SLAB User Manual

Joel D. Miller

SLAB is a software-based real-time virtual acoustic environment (VAE) rendering system being developed in the Spatial Auditory Displays Lab at NASA Ames Research Center.

Note: The SLAB User Release is a work-in-progress. Functionality described herein is subject to change. This document will be revised frequently.

Contents

- Introduction
  - The User Release
  - System Requirements
  - Installation
  - Citing SLAB
  - Acknowledgements
- Overviews
  - SLAB Render API
  - SLABWire
  - SLABTrak
  - Installer
  - Coordinate System
  - Sound Sources
  - HRTF Databases
  - Error Handling
  - Tips
- Applications and Utilities

    - SLABReg
    - SLABScape
    - SLABSound
    - SLABWireDemo
    - SLABLatency
    - SLABX
- SLABScape Help Page
- SLAB Render Plug-Ins
    - Overview
    - Starter Project
    - Spatial
- slabtools
- Appendix
    - Glossary
    - Known Issues

See also: SRAPI Reference Manual

SLAB User Manual v5.4.1
Copyright 2001-2004 U.S. Government as represented by the
Administrator of the National Aeronautics and Space Administration.

Initiated: July 5, 2000
Last Updated: May 20, 2004

Introduction

- The User Release
- System Requirements
- Installation
- Citing SLAB
- Acknowledgements

The User Release

The SLAB User Release is a collection of applications, libraries, documentation, MATLAB utilities, and source code for rendering and studying virtual acoustic environments. How to begin depends on your interest:

- just curious about 3D-sound - see SLABScape
- rendering your own HRTFs - see slabtools and SLABScape
- writing experiment software - see SRAPI, Overviews, and SLABX
- writing virtual environment software - see SRAPI and SLABScape
- writing your own rendering algorithms - see Render Plug-Ins and SLABScape
- general purpose real-time audio signal processing - see SLABWire and SLABWireDemo
- frequency-driven OpenGL displays - see SLABWireDemo

Top

System Requirements

Software

The software components listed below are recommended for running and developing SLAB applications.

Operating System: Windows2000
SLAB is currently developed under Windows2000. In theory, SLAB should be compatible with Windows98SE/ME, but SLAB is no longer tested under these operating systems.

Low-latency Audio Interface: ASIO or DirectSound 8.1 (or later)

ASIO: ASIO latencies can be as low as 5ms. If you plan on rebuilding SLABWire, see the SLAB readme file <SLAB Install Dir>\lib\src\slabw\asio\readme.txt for instructions on acquiring the ASIO 2.0 SDK.

DirectSound: SLAB can use DirectSound to achieve low-latency audio output (e.g., ~24ms). The sound card used with SLAB must have a non-emulated DirectSound driver installed. To run the SLABScape application, the DirectSound Runtime should be installed. To develop SLAB

applications, the DirectX SDK should be installed. If you have a previous DirectSound version installed, it might not be necessary to upgrade. First, give SLAB a try; if you encounter problems, you might want to upgrade. DirectSound 6 Runtime is installed with Windows98SE. The Microsoft DirectX download site is: http://www.microsoft.com/directx/. Developer information is available at: http://msdn.microsoft.com/directx/.

Note regarding DirectSound latency under Windows2000 vs Windows98SE:
Under Windows2000, the DirectSound driver accesses the hardware through a WDM driver. Some WDM drivers use the Windows KMixer which adds approximately 20ms of latency to sound output. Windows98SE WDM drivers also appear to have this additional latency (versus VXD drivers). SLAB's total internal latency is approximately equal to the sum of the DirectSound buffer size, the smooth-time value, and the KMixer latency (for WDM drivers).

Compiler (optional): Microsoft Visual C++ .NET
A compiler is necessary for SLAB application development. SLAB is being developed using Microsoft Visual C++ .NET. The SLAB header files and libraries have not been tested under other development environments. Missing library errors have been noticed when linking with the SLAB libraries using Microsoft Visual C++ 6.0. Thus, the SLAB libraries may not be compatible with pre-.NET versions of Microsoft Visual C++.

Hardware

The hardware components listed below have been used successfully with SLAB.

Workstation

Computer:
Minimum recommended requirements: Intel 650MHz PIII CPU, 256MB RAM.

ASIO Sound Device:
Windows2000: MOTU 828, Delta66.

DirectSound Sound Device:
Windows2000: Creative Audigy
Windows98SE: Turtle Beach Montego II Sound Card (Montego II digital output is not supported under Windows2000)

Digital Output

DirectSound Digital Output Device:
Windows2000: Creative Audigy
Windows98SE: Turtle Beach Montego II Digital I/O Upgrade

Digital-to-Analog Converter:
Lucid Technology ADA1000 A/D, D/A Converter

Headphone Display

Headphone Amplifier:
Symetrix SX204 Headphone Amplifier

Headphones:

Sennheiser HD 545 Reference Headphones

Top

Installation

SLAB User Release Installation Instructions

SLAB is available for download at the SLAB Home Page: http://human-factors.arc.nasa.gov/SLAB.

For installation instructions, see the *SLAB User Release Installation Instructions* under the *Downloads* section of the SLAB Home Page.

Windows Sound Schemes

When listening to SLAB with headphones it is probably best to disable system sounds by selecting the "No Sounds" sound scheme under Control Panel | Sounds and Multimedia | Sounds | Scheme. Otherwise, an uncomfortably loud system sound might occur while listening to SLAB.

Speakers Property

Some sound peripherals process the output signal based on the type of display attached to the output (e.g., headphones versus desktop speakers). When using SLAB, it is best to select an unprocessed output path under Control Panel | Sounds and Multimedia | Audio | Advanced | Speakers | Speaker Setup.

Top

Citing SLAB

As a courtesy, the use of SLAB in published research should be acknowledged in the publication by citing the SLAB Home Page:

[1] http://human-factors.arc.nasa.gov/SLAB

The preferred paper reference describing the SLAB User Release and its implementation:

[2] Miller, J. D. and Wenzel, E. M., "Recent Developments in SLAB: A Software-Based System for Interactive Spatial Sound Synthesis," Proceedings of the International Conference on Auditory Display, ICAD 2002, Kyoto, Japan, pp. 403-408, 2002.

Top

Acknowledgements

I would like to thank Beth Wenzel for her support, feedback, and for making SLAB possible, Jonathan Abel for providing numerous physical modeling and signal processing examples, Mark Anderson for programming assistance, and both Mark Anderson and Durand Begault for testing the waters and offering many helpful suggestions. I would also like to thank Marlene Hernan, Robert Padilla, and Robin Orans.

--joel

Developers

Joel Miller	lead designer and programmer
Jonathan Abel	physical modeling and signal processing MATLAB scripts
Mark Anderson	TrakLib Fastrak driver, SLABScape 3D View, SLABScape SLABScript Editor, sockets and registration database assistance
Mitch Clapp	SLABScape 3D View models and textures, web-based registration code
Renee Goldschmid	Five SLABWireDemo OpenGL displays, spectrum display improvements

Top

SLAB User Manual

Last Updated: May 20, 2004

Overviews

- SLAB Render API
- SLABWire
- SLABTrak
- Installer
- Coordinate System
- Sound Sources
- HRTF Databases
- Error Handling
- Tips

SLAB Render API

The SLAB Render API (SRAPI) is used to develop virtual acoustic environment applications. SRAPI is encapsulated in the CSRAPI and CScene class interfaces.

Thread Note: Due to CSRAPI's use of MFC and Windows messaging, CSRAPI must run in a user-interface CWinThread. This is the default situation for MFC applications. MFC applications are launched from a CWinApp-derived class; CWinApp is derived from CWinThread.

Source Code: The source code for SRAPI is in <install dir>\lib\src\srapi\. The source code for the Scene is in <install dir>\lib\src\scene\.

Further information on the SLAB Render API is available in the SRAPI Reference Manual.

Top

SLABWire

SRAPI uses the SLABWire API to configure its sound_in-to-DSP-to-sound_out signal processing chain. A user's manual does not exist for SLABWire. The SLABWireDemo application demonstrates the use of the SLABWire API.

Source Code: The source code for SLABWire is in <install dir>\lib\src\slabw\. The source code for SLABWireDemo is in <install dir>\src\SLABWireDemo\.

Top

SLABTrak

SLABTrak is a Polhemus Fastrak driver. Docs are available in <install dir>\doc\slabtrak\index.html. An example application is available in <install dir>\lib\src\SLABTrak\TrakConDemo. SLABScape uses SLABTrak to access the Fastrak.

Source Code: <install dir>\lib\src\SLABTrak\

Top

Installer

<install dir>\src\Installer\ contains the Visual Studio .NET Setup and Deployment project for the SLAB User Release. This project creates an MSI Windows Installer file. The Installer project can be used to distribute works based on SLAB. See the SLAB Software Usage Agreement for the legal terms governing SLAB-related distribution.

Top

Coordinate System

SLAB uses a right-handed, FLT (Front-Left-Top) coordinate system (i.e., if fingers curled from front to left, thumb points to top).

Location
+x front, through nose
+y left, through left ear
+z top, through top of head

Orientation
-yaw to right
+yaw to left
-pitch up
+pitch down
+roll right
-roll left

Polar
+azimuth to right
-azimuth to left
+elevation up
-elevation down
+range forward
-range backward

Convolvotron and Polhemus azimuth and elevation

These definitions are compatible with the Polhemus Isotrak and Fastrak head trackers and the Crystal River Engineering Convolvotron coordinate systems with the following exceptions:

SLAB azimuth = - Convolvotron and Polhemus azimuth
SLAB and Convolvotron elevation = - Polhemus elevation

Polhemus transmitter and receiver

The physical placement of the Polhemus transmitter and receiver often results in the following configuration (as specified on the Fastrak transmitter): +X forward, +Y right, +Z down. Since SLAB uses +Z up and +Y left, the signs of Y, Z, Yaw, and Pitch must be changed if your tracker driver follows this convention.

Top

Sound Sources

SLAB supports five sound source types:

- SrcASIO = ASIO sound device
- SrcWave = Waveform-Audio sound device
- SrcFile = Windows wave file
- SrcMemory = Windows wave file read into memory
- SrcSigGen = software signal generation

All sound sources must be allocated before calling RenderStart() to initiate real-time rendering. A sound source is given an ID value when it is allocated. This ID is passed to the source control functions. The number of sources available is limited by the computational resources of the computer.

To change the source allocation configuration, call SrcFree() to free all sound sources. SrcFree() cannot be called while rendering. Once the previous sources have been freed, a new source configuration can be allocated.

To generate periodic sound sources not supported by SrcSigGen, use SrcFile() in conjunction with a Windows wave file editing application. Since SrcFile() can loop wave files continuously, it can be used to create fixed periodic sound sources.

There are two methods of muting a sound source:

- Call SrcGain() using a dB gain value much lower than -97.0 dB (-96.3 dB is the dynamic range of 16-bit integer). Note: this does not affect the processing performed. The computational load remains the same.
- Call SrcEnable() to disable the source. The source is no longer rendered, eliminating the source entirely. Disabling a source reduces computational load.

See Also: CSRAPI Source Functions in the SRAPI Reference Manual.

Top

HRTF Databases

SLAB HRTF (Head-Related Transfer Function) databases contain the HRIR (Head-Related Impulse Response) and ITD (Interaural Time Delay) information needed to perform spatialization. The default SLAB HRTF database is "jdm.slh" and the default HRTF directory is "<install dir>\HRTF". Use the SLABScape application to change these defaults (see menu item: SLAB | SLAB Registry Settings...).

SLAB HRTF Database Format - Version 1

Version 1 databases are binary files with all values stored in 16-bit integer. They typically have the suffix .dat. The following parameters are assumed:

Azimuth, number of = 13, (180 150 120 90 60 30 0 -30 -60 -90 -120 -150 -180)
Azimuth increment = 30
Azimuth zero index = 6
Elevation, number of = 11, (90 72 54 36 18 0 -18 -36 -54 -72 -90)
Elevation increment = 18
Elevation zero index = 5
Number of ears = 2
Number of ITD points = 143
ITD positive = left ear lag
ITD negative = right ear lag
Number of HRIR points = 256
Number of HRIR sample bytes = 2 (16-bit integer)
Maximum HRIR sample value = 32767
Minimum HRIR sample value = -32768
Float HRIR sample scaling factor = 32768.0f

The table below illustrates the HRIR and ITD data storage format (ordered by individual data values):

```
 AZ    EL
 180,  90, left  ear, hrir pt 0        Left ear points
...
                      hrir pt 255
 180,  90, right ear, hrir pt 0        Right ear points
...
                      hrir pt 255
 180,  72, left  ear, hrir pt 0        Elevations (grouped by azimuth)
...
      -90, right ear, hrir pt 255
 150,  90, left  ear, hrir pt 0        Azimuths
...
-180, -90, right ear, hrir pt 255

 180,  90, delay                       Delays
 180,  72, delay
...
-180, -90, delay
```

SLAB HRTF Database Format - Version 2

Version 1 to Version 2 changes:
1) The database now contains a header.
2) The 16-bit integer data type has been changed to single-precision floating-point.
3) The database files typically have a .slh suffix.

HRTF Database Header

```
typedef struct
{
short  nVersion;         // database format version
char   strName[32];      // subject's name
char   strDate[8];       // date measured
char   strComment[256];  // text comment
short  nAzInc;           // degree increment of az locations
short  nElInc;           // degree increment of el locations
short  nNumPts;          // number of HRIR points per entry
long   lSampleRate;      // sample rate when capturing HRIR
} HRTFHeader;
```

The HRIR and ITD data immediately follow the header as specified above for Version 1. The azimuth and elevation increments and the number of HRIR points are now determined from the header.

See Also: CSRAPI::LstHRTF() in the SRAPI Reference Manual.

Top

Error Handling

There are two error catching modes in SLAB: Non-Render-Time and Render-Time. When SLAB is rendering (i.e., after a call to RenderStart()), SLAB is in the Render-Time error catching mode. When SLAB is not rendering (i.e., after the SLAB object is allocated, after a call to RenderStop(), or after an error), SLAB is in the Non-Render-Time error catching mode.

The two error catching modes are required because of Render-Time multi-tasking. When rendering, SLAB runs in two threads of execution, the API Thread (same as the user's thread) and the Render Thread, a thread created during a call to RenderStart(). Errors can occur inside the Render Thread without an API call (e.g., a state error in a Render Plug-In not directly connected to an API call). The user cannot check the return value of an API function to catch this error (unless "polling" and then only during the next "error poll", see below).

Non-Render-Time Error Catching

Function Return Values
Many SRAPI functions return SLABError. Non-Render-Time errors are usually caught with function return values.

Render-Time Error Catching

There are two additional methods for catching Render-Time errors:

1. Error Messaging
   When using Error Messaging, SLAB sends error messages to the user's application via Windows messaging. Before calling RenderStart(), call SetNotify() specifying a notification window and message ID value. If a Render-Time error occurs, the notification message will be sent to the notification window. To catch the message, map the notification message ID to a message handler using a CWnd message map. Use an error query function to verify the notification corresponds to an error (notifications can also be used to indicate the completion of one-shot playback).
2. Error Polling
   Whenever an error occurs, SLAB enters an "error state." In an error state, all error-returning SRAPI functions return the current error. Error Polling refers to catching *existing* errors via function return values or error status functions. In other words, the user is checking for an error not necessarily caused by the function itself (e.g., an error caused by a previous function where the return value wasn't checked).

Error State

SLAB enters an "error state" whenever an error occurs.

In an error state:

- rendering is stopped
- the sound output stream is stopped
- script processing is stopped
- all functions returning SLABError return the current error
- error information can be queried via ErrorState(), Error(), ErrorString(), and ErrorStack()
- error state maintained until explicitly cleared with ErrorClear()

Error Functions Overview

The following functions exist for querying error information and clearing error state:

- ErrorState(): returns true if in error state, false if not
- Error(): returns the current SLABError
- ErrorString(): returns a string describing the current error
- ErrorStack(): returns a string providing call stack error information for the current error
- ErrorClear(): clears error state
- SetNotify(): sets the notification window and the notification message ID value
- LogName(): sets the name of the log file used for logging error information
- Reset(): resets SLAB

See Also: CSRAPI Error Functions in the SRAPI Reference Manual.

Top

Tips

Tips for writing psychoacoustic experimentation software, etc.:

- To disable source-listener range-dependent gain scaling use SrcRadius(idSrc, dRadius, 0.0).
- To place sound sources by specifying azimuth and elevation use LstPosition(0.0, 0.0, 0.0, 0.0, 0.0, 0.0) and SrcLocatePolar(idSrc, az, el, range). Note: SLAB does not have a listener-relative source location function.
- To specify sound source time delays use LstPosition(0.0, 0.0, 0.0, 0.0, 0.0, 0.0) and SrcLocatePolar(idSrc, az, el, dSoundSpeed * time_delay).

Top

SLAB User Manual

Last updated: May 20, 2004

Applications and Utilities

- SLABReg
  SLABReg is used to set the SLAB Registry Settings. These settings are required.
  Source Code: <install dir>\src\SLABReg\

- SLABScape
  SLABScape provides a graphical user interface for experimenting with the SLAB Render API Scene parameters. It can also used to modify the SLAB Registry Settings (e.g., Default HRTF). For more information, see the SLABScape Help Page.
  Source Code: <install dir>\src\SLABScape\

- SLABSound
  SLABSound provides ASIO device information.
  Source Code: <install dir>\src\SLABSound\

- SLABWireDemo
  SLABWireDemo demonstrates the use of SLABWire. It also contains OpenGL displays that react to the frequency content of a sound source.
  Source Code: <install dir>\src\SLABWireDemo\

- SLABLatency
  SLABLatency is used to perform latency measurements.
  Source Code: <install dir>\src\SLABLatency\

- SLABX
  SLABX demonstrates the use of the SLAB Render API.
  Source Code: <install dir>\src\SLABX

SLABX

SLABX demonstrates how to write a typical SRAPI-based application using Microsoft Visual C++. Note: The SLAB User Release has only been tested with the Microsoft Visual C++ compiler.

SLABX contains two demos, *Trajectory Demo* and *HRTF Database Per Source Demo*. In the *Trajectory Demo*, a white noise sound source is placed in a circular trajectory about the listener's head. In the *HRTF Database Per Source Demo*, one wave file sound source is placed at (0.5m,0.5m,0.0m) and another is placed at (0.5m,-0.5m,0.0m). The user can select between three different HRTF databases for each source. The slabtools mhrir.m and mitd.m were used to create the modified databases.

The SLABX documentation is in the form of source code comments in SLABXDlg.h and SLABXDlg.cpp. These files are the only SLABX files that contain SRAPI code. Begin by reading SLABXDlg.h...

Source Code: <install dir>\src\SLABX

Top

SLAB User Manual

Last updated: May 20, 2004

SLABScape

SLABScape provides a graphical user interface to the SLAB Render API. SLABScape also allows the user to specify sound source trajectories, enable Fastrak head tracking, edit and play SLAB Scripts, A/B different rendering strategies, and visualize the environment via a Direct3D display.

Contents:

- Menus
- Toolbar
- Shortcuts
- Source Dialog
- Environment Dialog
- Listener Dialog
- Fastrak Settings Dialog
- SLAB Settings Dialog
- SLABScript Editor
- 3D View
- Status Bar

Menus

Menu definitions:

- **File**
  - **New** - initializes scene parameters to their default settings.
  - **Open...** - prompts to open a **.scn** "SLAB Scene" file.

    Note: SLABScape does not strictly adhere to the MFC (Microsoft Foundation Classes) view/document model. If you attempt to reopen an open scene, MFC prevents the open and does nothing. As a work-around, simply click the File New button before reopening.
  - **Save** - saves the current scene. The sound source and listener controlled by controller (tracker or Firefly) setting is not saved, nor is the Render Plug-In ID if it is above RENDER_PLUGIN. This is because it is not known if the controller or renderer will be available next time SLABScape is executed.
  - **Save As...** - prompts to save the current scene to a file.
  - **Most Recently Used Files** - lists recently used scene files.
  - **Exit** - exits the application.
- **View**
  - **Scene**

        - **Source Dialog...** - displays the Source Dialog.
        - **Environment Dialog...** - displays the Environment Dialog.
        - **Listener Dialog...** - displays the Listener Dialog.
    - **Controllers**
        - **Fastrak Settings...** - displays options for configuring a Polhemus Fastrak tracker.
        - **Firefly Settings...** - displays options for initializing and configuring Firefly. These controls are currently implemented in-house only.
    - **Simulation Settings...** - displays simulation parameter options for tick update rate and hand control.
    - **Parameter Overlay** - overlays the 2D view with source and listener scene information. All distances are in meters, all angles in radians. Azimuth, elevation, and range are relative to the origin.
      Sources: x, y, z, azimuth, elevation, range, gain
      Listener: x, y, z, yaw, pitch, roll
    - **Update Rate** - displays the update rate of the SLAB Renderer and the visual display in the status bar.
    - **Monitor SLAB Autos** - monitors the positions of the sources and listener as defined by the SLAB Renderer (the SLAB Renderer can automatically update its own parameters when processing a script).
    - **Toolbar** - controls the display of the toolbar.
    - **Status Bar** - controls the display of the status bar.

- **SLAB**
    - **Render** - starts or stops the SLAB Renderer. Renders using the currently selected renderer (see Plug-Ins menu).
    - **SLAB Settings...** - provides access to all pre-render-time SLAB parameters (except materials, see Environment Dialog). The signal processing parameters, script settings, input/output settings, and the ASIO channel map are set using this dialog.
    - **SLAB Registry Settings...** - lists the SLAB installation registry settings. The user can modify the default HRTF and the default HRTF directory registry settings using this dialog.
    - **SLABScript Editor...** - displays the SLABScript Editor.

- **Plug-Ins**
    - **About Plug-Ins...** - displays information about all plug-ins found in the SLAB Render Plug-In directory (<install dir>\bin\rplugin). The SLAB Renderer must be started before plug-in information is available.
    - **Installed Plug-Ins** - all SLAB Render Plug-Ins will be added to the Plug-Ins menu after the SLAB Renderer is started.

      Warning: Only the Left-Monotic, Right-Monotic, Diotic, and DioticV plug-ins are affected by the Monotic/Diotic Gain setting on the source dialog. All other

plug-ins use the source gain setting. Thus, plug-ins that do not attenuate the sound signal in their rendering model can result in a significant volume increase when selected! (e.g., "Spatial" attenuates the signal due to spherical spreading and the head-related impulse response in addition to the gain setting; "Plug-In Example" does not attenuate beyond the gain setting.)

The following plug-ins are included in the SLAB User Release:

- **Spatial** - selects the spatial display SLAB Render Plug-In.
- **Left-Monotic** - selects the left-monotic display SLAB Render Plug-In.
- **Right-Monotic** - selects the right-monotic display SLAB Render Plug-In.
- **Diotic** - selects the diotic display SLAB Render Plug-In.
- **DioticV** - selects the variable diotic (source y = 0.0) / left/right monotic (source y left or right) display SLAB Render Plug-In.
- **Plug-In Example** - selects the example SLAB Render Plug-In (diotic display).

- **Help**
  - **Help...** - displays this help page.
  - **About...** - displays about dialog box.

Top

Toolbar

Toolbar button definitions (from left to right):

- **New** - see Menus.
- **Open...** - see Menus.
- **Save** - see Menus.
- **Source Dialog...** - see Menus.
- **Environment Dialog...** - see Menus.
- **Listener Dialog...** - see Menus.
- **SLAB Render** - see Menus.
- **SLAB Settings...** - see Menus.
- **HRTF 1** - selects HRTF 1 defined in Listener Dialog.
- **HRTF 2** - selects HRTF 2 defined in Listener Dialog.
- **HRTF 3** - selects HRTF 3 defined in Listener Dialog.
- **Spatial** - see Menus.
- **Left- Monotic** - see Menus.
- **Right-Monotic** - see Menus.
- **Diotic** - see Menus.

- **Parameter Overlay** - see Menus.
- **Zoom Out** - zooms-out on the 2D view. The zoom level changes the source and listener placement slider ranges.
- **Zoom In** - zooms-in on the 2D view. The zoom level changes the source and listener placement slider ranges.
- **3D View** - toggles between the 2D view and the 3D View. The 3D View is rendered using Direct3D, so the quality of the display is determined by the Direct3D features of your graphics card.

Top

Shortcuts

Command	**Shortcut Key**
Sound Sources	
Control source number x	1-9
Cycle control	c
Reset source location forward	End
Up	u
Down	d
In	i
Out	o
Mute/UnMute	Ins
Increase gain	g
Decrease gain	a
Sound Source Trajectory	
Slower	s
Faster	f
Start/Stop	Space
Change direction	Enter
Listener	
Move to origin	Home
Forward	Up Arrow
Backward	Down Arrow
Left	Left Arrow
Right	Right Arrow

Up	Page Up
Down	Page Down
Spin	/

Top

Source Dialog

The Source Dialog allows the user to manipulate source parameters while rendering. The dialog's layout is similar to an audio mixer with a channel strip for each sound source. These strips contain the familiar gain fader and mute button, but also include a source motion droplist, a relative source location dialog button (REL), a current source controlled button (channel #), a pause/play button (PAU), and a rewind button (REW).

The source motion droplist contains five options: None, Circle, Rectangle, Track, and Firefly. The Circle and Rectangle trajectories are generated by SLABScape, *not* the SLAB Renderer (the SLAB Renderer has a built in trajectory mechanism that is currently only supported by the SLABScript Editor). The Circle and Rectangle trajectories occur in a horizontal plane. The Circle radius is defined by the initial radius. The Rectangle dimensions are determined by the initial location of the source. Keyboard shortcuts provide further trajectory control. The Track option enables source tracking. This causes the sound source to follow the location of a Polhemus Fastrak tracker receiver. The tracker is opened using the "Open Tracker" button on the "Tracker Settings" dialog (View | Controllers menu). Firefly allows a source to be located using USB camera capture. This option is currently implemented in-house only; it does nothing in the User Release.

The REL button pops-up the Relative Source Location dialog. This allows the placement of a source relative to the listener's head using azimuth, elevation, and range coordinates. This operation places the listener at the origin. Relative source placement allows one to listen to a specific location in an HRTF database or to investigate the effects of HRTF interpolation.

The channel# button determines the "current source." The current source is controlled via the Location sliders on the right side of the dialog and by using keyboard shortcuts.

The Monotic-Diotic Gain slider sets the source gain for the Left-Monotic, Right-Monotic, Diotic, and DioticV renderers.
Warning: If both release and debug versions of the plug-in exist, the debug version is not affected by this slider. Thus, a signification increase in volume may result if the debug version is chosen!

The "Allocate Sources..." button pops-up the "Source Allocation" dialog for allocating sound sources.

The "Source Radius" droplist sets the radius of the current sound source. The "Source Spread" droplist sets the spherical spreading coefficient for the current sound source (0 = none, 1 = normal, 2 = exaggerated).

Top

Environment Dialog

The Environment Dialog controls features of the listening environment. These features include the dimensions of a rectangular room, the materials of the room surfaces, and the number of sound images modeled. The number of sound images is controlled by the Reflections row of checkboxes. D refers to the direct path, XP to the positive x-axis plane, and so on. The surface materials are selected via the Surface Materials dialog displayed by pressing the Materials... button.

Top

Listener Dialog

The Listener Dialog controls the listener location and orientation, tracking options, and HRTF database selection. The Tracker Enable button enables head tracking. This causes the listener to follow the location of a Polhemus Fastrak tracker receiver. The tracker is opened using the "Open Tracker" button on the "Tracker Settings" dialog (View | Controllers menu). Selecting the Orientation Only button locks the listener at the origin while listener yaw, pitch, roll continue to be updated.

The HRTFs... button displays the "HRTF Databases" dialog. This dialog sets the HRTF databases loaded by the HRTF 1, HRTF 2, and HRTF 3 toolbar buttons.

Top

Fastrak Settings Dialog

These settings initialize a Polhemus Fastrak electromagnetic tracking peripheral.

- **Com Port** - Select the com port connected to the Fastrak. The Fastrak communication settings (DIP switches) must be set to 115200 bps, 8 data bits, no parity, no flow control.
- **Hemisphere** - Direction of valid Fastrak transmitter hemisphere (see Fastrak manual).
- **Receiver 2** - Selecting "Hand" will enables and map the second Fastrak receiver to the Hand. (The Hand is currently in development.) Enabling a second receiver drops the Fastrak update rate from 120Hz to 60Hz.
- **System Com** - Typically, SLABScape sets the com settings using the Microsoft Comm API. Some com devices do not support this feature (e.g., some USB-to-serial adapters) and use the Device Manager com settings instead. These devices will cause "Open Tracker" to fail. Selecting this option bypasses the Comm API settings.

  Note: for com devices that support Comm API com settings, this option will cause the com port to be initialized with the most recent Comm API settings. Thus, only use this option if you have to.
- **Open Fastrak** - Initiates communication with the Fastrak. Opening the Fastrak sets the current settings and cannot be canceled.
- **Save Settings** - Saves the current dialog settings in the registry.

- **Restore Defaults** - Restores dialog settings default settings.
- **OK** - Sets dialog settings.
- **Cancel** - Resets dialog settings to previous settings.

Top

SLAB Settings Dialog

These settings initialize SLAB prior to rendering and are described in more detail as non-render-time functions in the SRAPI Reference Manual.

- **Signal Processing Parameters** - see the SRAPI Reference Manual.
- **Script Settings** - if the "Process script file while rendering" checkbox is checked, the specified file will be parsed the next time the SLAB Renderer is started.
- **Input/Output Settings** - for an explanation of the DirectSound Settings see the SRAPI Reference Manual. If the Memory/File Sound Output Type is selected, SLABScape routes sound output to the specified file. In this mode, no sound is heard through the sound peripheral. Since rendering does not occur in real-time, the SLABScape user interface is not synchronized to sound output. For simple scenes, SLAB will render faster than real-time; for complex scenes, SLAB will render slower than real-time. This option is primarily for debugging, but can be used with the SLABScript Editor to render a "canned" dynamic scene. Output is stopped when the SLAB Renderer is stopped.

  Caution: Sound file output is generated at 172 kb/s, so do not forget to stop the SLAB Renderer!

Top

SLABScript Editor

The SLABScript Editor allows for the creation and auditioning of SLAB scripts.

Button definitions:

- Add Event - displays a dialog for adding SLABScript commands. The default parameters are all set to zero. Be sure to change this value for Source Index (valid values = 1-4).
- Open... - opens a previously saved script.
- Save As... - saves the current script.
- PrepScape - prepares SLABScape for script playback:
  - enables script processing (enables checkbox: SLAB Settings | Script Settings | Process script file while rendering)
  - sets the SLAB Settings script file to the last script loaded or saved (sets edit control: SLAB Settings | Script Settings | File)
  - enables SLAB auto monitoring (enables menu item: View | Monitor SLAB Autos)

To audition a script, press the PrepScape button and then start the Spatial renderer (select Spatial

Render Plug-In, press SLAB Render toolbar button).

Top

3D View

The 3D View displays a Direct3D view of the SLABScape environment. Right-clicking on the 3D View brings up a context-sensitive menu for adjusting various 3D View options. If your graphics adapter or desktop display settings do not support Direct3D acceleration, this button will be disabled.

Top

Status Bar

The two status indicator panes in the lower-right of the SLABScape window show the following:

- Update Rate Indicator - The left update number indicates the rate at which the SLAB Renderer is updated. The right update number indicates the update rate of the visual display. The percent number indicates the percent of CPU use.
- Clip and Underflow Indicator - The "C=" field indicates the number of renderer mixer clips (implementation renderer-dependent). The "U=" field indicates the number of DirectSound underflows.

Top

SLAB User Manual

Last Updated: May 20, 2004

SLAB Render Plug-Ins

- Overview
- Starter Project
- Spatial

Overview

All rendering in SLAB is performed with SLAB Render Plug-Ins. SLAB Render Plug-Ins are subclassed from CRPlugIn, built as DLLs, and placed in the SLAB bin\rplugin directory. All plug-ins found in this directory will be read into memory when a CSRAPI object is allocated. The SRAPI Render Functions exist for querying and selecting plug-ins. Through the CScene class, Render Plug-Ins are given access to SLAB's scene parameters, input delay lines, and output stream.

See Also: CSRAPI Render Functions, CRPlugIn, and CScene in the SRAPI Reference Manual.

Top

Starter Project

The *rmyplugin* and *rdiotic* projects in the SLAB src\ directory can be used as Render Plug-In examples. The *rmyplugin* project is also used as a starter project for creating user plug-ins. The *rdiotic* project contains the source code for the SLAB monotic and diotic renderers and shows how to place multiple plug-ins in one DLL.

Follow the steps below to create your own plug-in using Microsoft Visual C++:

1. Create a new Win32 DLL project.
2. Delete the generated .cpp from the project and from the directory.
3. Copy rplugin.cpp from SLAB's src\rplugin\ directory to your new project directory. Rename it if you wish. Add it to the project.
4. Add the SLAB include\ directory to the **Additional Include Directories** compiler property.
5. For SLAB to find your DLL it must reside in the SLAB bin\rplugin\ directory. Set the linker **Output File** property as follows:
    1. for debug DLLs, enter the name of your DLL appended to the SLAB bin\rplugin\ path:
       "<SLAB install dir>/bin/rplugin/r*d.dll"
       where '*' denotes a name of your choosing, 'r' indicates the DLL is a "render" plug-in, 'd' indicates the DLL is the debug version of the DLL
    2. for release DLLs, enter "<SLAB install dir>/bin/rplugin/r*.dll"

Source Code: The source code for rmyplugin is in <install dir>\src\rmyplugin\. The source code for rdiotic is in <install dir>\src\rdiotic\.

Top

Spatial

The Spatial Render Plug-In contains SLAB's spatialization algorithm. SAR is a Search And Rescue communications plug-in that demonstrates how to extend Spatial for special-purpose applications.

Source Code: The source code for Spatial is in <install dir>\src\rspatial\. The source code for SAR is in <install dir>\src\rsar\.

Top

SLAB User Manual

Last Updated: May 20, 2004

slabtools

The *slabtools* are MATLAB utilities for visualizing and manipulating HRTF data and for visualizing SLAB processing. To use the slabtools, the slabtools directory should be added to your MATLAB path. General slabtools help can be obtained by typing "help slabtools" at the MATLAB command prompt. Individual tool help is available by typing "help <toolname>".

Source Code: <install dir>\slabtools

Definition of the variables *map*, *hrir*, and *itd*

These variables are similar to the Snapshot variables of the same name. Snapshot is an HRTF measurement system developed by Crystal River Engineering.

- *hrir* contains HRIR taps stored in rows. All left ear HRIRs are followed by all right ear HRIRs (e.g. the right ear response `hrir(:,i+size(map,2))`corresponds to the left ear response `hrir(:,i)`).
- *itd* contains the interaural time delay (in samples).
- *map* maps azimuth and elevation database locations to *hrir* and *itd* indices (see *hindex.m*). Each azimuth and elevation pair is a two-element column vector `[ el; az ]`.

The number of columns in *map*, *hrir*, and *itd* is HRTF database-dependent. Since *hrir* contains the left and right ear responses, it will have twice the number of colums as *map* and *itd*.

The use of *map*, *hrir*, and *itd* is identical to Snapshot with the following exception:

- SLAB's *map* variable is grouped by azimuth (e.g. `[ [ 90; 180], [72; 180] ]`), Snapshot's, by elevation (e.g. `[ [ 90; 180], [90; 150] ]`)

sarc

sarc is the SLAB HRTF archive format. The goal of the sarc format is to provide a flexible MATLAB HRTF data structure and storage format for HRTF manipulation, visualization, and analysis. To learn more about sarc, type "help sarc" at the MATLAB command prompt.

- **sarc is presently in development!** We are currently upgrading our HRTF measurement system to the AuSIM HeadZap system. Thus, the sarc format will change to accommodate the data generated by HeadZap. I would like to support HRTF data from other formats as well (e.g., CIPIC). Any suggestions regarding the sarc format or similar efforts welcome! Please send suggestions to Joel D. Miller at jdmiller@mail.arc.nasa.gov.

Examples

SLAB\HRTF\jdm_ie.slh was created as follows:
```
>> mitd( 'jdm.slh', 'jdm_ie.slh', 0, 0 );
```

SLAB\HRTF\jdm_he.slh was created as follows:
```
>> mhrir( 'jdm.slh', 'jdm_he.slh', 0, 0 )
```

map2map()

map2map() is the intellectual property of Jonathan Abel and is covered under the same usage restrictions outlined in the SLAB Software Usage Agreement. Many thanks to Jonathan Abel for allowing map2map() to be released with the SLAB User Release!

SLAB User Manual

Last Updated: May 20, 2004

Appendix

- Glossary
- Known Issues

Glossary

Diotic	Diotic is a Render Plug-In that supports diotic auditory display.
DioticV	DioticV is a Render Plug-In that supports diotic, left-monotic, and right-monotic auditory display.
DSP Function	A DSP Function is a C++ class which is "plugged-into" SLAB's signal flow architecture. This occurs in SLAB's SLABWire layer and is hidden from the SLAB Render API user. See "Render Plug-In" for a higher-level version of this functionality.
HRTF Database	Set of HRIRs and ITDs measured about a sphere specified by azimuth and elevation.
HRTF Map	See HRTF Database.
LeftMonotic	LeftMonotic is a Render Plug-In that supports left-monotic auditory display.
MSVC++	Microsoft Visual C++. The SLAB development environment.
RightMonotic	RightMonotic is a Render Plug-In that supports right-monotic auditory display.
Render Plug-In	Render Plug-Ins are DLLs that can be loaded and inserted into the SLAB rendering architecture. The SLAB Render API supports five Render Plug-Ins: Spatial, DioticV, Diotic, LeftMonotic, and RightMonotic.
Scene	The Scene is a component of SRAPI. The user uses SRAPI Scene functions to set acoustic scene and renderer parameters.
Scene Library	The user must link with the *scene* library to use the SRAPI Scene API.
SLAB	SLAB is short for Sound Lab. What "SLAB" refers to is context-dependent. Generally speaking, SLAB refers to the entire collection of SLAB-related material, applications, libraries, documentation, etc. Frequently, however, "SLAB" will refer to one of the constituent parts (e.g., the SLAB Spatial Render Plug-In, SRAPI, SRAPI library, SLABWire, etc.).
SLAB Libraries	There are three SLAB libraries: SLAB Wire, SRAPI, and Scene.

SLAB Render API	The SLAB Render API (application programming interface) (SRAPI) is the interface through which the user interacts with a SLAB Render Plug-In. This API is used to allocate sound sources and control rendering. The Scene component of SRAPI is used to specify acoustic scene parameters.
SLABWire	SLABWire refers to a collection of objects for creating a sample stream input to DSP to sample stream output chain (or "digital wire"). This layer is hidden beneath SRAPI. SLABWire can also be used as a stand-alone API for writing general purpose streaming audio processing applications.
SLABWire Library	The SLABWire Library contains the SLABWire layer.
SLABScape Application	An application that demonstrates many of the features of SLAB.
Spatial Render Plug-In	Spatial is a Render Plug-In for rendering HRTF-based virtual acoustic environments.
SRAPI	SLAB Render API.
SRAPI Library	The user must link with the *srapi* library to use SRAPI, the SLAB Render API.
VAE	Virtual Acoustic Environment.

Top

Known Issues

- SLAB does not detect all occurrences of DirectSound buffer underflow. Usually, underflow results in an underflow counter increment. But, sometimes, stuttering is heard without an underflow increment. This is most likely to occur when the CPU is at maximum usage or a CPU usage spike occurs (e.g. reading a file). A missed underflow can be caused by a late Windows timer update, allowing the DirectSound pointers to wrap around the buffer to a valid location.

- The SLABScape "hand" is in development and not documented.

- slabref.pdf Acrobat Issues: (1) light blue background of SRAPI function name not in pdf, (2) class hierarchy diagrams not in pdf (Acrobat doesn't support PNG), (3) sections in strange order (order automatically generated by Acrobat).

Top

SLAB User Manual

Last Updated: May 20, 2004