcher – Click X in upper right corner
- Copy config files to User SAGE profile

Terminal Window:

- Type cd .sageConfig
- cp /usr/local/sage/sageConfig/* .
- Edit fsManager.conf file using your preferred text editor (e.g. vi fsManager.conf)
- Modify tileConfiguration stdtile-1.conf to stdtile-test.conf
- Change fullscreen to 1
- cp stdtile-1.conf stdtile-test.conf
- Edit stdtile-test.conf
- Modify the test config file to match the resolution of your display
- Save the test config file

- Verify SAGE test configuration– type sage in terminal window to run SAGE

Display Node Configuration

After installation and completing the test configuration of SAGE on the control computer, install SAGE onto all display nodes comprising the cluster, and configure the display node network.

- Modify fsManager.conf using your preferred text editor (e.g. vi fsManager.conf)
- Modify the IP address of the control computer
- Change Fullscreen to 2
- Modify stdtile-test.conf:
 - Enter the Dimensions of the tiled display
 - Enter the Resolution of each display node
 - Enter the Number of machines in the computer cluster
 - Enter the list of all display nodes IP addresses and Monitor arrangement (location) in tiled display

Sample Configuration

Single screen, HD resolution (1920x1080) with position of the screen shifted by 1920 pixels

```

TileDisplay
    Dimensions 1 1
    Millions 0.0 0.0 0.0 0.0
    Resolution 1920 1080
    PPI 90
    Machines 1

DisplayNode
    Name local
    IP 10.0.1.2
    Monitors 1 (0,0)
    Pos (1920,0)
  
```

Dual-screen setup on a single machine

```
TileDisplay
    Dimensions 2 1
    Mullions 0.0 0.0 0.0 0.0
    Resolution 1920 1080
    PPI 90
    Machines 1

DisplayNode
    Name local
    IP 192.168.1.10
    Monitors 2 (0,0) (1,0)
```

Quad-screen on a single machine, treated as a single large screen (2x2 with 1366x768 monitors) (i.e. Xinerama mode in Xorg configuration)

```
TileDisplay
    Dimensions 2 1
    Mullions 0.0 0.0 0.0 0.0
    Resolution 2732 1536
    PPI 90
    Machines 1

DisplayNode
    Name local
    IP 192.168.1.10
```

Same system as above but configured as multiple individual Xorg servers

Note the syntax IPAddresse:portnumber:Xserver. Since the four Xorg servers are running on the same host, different port numbers are needed for each tile. The multiple servers being used are considered four distinct machines (hence ‘Machines 4’).

To take screen borders into account, the ‘mullion’ sizes are specified as “left right bottom top” order with values specified in inches. PPI (pixels per inch) must be accurate.

```
TileDisplay
    Dimensions 2 2
    Mullions 0.1270 0.0762 0.0762 0.1270
    Resolution 1920 1080
    PPI 90
    Machines 4

DisplayNode
    Name local
    IP 192.168.1.10:22000:0.0

DisplayNode
    Name local
    IP 192.168.1.10:22002:0.1
    Monitors 1 (0,1)

DisplayNode
    Name local
    IP 192.168.1.10:22004:0.2
    Monitors 1 (1,0)

DisplayNode
    Name local
    IP 192.168.1.10:22006:0.3
    Monitors 1 (1,1)
```

Large single machine driving 18 screens: each single tile of 4080x768 is in fact 3 individual monitors driven by a Matrox “Triple Head ToGo” box, and each of the 3 GPUs drives two of these tiles, for a total of 6 Xorg servers.

```
TileDisplay
    Dimensions 2 3
    Mullions 0 0 0 0
    Resolution 4080 768
    PPI 34
    Machines 6

DisplayNode
    Name local
    IP 67.58.62.57:22011:0.0
    Monitors 1 (0,2)
DisplayNode
    Name local
    IP 67.58.62.57:22022:0.1
    Monitors 1 (1,2)
DisplayNode
    Name local
    IP 67.58.62.57:22033:0.2
    Monitors 1 (0,1)
DisplayNode
    Name local
    IP 67.58.62.57:22044:0.3
    Monitors 1 (0,0)
DisplayNode
    Name local
    IP 67.58.62.57:22055:0.4
    Monitors 1 (1,1)
DisplayNode
    Name local
    IP 67.58.62.57:22066:0.5
    Monitors 1 (1,0)
```

Cluster of 6 display nodes, each driving 3 screens (the screens are in Xinerama mode). The split of each desktop into 3 tiles allows for border configuration (mullions).

```
TileDisplay
    Dimensions 6 3
    Mullions 0.1270 0.0762 0.0762 0.1270
    Resolution 1366 768
    PPI 34
    Machines 6

    DisplayNode
        Name local0
        IP 67.58.62.70
        Monitors 3 (0,2) (1,2) (2,2)

    DisplayNode
        Name local1
        IP 67.58.62.71
        Monitors 3 (0,1) (1,1) (2,1)

    DisplayNode
        Name local2
        IP 67.58.62.72
        Monitors 3 (0,0) (1,0) (2,0)

    DisplayNode
        Name local3
        IP 67.58.62.73
        Monitors 3 (3,2) (4,2) (5,2)

    DisplayNode
        Name local4
        IP 67.58.62.74
        Monitors 3 (3,1) (4,1) (5,1)

    DisplayNode
        Name local5
        IP 67.58.62.75
        Monitors 3 (3,0) (4,0) (5,0)
```

4. Starting SAGE

SAGE can be started in two ways:

- 'sage' command: this is a python-based graphical user interface to start and stop SAGE, and to configure main SAGE services through the 'extended' panel. It requires the user to be logged into the graphical console of the master node (inconvenient for remote administration).
- shell script: GO-sage to start, and KILL_ALL to clear all the processes.

GO-sage:

```
# start in the home directory
  • cd $HOME
  • ./KILL_ALL

# allows Xorg connection and disable screen saver and screen blanking
  • xhost +local:
  • xset dpms force on
  • xset -dpms
  • xset -dpms s reset
  • xset s off s noblank

# start SAM
  • cd $SAGE_DIRECTORY/bin
  • ./sam.sh &

# start fsManager
  • cd $SAGE_DIRECTORY/bin
  • ./fsManager &
  • sleep 3

# start application launcher
  • python appLauncher/appLauncher.py -v &
  • sleep 2

# start DIM
  • # no sharing site
  • # python ../dim/dim.pyc &
  • # with sharing
  • python ../dim/dim.pyc --shared_host site1:10.0.1.2,site2:10.0.8.2 -v &
  • sleep 2

# stat the file library server
  • cd fileServer && python fileServer.py &
```

KILL_ALL: (*uses a file called '\$HOME/tileNodes.list' containing a list of IP addresses*)

```
# kill SAGE python services
  • killall -9 python

# kill SAM
  • killall -9 sam

# kill processes on all machines
  • for node in `cat $HOME/tileNodes.list`
  • do
    • ssh -fx $node "killall -9 fsManager sageDisplayManager imageviewer mplayer bplay-noglut VNCViewer render atlantis atlantis-mpi checker pdfviewer image3d"
  • done

example of tileNodes.list file (text file with a list of IP addresses)
  • 127.0.0.1
  • 10.0.8.2
  • 10.0.8.3
```

5. SAGE Users Guide

5.1. Basic SAGE

SAGE User Interface

The SAGE display ‘canvas’ includes a graphical user interface for managing files, media and sessions, arranging the canvas lay-out, sharing a SAGE display canvas via a distributed session with connected collaborators, and other tasks.



Media and Session folders appear on the left side of the display canvas. These folders consist of files and session states that have been saved for loading into SAGE.

At the top of the SAGE display ‘canvas’, there are tabs for users to connect and share SAGE assets with other collaborators also running SAGE. In addition to sharing SAGE sessions, there is also a tab used to remove content from the SAGE display canvas.

Widgets to organize displayed content are located on the right side of the interface. Various configurations are provided to tile the display or to segment a portion of the larger display either vertically or horizontally and scaling the content to fit within a desired configuration without image overlap.

In addition to these organizational widgets, there are PDF and EDL buttons to assemble a series of files into a single one file. The PDF button is used to create one pdf from multiple pdf files. The EDL button, assembles multiple movie files of the same resolution into a single .m3u playlist.

Media / Session Folders

Media folders contain a number of different file types, which are organized into separate folders inside the SAGE file library (inside the user’s directory `~/.sageConfig/fileServer/fileLibrary`).

The association between a file category and file extensions is configured in the following file:
`$SAGE_DIRECTORY/sageConfig/fileServer/fileServer.conf`

Audio	Images	Office	PDF	Playlist	Movies	3D
sound files	image files	Microsoft office documents	PDF documents	Playlist files	Movie files	3D Images

Audio	Images	Office	PDF	Playlist	Movies	3D
.mp3 .wav	.jpg .jpeg .gif .png .tif .tiff .bmp .pcx .rgb .rgba .ico	.doc .docx .ppt .pptx .xls .xlsx	.pdf	.m3u .pls .wpl	.avi .mpg .mpeg .mov .mp4 .flv .wmv .m4v .mp4 .mkv .vob .webm	.mpo .jps .pns

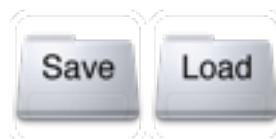
New file extensions can be added into the configuration file; restarting SAGE to take effect. The same file associates a file type and a command to launch the pertinent SAGE application:

```
app:image = imageviewer
app:video = mplayer -ao sam,alsa -sws 4 -softvol -softvol-max 300 -loop 0 -
quiet -vo sage -framedrop -title %
app:playlist = mplayer -ao sam,alsa -sws 4 -softvol -softvol-max 300 -loop 0 -
quiet -fixed-vo -vo sage -framedrop -playlist
app:audio = mplayer -ao sam,alsa -softvol -softvol-max 300 -loop 0 -quiet
app:pdf = pdfviewer
app:office = officeviewer
app:3dpano = Stereo3D
```

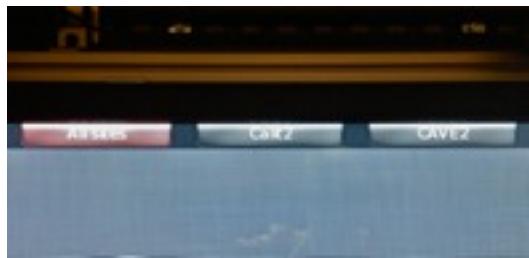
Selecting a Media folder displays file directories (photo left) and image thumbnails stored in those directories. Images are selected by navigating to the desired image thumbnail and clicking on the highlighted image.



Session folders allow users to save the state and configuration of the SAGE display canvas, preserving media windows location and size for future use:



Connect / Remove



At the top of the SAGE interface are tabs which are used to connect to SAGE collaborators for sharing of SAGE content. SAGE sessions are shared by connecting to SAGE user IP addresses. These connections can be made to 'All sites' or to individual sites set as default collaborators via interface customization. Files are shared by dragging and dropping onto the tab of the desired location.

The 'Remove' tab is used to clean the SAGE display canvas by dragging and dropping unwanted content and depositing items onto the tab.

Organization / Tile



As more files and media are displayed to the SAGE canvas, organizing the assets to most efficiently use the display real estate, and to avoid overlapping is advantageous.



The SAGE user interface includes a tiling feature to quickly tile the content across the available space. Assets are scaled to fit without overlap.

The vertical and horizontal buttons allow users to segment the SAGE display canvas into distinct areas; content can be independently organized and/or isolated for better organization and more efficient work sessions. Users control display segmentation by dragging the vertical and horizontal dividers to the desired location on the SAGE display.

PDF / EDL

The PDF and EDL buttons are used to organize like file types into a single file. Multiple pdfs dragged and dropped to the PDF button are concatenated into a single pdf. Similarly, multiple movie files can be assembled into a EDL (edit decision list) as a .m3u media playlist file. All movie files merged into a single playlist must be of the same resolution.

Document Widgets

Users navigate data files consisting of multiple pages with widgets displayed at the bottom of the document display window. These widgets are single page backward, multi-page backward, and multi-page forward, and single page forward navigation for more efficient access of large documents.

		Page 1 of 5		
go to first page	previous page	status info	next page	last page

Movie player controls are displayed at the bottom of the media window and include controls to: step back, play, pause, stop, step forward, volume control and mute.

go to beginning	play / pause	stop	step ahead	mute / unmute

SAGE Pointer

Users interact with the SAGE using an on-screen pointer. Each user has a unique pointer with which they control (size, position) windows displayed in SAGE, as well as interact with SAGE interface controls such as display, open, delete files/sessions stored on the SAGE server, organize and adjust the display environment according to preset configurations, and connect to collaborators at remote sites.

The SAGE Pointer application is used to interact with SAGE. It can be configured and customized to select which edge with which to enter SAGE, personalize the label and color of the cursor (when in SAGE); as well as to share media and data files to SAGE (pdfs, images, videos, YouTube and Vimeo links?).

The SAGE Pointer application needs to be downloaded to users' desktop or laptop computers to interact with SAGE. Running the SAGE Pointer application will allow users to connect to SAGE and share their desktop display.

Download SAGE Pointer Application

The screenshot shows the Mac OS X System Preferences window. The 'Universal' icon in the bottom right corner of the main grid is highlighted with a yellow box. Below the grid, the 'Universal Access' pane is open, also with its title bar highlighted with a yellow box. The pane contains settings for VoiceOver, Zoom, Display, and contrast, along with checkboxes for 'Enable access for assistive devices' and 'Show Universal Access status in the menu bar'.

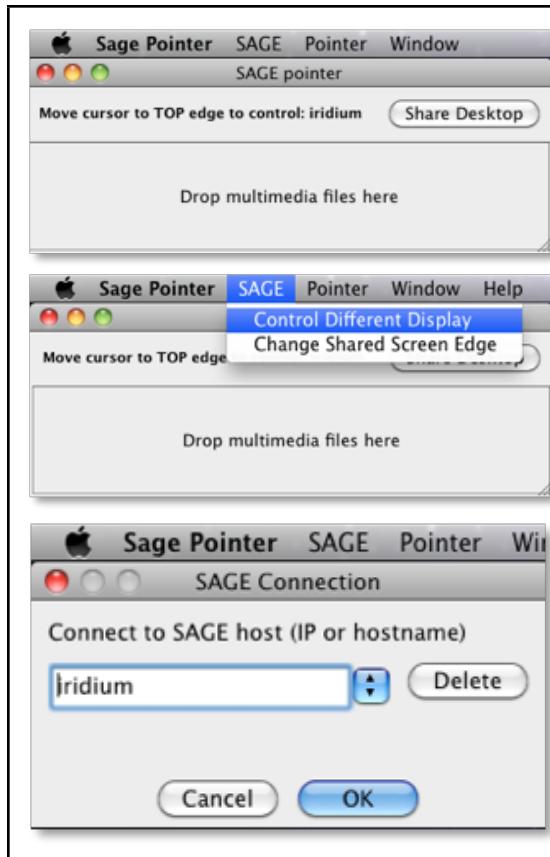
Windows:

- Unzip the downloaded file (SagePointer-win32)
- Run sagePointer.exe

Mac:

- System Preferences >Universal Access (Snow Leopard) or Accessibility (Lion / Mountain Lion)
- Select - Enable access for assistive devices
- Unzip the downloaded file (SagePointer-mac or SagePointer-snow)
- Drag sagePointer.app to Applications folder

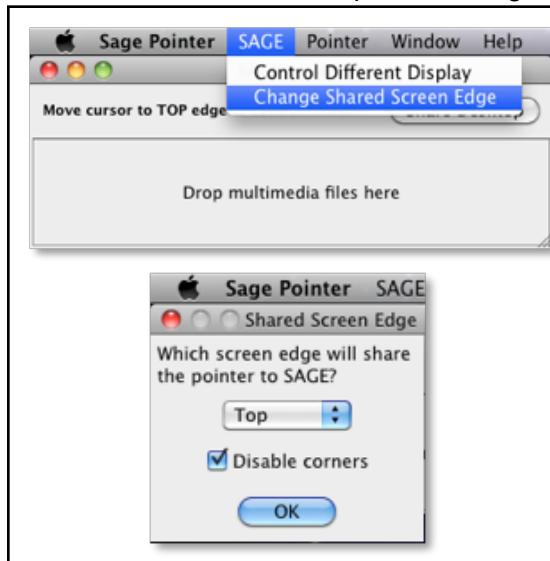
Connect to SAGE



- Launch the SAGE Pointer application (a small rectangular window will appear at the top left corner of the desktop display)
- Select SAGE drop-down menu
- Select Control Different Display
- Enter IP address or hostname of the SAGE control computer (head node where SAGE is installed)
- Select OK

Customize SAGE Pointer Screen Edge

Users can select which screen edge is used to move the cursor from their local machine to SAGE. The default desktop screen edge is “top”.



- To choose another screen edge of your desktop computer to extend to SAGE:
- Select SAGE drop-down menu
- Select Change Shared Screen Edge
- Choose preferred edge to use to interact with SAGE
- To allow OS Hot Corners check – Disable Corners
- Select OK

Cursor Color & Label

The user's cursor color and label appearing in SAGE are modified in the "Pointer" menu.

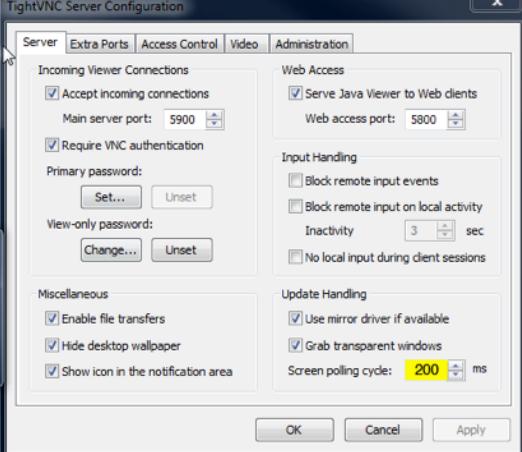
- Select Pointer drop-down menu
- Select Change Label
- Enter text label for cursor in the field provided
- Select Pointer drop-down menu
- Select Change Color
- Select cursor color via color-picker widget

Share Files

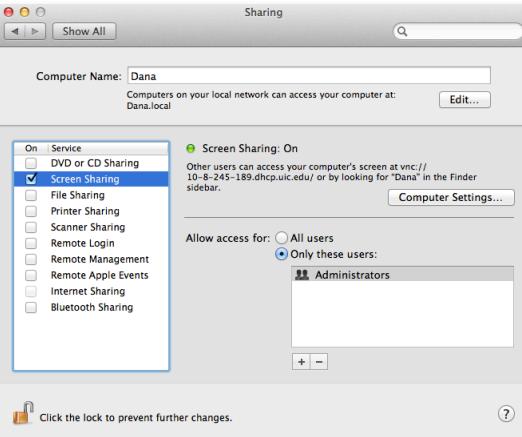
Media files (pdf, jpg, gif, png, tiff, mov, avi, etc) can be shared to SAGE by dragging and dropping selected files to the SAGE Pointer window on your laptop. Allowed file formats are the same as defined by the file library above.

Large files may take some time to transfer, and should be limited to <2GB.

Desktop Sharing Using VNC (Windows)

	<ul style="list-style-type: none"> Download, install and run TightVNC <www.tightvnc.com/download.php> Right click on TightVNC icon Under Configuration - set up a view-only password (8 chars maximum; passwords are sent in cleartext to the SAGE console) Under Update Handling - set Screen polling cycle to 200ms
	Any other VNC server should work fine (e.g. RealVNC)

Desktop Sharing (Mac)

	<ul style="list-style-type: none"> Under System Preferences / Internet & Wireless - select Sharing Select - Screen Sharing
--	--

Sharing the Desktop



- Launch SAGE Pointer application

- Select - Share Desktop

Windows (XP), Mac (10.6.x and below):

- Enter VNC password (password used in Desktop Sharing Using VNC *above* (**passwords are sent in cleartext to the SAGE console**))

Mac OSX Lion (10.7.x):

- Enter system password (**passwords are sent in cleartext to the SAGE console**)

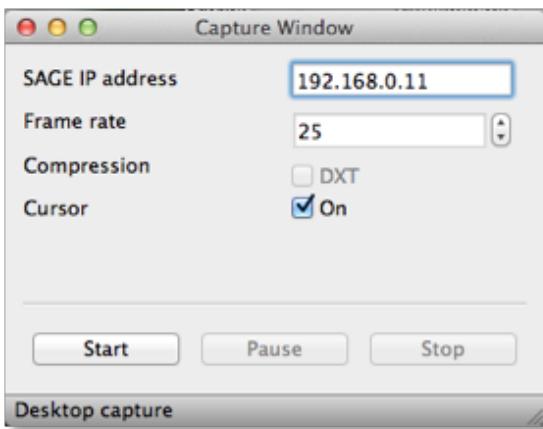
- Select User Username

- Enter system username

- Select OK

5.2. Intermediate SAGE

Using the high-speed desktop sharing application called *qShare*.



- Launch the downloaded qShare application (Windows: Sage Qshare Win32; mac: Qshare-mac)
- Capture Window>SAGE IP Address - enter the IP address of the SAGE control computer's network interface (remember that qShare will connect to the display nodes directly, hence the machine running qShare should be able to communicate with all the display nodes – check the IP addresses specified in the stdtile.conf file)
- Frame rate = default (25) – lower frame rate to adapt to network conditions
- Start

Enabling Whiteboard

- Download Open-Sankoré <<http://open-sankore.org/>> (*Open-Sankoré is only one example application; many others exist such as Autodesk SketchBook*).

- Launch Open-Sankoré

The Whiteboard is displayed using qShare to SAGE

Skype Session

- Download Skype <<http://www.skype.com/en/download-skype/skype-for-computer/>>
- Launch Skype

The Skype session is displayed using qShare to SAGE

5.3. SAGE Structure

When loading configuration files in SAGE, the current directory is looked in first, followed by the user's .sageConfig folder (user home directory \$HOME/.sageConfig folder). And then, the SAGE installation folder specified by the shell environment variable \$SAGE_DIRECTORY (this could be inside any folder of the user directories or in default system folder /usr/local/sage).

This technique has the benefit of letting the administrator setup a basic configuration, but users can copy configuration files into their own directory to customize. The customized files will be found before the system files and loaded.

Structure - Binary or system-wide installation

SAGE has been installed from a binary RPM file or installed by the 'root' user in the default system folder /usr/local/sage.

/bin: folders with SAGE executables and SAGE and SAM applications

SAGE / SAM applications:

- atlantis: sample Glut OpenGL application
- bplay-noglut: uncompressed animation playback
- checker: sample benchmark application (streaming a memory buffer)
- decklinkcapture: blackmagic decklink HD video capture, with SAM audio streaming
- image3d / images3d: stereo 3D image viewer
- imageviewer: basic image viewing application based of ImageMagick
- mplayer: Mplayer video playback with SAGE and SAM plugin
- officeviewer: shell script converting Office files (doc, xls, ppt, etc.) on-the-fly to PDF using LibreOffice environment
- qshare: high-speed desktop sharing
- render: sample Glut OpenGL application
- sage: python application to start/stop SAGE
- samugen: basic SAM audio test application
- Stereo3D: shell script converting popular 3D image file formats for the images3d SAGE application

- *uv*: Ultragrid HD/2K/4K audio and video streaming software
- *VNCViewer*: VNC viewer application used for desktop sharing. One instance is launched for each shared desktop. It bridges the remote desktop to the SAGE wall.
- *webcam*: a trivial V4L (video for Linux) webcam capture application.

SAGE / SAM Runtime

- *fsManager*: main SAGE process started only on the master node of the cluster or main machine in single-node setup. *fsManager* coordinates all the SAGE applications and the wall but is not involved in streaming. Applications stream directly to display nodes.
- *sageDisplayManager*: display process run on every display node of the system. Multiple instances might run on the SAM machine if multiple Xorg server are used. The processes are started automatically by *fsManager* using ssh commands and according to the tiling configuration file (*stdtile.conf*).
- *fsConsole*: command-line admin application into *fsManager*
- *sam*: streaming audio manager: server for SAM, runs on the audio rendering machine (*sam.sh* shell script provides the most common parameters for stereo playback)

SAGE Services

- *appLauncher*: folder containing the utility service that starts SAGE applications on the behalf of the user. Applications are staged and launched using *ssh* command onto the specified application machines, according to the ‘*applications.conf*’ file.
- *fileServer*: folder containing the utility service that handles the file library. It handles upload from remote users and start SAGE viewing applications using the *appLauncher*.

SAGE Directories

- */fonts*: font files used by SAGE in the user interface (on the wall)
- */images*: images used by SAGE for user-interface elements
- */include*: headers to write applications
- */lib64*: SAGE libraries
 - *libquanta*: network library
 - *libsail*: SAGE application interface library (used by SAGE applications)
 - *libsac*: Streaming Audio Client library for audio streaming
- */resources*: initial set of sample media files used at installation
- */ui*: python user interface application for administration
- */sageConfig*: folder containing various SAGE configuration files

fsManager.conf is the main SAGE configuration file for the *fsManager* process. The main parameters are as follows, though more parameters exists for advanced users or are deprecated.

- *fsManager*: IP address of the head node
- *systemPort*: SAGEcommunication port
- *uiPort*: port used by various user interface components
- *conManager*: [optional] IP address to register your SAGE session. Used by various user-interface components to discover active SAGE sessions
- *tileConfiguration*: file describing the wall layout and the mapping from machines to displays
- *fullScreen*: 0 is not windowed mode, 1 is fullscreen. Use value 2 if value 1 is not effective (dependent on the window manager)
- *receiverSyncPort*: port used for the synchronization tasks
- *receiverStreamPort*: port using for pixel streaming

stdtile.conf: tiling configuration file containing the wall layout. The file name is specified in *fsManager.conf*.

- **global section TileDisplay**: describes the dimension of the wall in number of tiles or screens. Mullions is the size in inches of the borders of the screens; used with the PPI value (pixel per inch), the number of pixels hidden behind the screen borders can be calculated. All screens comprising the wall must have uniform resolution. The number of machines driving the system must be specified. One ‘sageDisplayManager’ will be launched for each machine.
- **DisplayNode sections**: describes the IP address and the position on the wall of the attached display. The coordinate system origin (0,0) is the lower left. Vertical coordinate (y) increases upward and horizontal coordinate (x) increases right.
DisplayNode Syntax:
 - *Name <local>*: meaningless string
 - *IP <127.0.0.1>*: IP address

The IP address can be complimented with port number and Xorg server number:
IP:port:xorg. Ports have to be unique on any given machine, but can be the same on all the machines. Example: IP 127.0.0.1:22002:0.0 (server on localhost with port 22002 using Xorg server :0.0) Monitor 1 (0,0): list of monitor coordinates attached to this machine [e.g. Monitors 3 (0,0) (1,0) (2,0)]

An optional position offset can be added to move a borderless window on a large desktop (implemented using a SDL command). Pos (1920, 1080) the window moves right and up by 1920x1080 pixels.

dim.conf: configuration for DIM (Direct Interaction Manager), the UI python layer displayed on the wall.

[dim]

global section

default scale of widgets relative to display size

- *MAX_SCALE* = 8

actual scale is determined at runtime and will be between these two limits

- *MIN_SCALE* = 0.5

Pixel Per Inch, scale the widgets

- *PPI* = 50

minimize area size (height); percentage of the whole display height

- *MINIMIZE_BAR_SIZE* = 0.03

size of collapsed panels in percent of the parent section

- *COLLAPSE_RATIO* = 0.03

enable master/slave synchronized sharing (mirroring)

- *SHARED_MIRRORING* = False

display any UI element (media library, icon section, sharing buttons)(default: True)

- *SHOW_USER_INTERFACE* = True

background image (stretched to fill the wall)

- [background]

- *show* = True

- *file* = images/backgrounds/pattern.jpg

overlay logo image, centered and blended over the background

- [logo]

- *show* = True

- *file* = images/evl-logo.png

- *transparency* = 24
 - *width* = 3000
 - *height* = 1132
- logo image in the lower right corner (over the minimize region)
- [corner]
 - *show* = True
 - *file* = images/evl-logo.png
 - *transparency* = 200
 - *width* = 185
 - *height* = 70
- miscellaneous elements
- [buttons]
- display a quit button
- *quit_button* = False
- display extras application buttons, in bottom center region
- *application_buttons* = False
- show a clock at the top left
- *clock* = True
- display some text next to the clock, often used to advertise the IP address of the wall
- *label* = 131.193.77.42

applications/applications.conf file for configuring basic SAGE applications, primarily media viewing applications: mplayer, imageviewer, PDFviewer and VNCviewer. A single configuration file is generated for each instance running, based on the template found in this file.

Mplayer (movie player example)

- *configName default*: default configuration
- *Init 100 100 -1 -1*: initial size and location (which can be overloaded by DIM or the application)
- *exec 127.0.0.1 mplayer*: start on the local host (the parameters are added later by the application launcher based on *fileServer.conf*)
- *nwProtocol TCP*: protocol used, TCP or UDP
- *masterIP 127.0.0.1*: master address, only used for parallel application
- *pixelBlockSize 128 128*: default pixel block size for streaming
- *staticApp*: behavior of the application (can be removed if application is always streaming)
- *sync SAGE_BLOCK_HARD_SYNC*: synchronization mode, values are *SAGE_BLOCK_HARD_SYNC*, *SAGE_BLOCK_NO_SYNC* or *SAGE_BLOCK_SOFT_SYNC*

applications/myapp.conf configuration file for each application (required). The name is declared at configuration time in the SAGE API. Most of the parameters can be overridden by the application at runtime, or omitted, in which case standard values are used by default. Example:

- *fsIP 127.0.0.1*: address of machine running *fsManager*
- *fsPort 20002*: port of *fsManager*
- *masterIP 127.0.0.1*: address of the main process in parallel applications
- *msgPort 23010*: communication port
- *syncPort 13010*: synchronization port
- *nodeNum 1*: number of rendering nodes in the application
- *appId 2*: application ID for parallel application

- *pixelBlockSize* 64 64: default pixel block size (might be overloaded by automatic optimizations).
- *winX* 0: initial X window position (horizontal)
- *winY* 0: initial Y window position (vertical)
- *winWidth* 1280: initial window width (-1 can be used to let SAGE restore the window size when using a session)
- *winHeight* 720: initial window height (-1 can be used to let SAGE restore the window size when using a session)
- *nwProtocol* TCP: network protocol used, TCP or UDP
- *asyncUpdate* true: asynchronous streaming application
- *streamType* SAGE_BLOCK_HARD_SYNC: synchronization mode, values are SAGE_BLOCK_HARD_SYNC, SAGE_BLOCK_NO_SYNC or SAGE_BLOCK_SOFT_SYNC
- *frameRate* 30: maximum frame rate for the application, SAGE will block the application if it tries to stream faster. There is no effect if the rate is slower.

fileServer/fileServer.conf is the file server configuration for the media library. Files are organized by media types: pdf, images, movies, playlist, 3D images, Office, and audio files. The media library maintains a list of thumbnails for the assets in the thumbnail folder. Images are converted internally to the DXT format for speed (.dxt extension).

FILES_DIR = fileLibrary: location of the user's file library; by default \$HOME/.sageConfig/fileServer, but can be put anywhere on the file system of the master node by specifying an absolute path (starting with a "/")

Example:

- *FILES_DIR* = /data/sage/fileServer/fileLibrary: files are in \$HOME/.sageConfig/fileServer/fileLibrary

/dim: Direct Interaction Manager for SAGE: user interface visible onto the wall display

6. SAGE for Developers

SAGE is open-source available for free with the BSD license. Source code (read-only access) can be downloaded using any subversion client <svn://iridium.evl.uic/sage2012/>.

The main branch is accessible with the command (source code is put into ‘sage-dev’):

```
svn checkout svn://iridium.evl.uic.edu/sage2012/trunk sage-dev
```

6.1.1. SAGE Structure

The basic SAGE structure is described in the previous section. A SAGE source tree separate from the installation directory is recommended for developers modifying SAGE code or writing applications. For example, source code could be installed in *\$HOME/Dev/sage-dev* and the installation in *\$HOME/sage*. In this case, the *SAGE_DIRECTORY* variable should be set to *\$HOME/sage*. After code modification, the user should do a ‘make install’.

To compile SAGE (inside the SAGE source directory structure):

- `./configure` (only the first time after a ‘checkout’)
the configure file is a shell script setting up the applications and compiling dependencies
 - `make -j 8` (compile in parallel on multi-core machines)
 - `make build` (build inside the folder)
 - `make install`
the install location is setup by the *PREFIX* variable, which could be passed on the command-line to make: `make install PREFIX=$HOME/toto` or directly edited inside the file *Makefile*.

If compilation options need to be edited for a specific platform, the *config.mk* file is edited to specify options. The variable *MACHINE* and *ARCHITECTURE* help structure the process:

- **MACHINE:** values
 - Darwin*: for Mac OS X
 - Linux*: for linux
 - Solaris*: for SunOS Solaris
- **ARCHITECTURE:** values
 - i386*: on Mac OS X
 - i686*: on Linux 32bit
 - x86_64*: on Linux 64bit
 - ia64*: on Linux Itanium 64bit
 - ppc64*: on Linux PPC PS3, etc

6.1.2. Libraries

SAGE builds three main libraries:

- `libsail` for the ‘SAGE application interface library’ used to stream pixels.
- `libsac` for ‘streaming application client’ used for streaming audio to SAM (Streaming Audio Manager).
- QUANTA network library.

6.1.3. Dependencies

For the latest Linux OS distribution, SAGE needs the following packages to be installed as part of the OS: `make`, `gcc-c++`, `libSDL-devel`, `readline-devel`, `freeglut-devel`, `ImageMagick-devel`, `libjpeg62-devel`, `glew-devel`, `libSDL_ttf-devel`, `libpoppler-devel`, `LibVNCServer-devel`, `libqt4-devel`, `yasm`, `libbz2-devel`, `libjpeg-turbo`

At runtime, the following packages are needed: libjpeg62, libSDL-1_2-0, libreadline6, libglut3, ImageMagick, glew, libSDL_ttf-2_0-0, python-wxWidgets, python-numpy, libpoppler25, python-pygame, LibVNCServer, libqt4, libjpeg-turbo

SAGE uses OpenGL as a drawing API through the SDL library and GLEW library (for OpenGL Shading Language GLSL). GLUT is used by some applications. ImageMagick is used to load a common image format. Fonts are managed with TrueType Font. PDF rendering is handled by the Poppler library. Desktop sharing uses the LibVNCClient library. SAM use the Qt toolkit. Mplayer movie player depends on various libraries, and the YASM assembly compiler. Other libraries are already installed in most common Linux distribution. Windows and Mac OS X might require a few extra libraries commonly available; ‘Brew’ a package manager for Mac OS X is a useful tool to install commonly used packages.

SAGE includes a copy of SAM (current version 0.4) to handle application audio needs. Developed by the Sonic Arts R&D Group at Calit2 UCSD, SAM can be downloaded from <<https://code.google.com/p/streaming-audio/>>. Source code for both SAM and a SAM client library used to develop new SAM clients are included. SAM currently depends on Qt 4 and the JACK Audio Connection Kit audio API, and is currently only being tested for Linux and Mac OS X. SAM clients can run on Linux, Mac, or Windows systems with or without a sound card. SAM receives uncompressed audio via RTP streams from an arbitrary number of clients over high-performance wide-area networks. Control data is exchanged via SAM and clients using Open Sound Control (OSC). OSC is also used to communicate with third-party user interfaces. Advanced audio rendering modules (spatialization, etc.) can be added to SAM using a rendering interface.

6.1.4. Folder Structure

The SAGE code is in the src folder. Some networking code is bundled in the QUANTA folder building the *libquanta* library. All the applications reside in the app folder (one folder per app). Each application is composed of at least one C++ source file, one *Makefile*, and one application configuration file.

When writing a specific application, copy an existing application *Makefile* (e.g. *atlantis/Makefile*) to edit. Add the new application to the *subprojects* list in the main *Makefile* to automatically compile by the build process. If the new application is called *my_new_app*, add *app/my_new_app* to the list.

6.2. Files

SAGE core is comprised of two main binaries (*fsManager* and *sageDisplayManager*), one library, and a few utilities. In the ‘src’ folder, the distribution is as follows:

***fsManager*:** misc.cpp sageMessage.cpp sageRect.cpp streamInfo.cpp sageBlock.cpp sageBuf.cpp sageBlockPool.cpp sageFrame.cpp sageBlockPartition.cpp tinyxml.cpp tinyxmlparser.cpp tinyxmlerror.cpp tinystr.cpp fsManager.cpp fsServer.cpp fsCore.cpp sageVirtualDesktop.cpp tileConfig.cpp displayInstance.cpp fsMain.cpp

***sageDisplayManager*:** misc.cpp sageMessage.cpp sageRect.cpp streamInfo.cpp sageBlock.cpp sageBuf.cpp sageBlockPool.cpp sageFrame.cpp sageBlockPartition.cpp tinyxml.cpp tinyxmlparser.cpp tinyxmlerror.cpp tinystr.cpp sageDisplayManager.cpp fsClient.cpp sageSync.cpp sageDisplay.cpp sageTexture.cpp sageShader.cpp pixelDownloader.cpp streamProtocol.cpp sageTcpModule.cpp sageUdpModule.cpp sageReceiver.cpp sageDraw.cpp sageDrawObject.cpp overlayPointer.cpp overlayButton.cpp overlayApp.cpp overlayIcon.cpp overlayLabel.cpp overlayMenu.cpp

overlaySizer.cpp overlayPanel.cpp overlayThumbnail.cpp overlaySplitter.cpp
overlayEnduranceThumbnail.cpp sageBlockQueue.cpp sageSharedData.cpp
sageEvent.cpp sdlSingleContext.cpp font.cpp image.cpp

uiConsole (command-line access to fsManager): uiConsole.cpp fsClient.cpp suil.cpp
sageMessage.cpp misc.cpp

fsConsole (another command-line tool to access fsManager): fsConsole.cpp fsClient.cpp
suil.cpp sageMessage.cpp misc.cpp

libsail.so: misc.cpp sageMessage.cpp sageRect.cpp streamInfo.cpp sageBlock.cpp
sageBuf.cpp sageBlockPool.cpp sageFrame.cpp sageBlockPartition.cpp tinyxml.cpp
tinyxmlparser.cpp tinyxmlerror.cpp tinystr.cpp sail.cpp envInterface.cpp fsClient.cpp
sageSync.cpp streamProtocol.cpp sageTcpModule.cpp sageUdpModule.cpp
sageDoubleBuf.cpp sageConfig.cpp sageStreamer.cpp sageBlockStreamer.cpp appUI.cpp
appWidgets.cpp libsage.cpp suil.cpp

sage bridge tools: deprecated

SAM is handled from the ‘sam’ folder in the root of the source tree. **SAM** is the server running on the audio rendering machine, and *libsac* contains the API used by applications. See <<https://code.google.com/p/streaming-audio/>>.

6.3. SAGE Code Structure

Every SAGE application follows the same structure: an initialization phase describing the pixel format, the creation of widget elements, a streaming phase, and the processing of user-interface events:

- initialization
- create user interface
- while not ended
 - swap pixel buffers
 - process messages
- clean up

Various sample applications are provided in the ‘app’ directory: *render*, *atlantis*, *checker*.

Two compatible software levels can be used to program a SAGE application: a low-level API in C++, and a simpler higher-level C API. The C API (as exposed the file *libsage.h*) is described.

6.3.1. Initialization

Create and initialize a SAIL object: the SAIL object represents one instance of a SAGE application.

sail createSAIL(const char *appname, int width, int height, enum sagePixelFormat pixelfmt, const char *fsIP, int roworder = BOTTOM_TO_TOP, int frate = 60, sageWidgetFunc wFunc = NULL)*

- *appname*: application name, exposed in user-interface elements and used to load configuration file (“*appname.conf*”)
- *width*: pixel width of the application (constant during the life of the application)
- *height*: pixel height of the application (constant during the life of the application)
- *pixelfmt*: pixel format, can be: *PIXFMT_555*, *PIXFMT_555_INV*, *PIXFMT_565*, *PIXFMT_565_INV*, *PIXFMT_888*, *PIXFMT_888_INV*, *PIXFMT_8888*,

*PIXFMT_8888_INV, PIXFMT_RLE, PIXFMT_LUV, PIXFMT_DXT, PIXFMT_YUV,
PIXFMT_RGBS3D, PIXFMT_DXT5, PIXFMT_DXT5YCOCG*

- *fsIP*: IP address of *fsManager*, can be left NULL if specified in application configuration file
- *roworder*: *BOTTOM_TO_TOP* or *TOP_TO_BOTTOM_TO*, removes the need to flip the pixel buffer.
- *frate*: maximum frame rate, upper limit before SAGE limits the streaming speed (blocks the application)
- *wFunc*: user-provided function to create application widget

Example: *sail *myapp = createSAIL("myapp", 1920, 1080, PIXFMT_888, NULL)*

Blocking call that returns the wall size (return 1 in case of success and 0 in case of error)

- **int getWallSize(int &width, int &height)**

Can be called before creating the SAIL object to find out the size of the target wall (in pixels). This call uses the ‘*fsManager.conf*’ file to locate the IP address of the *fsManager*.

6.3.2. Pixel Streaming

During the streaming phase, either a pixel buffer pointer is requested, fills the buffer, and then request a buffer swap (hence streaming), or a buffer is passed to SAGE (which in turn copies its content and triggers streaming).

- *unsigned char* nextBuffer(sail *sageInf)*: return the next buffer to be filled
- *void swapBuffer(sail *sageInf)*: swap buffers and stream
- *unsigned char* swapAndNextBuffer(sail *sageInf)*: swap buffers and return the next buffer to be filled
- *void swapWithBuffer(sail *sageInf, unsigned char *pixptr)*: Fill the next buffer and swap
- *void deleteSAIL(sail *sageInf)*: disconnect and delete a SAIL object
- *void processMessages(sail *sageInf, application_update_t *up, sageQuitFunc qfunc, sageSyncFunc sfunc)*: process SAGE message from *fsManager* and DIM: the application receives a few updated messages from SAGE and some user-interaction events triggered through its user interface

The user can provide a ‘quit’ function to be called for before shutdown (to clean up memory, etc.).

A ‘sync’ function can be called when a synchronization event occurs (between multiple shared instances of one application).

Main events:

- *APP_QUIT*: when the application is closed
- *EVT_APP_SHARE*: when the application is shared with another wall
- *EVT_APP_SYNC*: synchronization event

‘application_update’ structure:

- *int updated*: application has been moved / resize
- *int app_id*: application identifier
- *int wall_width and int wall_height*: (SAGE internals)wall size
- *int app_x, app_y, app_w, app_h*: application position and size (width, height)

6.3.3. Widgets

In the *createSAIL* call, the developer can pass a function to be called to create the on-display application user interface (like buttons for movie player or PDF viewer).

- *sageWidgetFunc* of type “*typedef void(* sageWidgetFunc)(sail*)*,” (function returns void and takes a *sail** object as parameter).

Once the user-interface elements hierarchy is built, the main panel is passed to SAGE for display and event processing.

- *sageInf->addWidget(bottomPanel);*

SAGE will automatically draw the elements above or around the application window. The user does not need to draw the user interface just process the events after swapping frames.

Example: creating a few UI elements, setting up callbacks for buttons and passing the main widget to SAGE.

```
// Create a label
label *label1 = new label;
label1->setSize(150, 20);
label1->alignLabel(LEFT);
label1->setLabel("my label", 10);
label1->align(CENTER, BOTTOM_OUTSIDE);

// Create an icon button
thumbnail *pauseBtn = new thumbnail;
pauseBtn->setSize(50,50);
pauseBtn->alignY(BOTTOM, 2);
pauseBtn->setUpImage("images/pause2.png");
pauseBtn->setDownImage("images/play2.png");
pauseBtn->setToggle(true);
pauseBtn->onDown(&playpause); // down callback
pauseBtn->onUp(&playpause); // up callback
pauseBtn->setTransparency(150);

// Create an label button
button *btn1 = new button;
btn1->alignLabel(LEFT);
btn1->setSize(100,20);
btn1->setLabel("Quit", 10);
btn1->onUp(&quit);
btn1->setFontColor(255,255,0);

// sizer to align elements
s1 = new sizer(VERTICAL);
s1->addChild(btn1);
s1->addChild(pauseBtn);
s1->addChild(label1);
s1->alignY(TOP, 5);

// Group it all inside a panel
panel *bottomPanel = new panel(37,37,37);
bottomPanel->align(CENTER, BOTTOM_OUTSIDE,0,2);
bottomPanel->setSize(resX, 100);
bottomPanel->fitInWidth(false);
bottomPanel->setTransparency(150);
bottomPanel->addChild(s1);

// Pass the UI elements to SAGE
sageInf->addWidget(bottomPanel);
```

See examples of widget creating on ‘atlantis’, ‘PDFViewer’ or ‘mplayer’ applications.
Available widgets are: button, icon, label, sizer, and panel.

6.4. Application Details

atlantis: sample GLUT OpenGL application

Pixels are read from the GPU by a standard OpenGL read-back call.

- GLubyte *rgbBuffer = **nextBuffer**(sageInf);
- **glReadPixels**(0, 0, winWidth, winHeight, GL_RGB, GL_UNSIGNED_BYTE, rgbBuffer);
- **swapBuffer**(sageInf);

bplay-noglut: uncompressed animation playback (developed by NCSA/UIUC)

Reads BMV file generated by included img2bmv utility (takes a list of images).

Example: bplay-noglut -m -v -f 30 animation.bmv

- -m: monoscopic
- -v: verbose
- -f 30: 30 fps

checker: sample benchmark application (streaming a memory buffer)

Simplest SAGE application streaming a memory buffer.

Endless loop:

- **memset**(buffer, color, resX*resY*3);
- buffer = **swapAndNextBuffer**(sageInf);

decklinkcapture: blackmagic decklink HD/2K/4K video capture, with SAM audio streaming
Video format must be specified on the command line (frame resolution and frame rate).

Example: decklinkcapture -d 0 -m 10 -i 1 -v -y

capture 1080i59.94 from card 0 and SDI input with de-interlacing filter and passed to SAGE
decklinkcapture.conf to specify the SAGE parameters

Usage: decklinkcapture -d <card id> -m <mode id> [OPTIONS]

- supported video input connections (-i [input #]):
 - -d <card id>:
device 0 [DeckLink SDI]
 - -m <mode id>:
 - 3: HD 1080p 23.98 1920 x 1080 23.976 FPS
 - 4: HD 1080p 24 1920 x 1080 24 FPS
 - 6: HD 1080p 29.97 1920 x 1080 29.97 FPS
 - 7: HD 1080p 30 1920 x 1080 30 FPS
 - 9: HD 1080i 59.94 1920 x 1080 29.97 FPS
 - 10: HD 1080i 60 1920 x 1080 30 FPS
 - 12: HD 720p 59.94 1280 x 720 59.9401 FPS
 - 13: HD 720p 60 1280 x 720 60 FPS
 - -i <video input number>
 - 1: SDI, 2: HDMI, 8: Component, 16: Composite, 32: S-Video,
 - -c <channels> audio Channels (2, 8 or 16 - default is 2)
 - -s <depth> audio Sample Depth (16 or 32 - default is 32)
 - -j physical inputs for SAM inputs (audio)
 - -u <SAM IP> stream using SAM (audio), 127.0.0.1 by default
 - -v stream using SAGE (video)
 - -y apply de-interlacing filter

images3d: stereo 3D image viewer

Converts side-by-side stereo image to SAGE S3D pixel format. *Stereo3D* script converts MPO/PNS/JPS stereo formats to a side-by-side image and launches the *images3d* application.

imageviewer: basic image viewing application based on ImageMagick
Converts default images to DXT and caches the DXT file.

mplayer: video playback with SAGE and SAM plugin

SAGE display module: -vo sage

stereo 3D support: -vo sage:stereo:[option]

- *stereo*
will take a side-by-side movie (2*width) and streams it in stereo mode
default: off
- *unsquish*
expand the width to double-width
default: off
- *rightfirst*
flip the stereo
default: off
- *guessformat*
guess stereo format: full-width or half-width
default: on

SAM audio module: -ao sam

-ao sam:[options]

Example: `mplayer -ao sam:ip=xxx.xxx.xxx.xxx:port=7770`

connects MPlayer to SAM at the given IP address and port

Options:

- name=<client name>
client name to pass to JACK
- estimate
estimates the amount of data in buffers (experimental)
- ip
SAM IP address
- port
port for contacting SAM
- type
stream type (0=basic,1=tracked,2=array)
- timeout
timeout in seconds for contacting SAM

Mplayer options can be set permanently in the file `$HOME/.mplayer/config`

- ao="sam:ip=67.58.41.28:type=1"
- lavdopts=fast=1:threads=4
- delay=0

officeviewer: shell script converting Office files (doc, xls, ppt, docx, xlsx, pptx, etc) on-the-fly to PDF using LibreOffice environment. Calls ‘*ooconvert*’ application to convert the file to PDF and launches *pdfviewer* with the produced PDF file.

qshare: high-speed desktop sharing (Mac, Linux and Windows)

Requires 1Gbps link minimum for a laptop/desktop standard screen resolution, and 10Gbps link for a 4 Mpixels or 4K display. Configured by *qshare.conf*. User interface to set target *fsManager* and frame rate, which can be adjusted dynamically.

render: sample GLUT OpenGL application

samugen: basic SAM audio test application

Usage: samugen --name or -n client name

- --ip or -i SAM ip
- --port or -p SAM port
- --channels or -c number of channels to stream
- --type or -t type of SAM stream
- [--x or -x initial x position coordinate]
- [--y or -y initial y position coordinate]
- [--width or -w initial width for SAM stream]
- [--height or -h initial height for SAM stream]
- [--depth or -d initial depth for SAM stream]

Example: samugen -n "Example Client" -i "127.0.0.1" -p 7770 -c 2 -t 0

uv: Ultragrid HD/2K/4K audio and video streaming software <http://www.ultragrid.cz/en>

Usage: uv [-d <display_device>] [-t <capture_device>] [-r <audio_layout>] [-s <audio_capture>] [-l <limit_bitrate>] [-m <mtu>] [-c] [-i] [-6] [-m <video_mode>] [-p <postprocess>] [-f <fec_options>] [-p <port>] [--mcast-if <iface>] address(es)

- -d <display_device>
 - select display device
 - use '-d help' to get list of supported devices
- -t <capture_device>
 - select capture device
 - use '-t help' to get list of supported devices
- -c <cfg>
 - compress video (see '-c help')
- -r <playback_device>
 - audio playback device (see '-r help')
- -s <capture_device>
 - audio capture device (see '-s help')
- address(es)
 - destination address

Basic SAGE usage:

- sending: uv -t decklink:0:6:2vuy:connection=HDMI -c JPEG:90 -m 9000 67.58.41.5
- receiving: uv -d sage
- capture 4K, audio playback and display with SAGE: uv -t decklink:2,3,0,1:6 -m 8800 -s embedded 127.0.0.1 -d sage -M tiled-4K -r alsa
- No SAM support yet (only local playback with ALSA or through a SDI device)

VNCViewer: application used for desktop sharing

One instance of VNCViewer is launched for each shared desktop. It bridges the remote desktop to the SAGE wall. VNCViewer is launched through the SAGEPointer application, and not manually from the command line, and does not work if a NAT network connection is

used. VNCViewer requires LibVNCServer library compiled with password encryption support (gcrypt library) for Mac OS X client. VNC passwords are sent through a SSL socket for more security.

Usage:

- -l: listen mode (server initiates the connection)
- -s [hostname]: machine running the VNC server (required)
- -d [display number]: X11 display number (default: 0)
- -r [frame rate]: maximum frame rate (default: 10.0)
- -p [password]: vnc password or login password with username] (required)
- -u [username]: user name (for server requiring login) (optional)
- -h: help

Examples:

- VNCViewer -s 127.0.0.1 -p mypasswd (vnc usage)
- VNCViewer -s 127.0.0.1 -u toto -p mypasswd (login usage, like Mac OS X Lion)

webcam: a trivial V4L (video for Linux) webcam capture application with no audio support

Usage: **webcam** [options]

Options:

- -d | --device name
 video device name [/dev/video0]
- -h | --help
 print this message
- -m | --mmap
 use memory mapped buffers
- -r | --read
 use read() calls
- -u | --userp
 use application allocated buffers
- -w width
- -h height

Example: **webcam** -w 640 -h 480 -d /dev/video0

Most of the video settings are retrieved from the device and must be set a priority.

- v4l2-ctl --set-fmt-video=width=1280,height=720,pixelformat=1
- v4l2-ctl --verbose --set-ctrl=exposure_auto_priority=0
- v4l2-ctl --verbose --set-ctrl=exposure_auto=3
- v4l2-ctl --set-ctrl=white_balance_temperature_auto=0
- v4l2-ctl --set-ctrl=exposure_auto_priority=0
- v4l2-ctl --set-ctrl=backlight_compensation=0
- v4l2-ctl --set-parm=30

or use a graphical tool such as '**qv4l2**'

6.5. Advanced Topics

The DIM (Direct Interaction Manager) comprises all the UI elements seen in SAGE (excluding application elements) - the wall partition and multi-touch systems, the pointers, the file library access, and the sharing of content.

Not documented at this time.

7. Troubleshooting/Technical Information

7.1. Hardware

7.1.1. OpenGL Hardware Driver

Verify the correct driver is installed for the graphics card. NVIDIA graphics cards with the NVIDIA driver are recommended (the open-source ‘nouveau’ driver might work, but it is not tested with SAGE). If needed, set the shell environment variable to the correct value (`export DISPLAY=:0.0` or `setenv DISPLAY :0.0` depending on the shell). If connecting via ssh, make sure the locally connected user (driving the screen) allows connection to the Xorg server. To allow this, the connected user should do: `xhost +local:` to enable local connection.

To validate the installation, a few commands are available:

- `nvidia-settings -q gpus -q screens`
lists screens and gpus
- `nvidia-xconfig --query-gpu-info`
lists the connected displays
- `glewinfo | grep 'OpenGL\|Running'`
returns GPU and driver versions
- `xwininfo -root`
provides the current screen resolution

7.1.2. Video Capture Hardware Driver

Verify the correct driver for the HD/2K/4K capture card is installed. The Blackmagic Design DeckLink card series is recommended. A consistent Linux driver is provided.

Make sure the driver is loaded:

- `lsmod | grep blackmagic`
- `blackmagic` 491070 0
- `dmesg | grep -i blackmagic`
- `blackmagic: Loading driver (version: 9.7.3a5)`

If the driver is not loaded, reboot.

To validate the installation, validate the firmware installed:

- `/usr/bin/BlackmagicFirmwareUpdater status`
- `blackmagic0 [DeckLink SDI] 0x1d OK`

If the upgrade the firmware command is returned (common after a driver update), run:

- `/usr/bin/BlackmagicFirmwareUpdater update 0`

7.1.3. Network Driver

Make sure the 10Gbps Ethernet driver is installed properly (e.g. Myrinet or Intel driver). Run ‘`nuttcp`’ or ‘`iperf`’ benchmarks to validate the bandwidth between local machines; over 9Gbps connectivity should be expected with either TCP or UDP, using modern hardware.

7.2. Software

7.2.1. NFS Setup

It is highly recommended that all SAGE nodes share the user directories using a single shared filesystem. The most common on Linux is NFS. See the distribution documentation for set up.

7.2.2. SSH keys

Since SAGE uses ssh to connect to all cluster nodes and execute applications, ssh keys must be setup for the user running SAGE. If the cluster is set up using NFS, the home

directories will already be the same on all the nodes and require nothing further. If not, the public SSH key will need to be copied from the master to all nodes. Copy the key from `~/.ssh/id_dsa.pub` (or `id_rsa.pub` depending on the encryption being used) to all nodes `~/.ssh/authorized_keys`. The master's key will be checked against the keys in the node's `authorized_keys` file, and if found it will login without asking for a password.

To validate the installation, connect once to ALL the machines comprising in the SAGE system (master node, display nodes, application nodes, video capture node, etc):

- `ssh 127.0.0.1`
- `ssh localhost`
- `ssh [my real hostname]` or `ssh [my real IP address]`
- `ssh [IP display1]` and `ssh [IP display2]` ...

7.2.3. Linux Notes

It is critical to have Linux configured correctly in order for SAGE to launch and operate properly. Some items for consideration follow.

- Disable the firewall on OpenSuse:

```
chkconfig SuSEfirewall2_init off  
chkconfig SuSEfirewall2_setup off  
rcSuSEfirewall2 stop
```

- Make sure the hostname is setup correctly when setting up the network; SAGE requires properly configured hostname and hosts file.

- Configure `.xinitrc`

```
cp ~.xinitrc.template ~.xinitrc
```

edit `~.xinitrc` and add the following lines to the section that says add your lines here:

```
xhost +local:  
xset -dpms  
xset s off  
xset s noblank
```

- Configuring X

Use `nvidia-xconfig --query-gpu-info` to get the BusID of the graphics card.

In either `/etc/X11/xorg.conf` or `/etc/X11/xorg.conf.d/50-device.conf`, configure the card/s.

Example of 4 outputs from a single GTX690:

```
Section "Device"
    Identifier      "Device0"
    Driver          "nvidia"
    VendorName     "NVIDIA Corporation"
    BoardName       "GeForce"
    BusID           "PCI:5:0:0"
    Option          "TripleBuffer" "1"
    Screen          0
EndSection

Section "Device"
    Identifier      "Device1"
    Driver          "nvidia"
    VendorName     "NVIDIA Corporation"
    BoardName       "GeForce"
    BusID           "PCI:5:0:0"
    Option          "TripleBuffer" "1"
    Screen          1
EndSection

Section "Device"
    Identifier      "Device2"
    Driver          "nvidia"
    VendorName     "NVIDIA Corporation"
    BoardName       "GeForce"
    BusID           "PCI:5:0:0"
    Option          "TripleBuffer" "1"
    Screen          2
EndSection

Section "Device"
    Identifier      "Device3"
    Driver          "nvidia"
    VendorName     "NVIDIA Corporation"
    BoardName       "GeForce"
    Option          "TripleBuffer" "1"
    BusID           "PCI:6:0:0"
    Screen          0
EndSection

Section "ServerLayout"
    Identifier      "Layout0"
    Screen          0 "Screen0"
    Screen          1 "Screen1" RightOf "Screen0"
    Screen          2 "Screen2" Below "Screen1"
    Screen          3 "Screen3" RightOf "Screen2"
    Option          "Xinerama" "0"
EndSection

Section "Extensions"
    Option          "Composite" "Disable"
EndSection
```

Also, edit /etc/X11/xorg.conf or /etc/X11/xorg.conf.d/50-screen.conf configuring the screens:

```
Section "Screen"
Identifier      "Screen0"
    Device        "Device0"
    Monitor       "Monitor0"
    DefaultDepth  24
    Option        "Stereo" "0"
        Option     "metamodes" "GPU-0.DFP-0: nvidia-auto-select +0+0"
    SubSection    "Display"
        Depth      24
    EndSubSection
EndSection

Section "Screen"
Identifier      "Screen1"
    Device        "Device1"
    Monitor       "Monitor0"
    DefaultDepth  24
    Option        "Stereo" "0"
        Option     "metamodes" "GPU-0.DFP-1: nvidia-auto-select +0+0"
    SubSection    "Display"
        Depth      24
    EndSubSection
EndSection

Section "Screen"
Identifier      "Screen2"
    Device        "Device2"
    Monitor       "Monitor0"
    DefaultDepth  24
    Option        "Stereo" "0"
        Option     "metamodes" "GPU-0.DFP-2: nvidia-auto-select +0+0"
    SubSection    "Display"
        Depth      24
    EndSubSection
EndSection

Section "Screen"
Identifier      "Screen3"
    Device        "Device3"
    Monitor       "Monitor0"
    DefaultDepth  24
    Option        "metamodes" "GPU-1.DFP-0: nvidia-auto-select +0+0"
    SubSection    "Display"
        Depth      24
    EndSubSection
EndSection
```

7.3. FAQs (Frequently Asked Questions)

7.3.1. Compiling SAGE

Where do I download the prerequisites for installing SAGE?

- SDL <<http://www.libsdl.org/download-1.2.php>>
- readline included with most Linux distributions or download from <<http://tiswww.case.edu/php/chet/readline/rltop.html#Availability>>
- freetype2 may be included with Linux distribution or download from <<http://download.savannah.gnu.org/releases/freetype/>>

Compilation error similar to: “undefined reference to `glEnable’”

OpenGL is required for SAGE to compile and run; this error probably means the compiler cannot find OpenGL if it is installed on the system.

Compiler complaining about “sdl-config”.

Version 1.2.8 or newer of libSDL must be installed.

I have all the required libraries installed but I still get the above error(s).

Makefiles make use of command “uname -p” in order to identify the system to install SAGE. It is necessary in order to look for the libraries in correct places. If all the prerequisites are installed and errors are still encountered, check the system supports “uname -p” command. If not, try changing to “uname -m” in the Makefiles. If the machine type is not recognized the Makefile will assume the underlying architecture is a 32-bit Linux system.

I see the following during compilation: “Try ‘uname –help’ for more information.”

The system doesn’t support “uname -p” command. See above for more info.

Compilation error similar to: “error: GL/glut.h: No such file or directory ...”

GLUT is actually not needed for SAGE to compile, but is needed for some test applications (render and atlantis). Make sure GLUT is installed.

Compiler complains about “readline”

Try adding “-lcurses” to READLINE_LIB in the sage/src/Makefile. If that doesn’t work, try adding “-ncurses” instead of “-lcurses”.

7.3.2. Running SAGE

“fsManager: cannot find the environment variable SAGE_DIRECTORY”

The environment variable SAGE_DIRECTORY must be set while running SAGE, and has to point to the directory where SAGE is installed. Apart from this the PATH env variable has to include “\$SAGE_DIRECTORY/bin” and the LD_LIBRARY_PATH variable has to include “\$SAGE_DIRECTORY/lib”. All of these env variables must be set on ALL of the cluster nodes as well.

“QUANTAnet_tcpClient_c::Connect: Can’t connect to server. fsManager : fail to connect to the connection manager”

There is a server that handles connections between users and SAGE called the connection manager. Whenever SAGE is initiated, it informs the connection manager it is available for users to connect to it. This error message means SAGE cannot connect to the connection manager whose location was specified in fsManager.conf. Change the “conManager” line to specify a different address of the connection manager. By default, SAGE will try to connect to a connection manager always running at EVL (131.193.48.235). However, SAGE will

continue run without problem even if not connected to a connection manager. See below FAQs on how to run a connection manager.

“QUANTAnet_tcpClient_c::Connect: Can’t connect to server. fail to send the message!”

This error is most likely caused by a wrong configuration file - probably the fsManager.conf. The first line must be setup properly: “sage_name”“local_ip”“public_ip” OR “sage_name”“public_ip”. For more detail look into Quick Start.

“QUANTAnet_tcpServer_c::init: Can’t bind local address”

This could mean one of two things: fsManager is already running (or not properly shutdown), or an application is still left running from the previous execution of SAGE. To stop fsManager, run KILL_ALL from the sage/bin directory. This should quit SAGE and some applications from the master and all the nodes. If an application is still running and the KILL_ALL script did not resolve, the application will need to be closed manually. To review applications still running, check all the ssh connections that are still open: “ps aux | grep ssh”. One or more of them will correspond to a SAGE application (they will have a “gStreamRcv” string in them). Kill that application and recheck all the ssh connections. Those applications pertaining to SAGE should be gone.

While running fsManager: “Segmentation fault”

A number of causes are possible: fsManager, gStreamRcv, sage applications... One possibility is that something SAGE related is still running from the previous session (refer to the following FAQ - **“QUANTAnet_tcpServer_c::init: Can’t bind local address”**). If the message is preceded by “failed to open sail config file”, refer to FAQ **When starting fsManager: “fail to open sail config file”**. If the segmentation fault seems to be coming from a SAGE application, check write permissions to the directory where the application binary resides. It could be just a bug, which will require killing SAGE, all applications and restarting.

I see window borders on my display nodes. How do I run SAGE in fullscreen mode?

In the fsManager.conf file (towards the bottom), line “fullScreen 0”, change 0 to 1 to yield fullscreen mode. Value 2 forces fullscreen mode on most window manager, if value 1 does not work.

When starting fsManager the display nodes seem to turn black really slowly or never turn black.

First check if something really CPU intensive is running on the master or the nodes. If not, the most likely cause are remnants from a previous SAGE session (probably an application still running or not killed completely). Refer to FAQ **“QUANTAnet_tcpServer_c::init: Can’t bind local address”**.

When starting fsManager: “fail to open sail config file”...

This will occur when fsManager or a SAGE application cannot find a configuration file. The most common cause is an improperly set up PATH variable (it should include \$SAGE_DIRECTORY/bin). A less common cause are the directory and file permissions where the application and SAGE are running. Also, verify all nodes have are set up properly.

7.3.3. Running SAGE Applications

While running *fsConsole*: “>>> Opening log file... Segmentation fault”

This is caused by file/directory permissions. Permission to write to the directory where the application and the *fsConsole* resides is required.

“*render*: No such file or directory” or similar

Make sure “*render*” binary exists in sage/bin directory and the PATH environment variable contains \$SAGE_DIRECTORY/bin.

The SAGE window of my application is black or it is not being updated. I only see the borders or an old frame inside the border.

This occurs when an application is not sending its pixels to SAGE. Some applications, including OpenGL applications that use GLUT, will not render their pixels if their window is not visible. Therefore, these applications must be run on the master node, otherwise the application will be hidden below the SAGE window and not able to render its pixels.

When starting applications: “fail to open sail config file”...

Refer to FAQ **When starting *fsManager*: “fail to open sail config file”**.

One application seems to run OK, but when I try to run more at the same time I get repeated errors similar to this one: “tvSyncClient :: checkSync(): Error receiving sync from master: Success”.

This could easily occur if the machine does not have a lot of free memory; SAGE will run out of memory quickly, especially if multiple applications are running simultaneously. To remedy, change “receiverBufNum” in *fsManager.conf* to a lower number (the default is 20 so try 10) to allocate less memory to each application. This parameter tells SAGE how many frames to keep in the circular buffer for each application.

7.3.4. Other Questions

How do I use SAGE without having to type my password every time SAGE or a SAGE application is started?

Since SAGE uses ssh to connect to all cluster nodes and execute applications, ssh itself requires a password each time a node is accessed by SAGE (app execution, SAGE startup...). This is generally not good for SAGE and it should be avoided, if possible. It is simple to set up ssh keys so a password is not required each time SAGE or a SAGE application is launched. If the cluster is set up using nfs, the home directories will already be the same on all the nodes and require nothing further. If not, the public ssh key will need to be copied from the master to all nodes. Copy the key from ~/.ssh/id_dsa.pub (or id_rsa.pub depending on the encryption being used) to all nodes ~/.ssh/authorized_keys. The master’s key will be checked against the keys in the node’s authorized_keys file, and if found it will login without asking for a password.

What are the prerequisites for running SAGE UI?

The following must be installed on any platform (Windows, Mac, Linux):

- Python (2.3 or later)
- wxPython (2.6.1 or later) – get prebuilt binaries from <<http://www.wxpython.org/download.php#binaries>>, or compile from source at <http://sourceforge.net/projects/wxpython/files/wxPython/2.6.2.1/wxPython-src-2.6.2.1.tar.gz/download?use_mirror=iweb>
- numarray (1.1 or later) – get from <<http://sourceforge.net/projects/numpy/files/OldFiles/>>

How do I run/set up my own connection manager/users server?

The default connection manager is set to EVL. To set up a custom connection manager refer to the sage/ui/usersServer/README for details. The fsManager.conf (conManager line) will need to be modified to tell SAGE to connect to a connection manager other than the default. Also, when starting SAGE an IP address of the connection manager to connect to must be specified: python sageui.py 123.123.123.123.

How do I run/install SAGE UI?

SAGE UI is included with SAGE and it's present in the sage/ui directory. Since the UI is written in python which is interpreted, there is no need to compile it. It will run as is on any Windows, Mac or Linux machine. To run it type "python sageui.py [conManager IP]" from the sage/ui directory. The optional parameter in square brackets is the IP address of the machine where the connection manager is running. If a connection manager other than the default is setup, specify its IP address at the end. For more information look in sage/ui/ README.

While trying to start SAGE UI I get the following error: "ImportError: No module named ..."

This means that python cannot find the required packages for running the SAGE UI. Make sure they are installed and present in the site-packages folder of the python installation. To confirm those packages can be found by python, start the interpreter by typing "python". In the interpreter, type: "import sys; sys.path". This will list all the paths available to python, and the paths where wxPython and numarray are installed. If wxPython and numarray are not listed, the wxPython and numarray folders can be moved to the site-packages folder (always accessible by python) or added to the PYTHONPATH environment variable. Also, for wxPython make sure wx.pth file in site-packages folder contains the folder name of your wxPython installation (such as "wx-2.6-gtk2-ansi"). An error can also result if multiple versions of python are installed, and some have these packages installed others not.

How do I install wxPython on Linux/Windows/Mac OS (before Mac OS X v10.4)?

To compile wxPython from source, follow the instructions found at <<http://www.wxpython.org/INSTALL.html>>. To install from a binary, double-click on the installer or "rpm -i" on Linux. Consult the wxPython download page at <<http://www.wxpython.org/download.php#binaries>> for further detail.

How do I install wxPython on Mac OS X 10.4?

Download TigerPython23Compat.pkg from pythonmac.org/packages and wxPython 2.6 from <http://sourceforge.net/projects/wxpython/files/wxPython/2.6.2.1/wxPython2.6-osx-ansi-2.6.2.1-Mac OS X10.3-py2.3.dmg/download?use_mirror=iweb>. Install both. Rename the file/Library/Python/2.3/site-packages/Extras.pth to Extras.pth.bak.

How do I install numarray?

Download numarray from <<http://sourceforge.net/projects/numpy/files/OldFiles/>>. For windows just run the binary. For Linux and MacOS uncompress it, go to that folder where you uncompressed it and type "sudo python setup.py install". Root permissions are required for installation.

While trying to launch SAGE I get the following error: "fsServer::checkClients() : Can't read a message from SDM 7. Connection was terminated..."

This is not a SAGE error, but likely a systems error such as a bad cable or inconsistent displayPort and HDMI EDID response. One of the displays may not be active, which

can go unnoticed if a black background is being used. If the system is not fully functional, or configured properly, SAGE will not start.

8. Appendices

8.1. Audio

The SAM website <<https://code.google.com/p/streaming-audio/>> expressly provides more detail on audio streaming.



DeckLinkcapture and Mplayer support audio streaming using SAM.

8.2. Room Management and Control

The **sabi.js** <<http://renambot.lakephoto.org/software/sabi-js/>> framework is used for room management and control, such as launching SAGE, starting applications (e.g. video capture), turning displays on/off, etc.

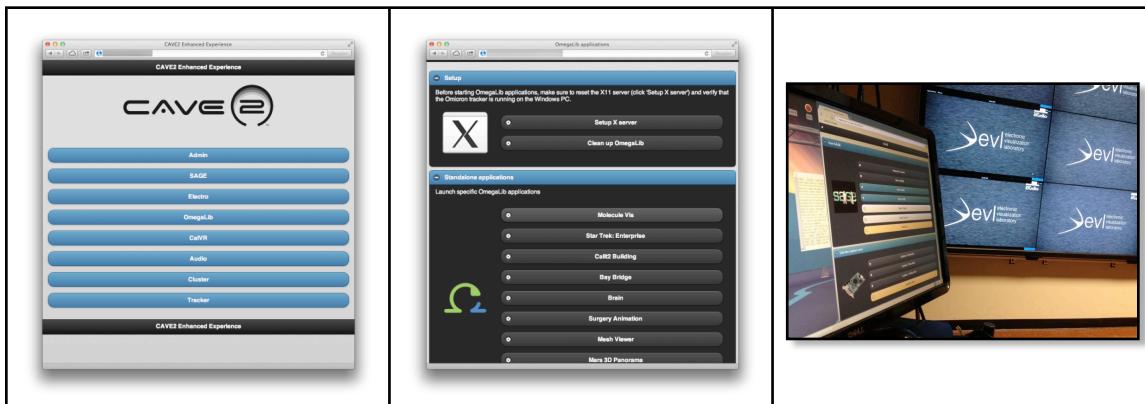
sabi.js controls devices through a HTML5/Node/Javascript framework, running commands locally, and scripts locally or on a remote server. It sends OSC messages and serial port commands to a network device.

Initial release:

- sabi.js v1_0 - sabi.js initial release - Version: 1.0
- Open-source BSD

Software dependencies:

- node.js and various packages
- client side: web packages (socket.io, jquery mobile, etc.)



8.3. UltraGrid Software

Software for low latency and high-quality video network transmissions <<http://www.ultragrid.cz/en>>.

- Specification <<http://www.ultragrid.cz/en/specification>>
- Installation <<http://www.ultragrid.cz/en/installation-and-running>>
- Wiki <https://www.sitola.cz/igrid/index.php/Main_Page>

