GRAME - Computer Music Research Lab.
Technical Note - 01-11-07

Callback adaptation techniques

St phane Letz
November 2001
Grame - Computer Music Research Laboratory
9, rue du Garet BP 1185 69202 FR - LYON Cedex 01
letz@grame.fr

Abstract

This document describes a callback adaptation technique developed for the PortAudio port on ASIO. This method handle buffers of different sizes and guarantee lowest latency added by buffer size adaptation.

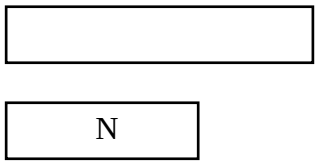
1 Introduction

The callback adaptation problem was encountered whilst doing a port of the PortAudio API on ASIO API. PortAudio is a cross-platform library that provides streaming audio input and output, ASIO is a Macintosh and Windows API that provides streaming audio input and output. Both API are callback based, but may use audio buffers of different size. This document presents an algorithm developed to allow adaptation between the 2 callback systems.

2 Adapting callbacks

The system we are considering is driven by an audio native callback (the Òhost Ó callback) usually called by the hardware interrupt. The hostr 67 Tm 0.01122hwas thefoalloring prtotype :r

3.3 Case 3 : host buffer size superior of user buffer size



Beginning of callback 2 : 100 new frames are received

Host input buffer = **100**
User input buffer = 30 : frames kept from the previous cycle
User ouput buffer = 0
Host ouput buffer = 0

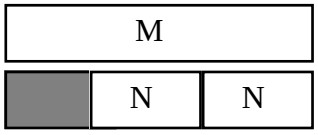
End of callback 2

Host input buffer = 0
User input buffer = 60 : remainder of (100 + 30)/70
User ouput buffer = **70**
Host ouput buffer = 0

There is a problem here because a complete host ouput buffer can not be produced (because 70 <100)**.** Thus the value has to be choosen to guarantee that a **complete host ouput buffer can be produced at each cycle**. We could just add **N null frames** to start the process. But adding null frames at the beginning of the first host output buffer (that is shifting the whole ouput stream), increases the global latency.

Thus we would like to find the **smallest number of null frames** that must be put in the first output host buffer to guarantee that the audio streaming process can be done. This value can be found by doing the following reasoning :

- **At the first host callback :** M frames are received, and M/N PortAudio callback can be called. The minimum number of needed frames to complete a full M ouput buffer is **M%N** (where % gives the **remainder** of M/N).



3.4 Case 4 : host buffer size inferior of user buffer size

```
        adaptor->userCallback = callback;
        adaptor->hostBufferSize = M;
        adaptor->userBufferSize = N;
        return adaptor;
}
```

// Computes the shift value used for the first ouput buffer : this value is used to initialize the write offset in the user input buffer or the host ouput buffer depending of M and N

```
void  InitCallbackAdaptor }
```

```
long framesInputUserBuffer;
// number of frames needed to complete the host output buffer
```

// **To Òsimulate Ó the hardware interrupt : the host ouput buffer has been used, it is now empty**

```
void WriteOutputHostBuffer(CallbackAdaptor* adaptor)
{
	a
}
```

// **Prints the state of the CallbackAdaptor structure**

```
void
```

5 Conclusion

A callback adaptation technique has been presented. It allows to use two callbacks that use buffeob lr diffeoent sizes,