

# **RAGE**

# **Tutorials**

## **2D/3D Sound**

Software & Documentation © The RAGE Team 2005

Version Relevance: JGE Release 0.1+

# CONTENTS

Are you reading the right tutorial?	3
Required Imports	3
Required Properties	3
Opening the SoundDevice	4
Closing the SoundDevice	4
2D Sound	5
2D Example Code	6
3D Sound	7

# ARE YOU READING THE RIGHT TUTORIAL?

This tutorial is intended to show you how to use the sound system to play 2D sound and then add positions to sounds to make it automatically 3D.

This tutorial does NOT cover sound formats or other concepts of audio.

## REQUIRED IMPORTS

Sound Package: `jge.hal.sound`

## REQUIRED PROPERTIES

<i>Property</i>	<i>Description</i>	<i>Example Value</i>
<code>class.SoundPlatform</code>	Implementation of <code>jge.hal.sound.SoundPlatform</code> to use in JGE.	<code>plugin.hal.sound.MySoundPlatform</code>

# OPENING THE SOUNDDEVICE

The SoundDevice is an interface through to the audio playing device on the system. Before using the SoundDevice at all, we need to open the device so that it is initialised.

Calling the following function will initialise it:

```
jge.hal.sound.SoundDevice  
+open():void
```

# CLOSING THE SOUNDDEVICE

When finished with the sound device it should be closed so that any resources it is consuming/owning can be released. Upon doing this all buffers will become useless and should be discarded. To use them again they will need to be reloaded, and sources will need to be recreated.

Calling the following function will de-initialise it:

```
jge.hal.sound.SoundDevice  
+close():void
```

# 2D SOUND

All audio is stored in a generic format encapsulated in the `jge.hal.sound.Sound` object. Resource loaders that load audio should return this object.

A buffer is the platform's representation of a Sound object, it is a handle to a copy of the audio on the platform. Before being able to play audio, a buffer must be created with the audio data in the Sound object.

Because a buffer is just a set of audio data, it needs to be attached to a source. A source is a location of audio, which can be played, looped, etc. Although a source can have a position, velocity, etc, we do not need to touch these functions for 2D sound. Once attached to a source, the source can be played which results in the data in the buffer being played. You can also set the gain of a source if you want its volume to be different.

Before playing audio the device will need to be initialised and its listener orientation will need to be set. Do not be concerned with this function call as of yet, its meaning will only be of use if you are doing 3D sound.

**Step 1: Initialise the SoundDevice and set its listeners orientation** to match 2D Sound.

**Step 2: Get the audio from file** (a Sound object). You will need a resource loader so load audio from a sound file (see the plugins website for one you can use).

**Step 3: Load the audio into a buffer.** This will put the audio in a format for the platform ready for playing.

**Step 4: Create a source** to play the audio from.

**Step 5: Attach the buffer to a source.**

**Step 6: Call play** on the source.

## EXAMPLE CODE

```
// cache references
ResourceManager rm = ResourceManager.getInstance();
SoundDevice sd = SoundDevice.getInstance();

// STEP 1: init the SD and set the listener orientation
sd.open();
sd.setListenerOrientation(0, 0, -1, 0, 1, 0);

// STEP 2: get the audio from the resource system
Sound sound = (Sound)rm.getResource("audio.wav");

// STEP 3: create a buffer from the audio
Buffer buffer = sd.createBuffer(sound);

// STEP 4: create a source
Source source = sd.createSource();

// STEP 5: attach the buffer to the source
source.setBuffer(buffer);

// STEP 6: play the source
source.play();
```

# 3D SOUND

3D sound is setup the same way 2D sound is, infact 2D sound is 3D sound but without effects on volume based on position, etc. This chapter will follow on from the previous and show how to set positions and other 3D properties of a source.

In 3D sound a listener exists, which is the location of where we are hearing audio from. There is always a single listener for the sound device and its properties are mutated by calling functions on the sound device. We had to set its orientation in the 2D sound chapter, in 3D sound we will also set its position and velocity. The listener also has a gain, which is like an overall volume control for all audio.

All Source objects have their own position, velocity, direction and pitch. The position is used relative to the listeners position, and effects are placed on the source emulate the sound in that position in a real world situation. The velocity and direction also have affects on the sound.

```
SoundDevice sd = SoundDevice.getInstance();

// For the purpose of this example, we will assume that
// the sources have been assigned buffers in the same
// way as demonstrated in chapter: 2D Sound.
Source src1 = sd.createSource();
Source src2 = sd.createSource();

// set the position of the sources
src1.setPosition( 0.0f, 0.0f, 0.0f);
src2.setPosition(10.0f, 0.0f, 10.0f);

// set the position of the listener to be the same
// as src1
sd.setListenerPosition(0.0f, 0.0f, 0.0f);

// set the orientation to be forward
sd.setListenerOrientation(0, 0, -1, 0, 1, 0);

// play the sounds
src1.play();
src2.play();
```

When playing the two sounds, you would hear sound one (src1) right close as if it was directly in front of your face and sound two (src2) you would hear at a distance on your right.

