AXEL Application Programming

Conrad Parker

AXEL Application Programming

by Conrad Parker

Table of Contents

1. Introduction .... **??**
    1.1. Streams, channels and chunks .... **??**
    1.2. Decks, tracks, layers and sounds .... **??**
    1.3. Envelopes and mix automation .... **??**
    1.4. Processing latency and caching .... **??**
Index .... **??**

List of Figures

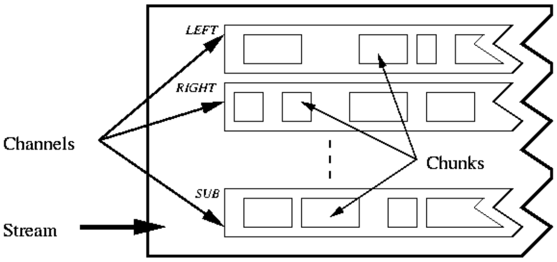
1-1. Inside an Axel stream .... **??**
1-2. Inside an Axel deck .... **??**

Chapter 1. Introduction

Axel is an audio sequencing and mixing library that provides a multichannel, sparse audio data container (streams), a structured mixing abstraction (decks), and widely useful means of generating control data (via envelopes) and of caching audio data.

1.1. Streams, channels and chunks

Figure 1-1. Inside an Axel stream



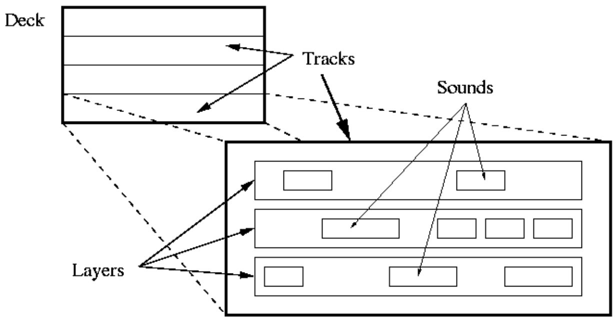
The abstraction of multichannel audio data in Axel is known as a stream. The structure of a stream is shown in Figure 1-1. A stream may consist of multiple channels, each of which can consist of an arbitrary number of sparsely placed chunks of raw audio data. The channels are named with spatial names such as LEFT, RIGHT and CENTRE as required for common home, studio and theatre environments.

Generic routines are provided for mixing, multiplying and blending streams of data.

1.2. Decks, tracks, layers and sounds

The top level structured mixing abstraction in Axel is known as a deck. A deck contains a number of tracks which are mixed in parallel. Each track may contain a number of layers which are mixed from bottom to top in series. Finally, these layers each contain a sequence of sounds with transparency. This structure is illustrated in Figure 1-2.

Figure 1-2. Inside an Axel deck



The sequence of sounds in a layer can be indexed by samples, seconds or tempo. Sounds provide audio data from any instrument or effect source, and these sources can each be reused multiple times. A sound can even source its audio data from another entire deck, thus decks can be used to sequence other decks. In this manner effects can be applied to sequences of decks, and sequences of decks can be stored as higher level units such as verses and choruses in a music application.

1.3. Envelopes and mix automation

The information describing how a parameter changes over time appears as a generic data source. In order to create this mix automation information Axel provides linear and spline envelopes. However, parameters could alternatively be controlled by other means such as from a recording of physical slider values, from a sine wave generator, or from a deck constructed solely to generate interesting parameter values.

1.4. Processing latency and caching

All sound sources in Axel, including streams, decks and envelopes, implement a base set of functionality such as for seeking and for processing small regions of data. The requirement for low processing latency is met by these semantics, which are optimised for sequential processing but allow the evaluation of arbitrarily small temporal slices.

Axel also provides a generic caching abstraction which can be applied to any sound source.

Index