社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－DPS－117 (40)
2004－CSEC－ 24 (40)
2004／3／5

仮想の“音の部屋”によるコミュニケーション・メディア voiscape の JMF と Java 3D を使用した実装

金田 泰
日立製作所 システム開発研究所
〒215-0013 神奈川県川崎市麻生区王禅寺 1099 番地
E-mail: kanada@sdl.hitachi.co.jp

電話にかわるべき音声コミュニケーション・メディア voiscape の確立をめざして研究をおこなっている．Voiscape においては 3 次元オーディオ技術によってつくられた仮想的な“音の部屋”を使用するが，音声通信と 3 次元音声にくわえて 3 次元グラフィクスを使用するプロトタイプを PC 上に開発した．このプロトタイプにおいては音声のキャプチャと通信のために JMF (Java Media Framework), 3 次元音声 / グラフィクスのために Java 3D を使用した．開発前はこれらの API をつなげば必要な基本機能がほぼ実現できるとかんがえていたが，実際にはこれらを直接つなぐことはできず，3 次元音声のためには Java 3D のインタフェースをとおして OpenAL を使用した．また，プロトタイプにおいては音質劣化や遅延などの問題を容易に解決することができなかったが，試行をかさねてこれらの問題をほぼ解決した．

An Implementation of a Virtual “Sound Room” Based Communication-Medium Called *Voiscape* Using JMF and Java 3D

Yasusi Kanada
Hitachi, Ltd., Systems Development Laboratory
Aso-ku Ozenji 1099, Kawasaki, 215-0013, Japan
E-mail: kanada@sdl.hitachi.co.jp

The author researches toward establishing voice communication media called *voiscape* which shall replace telephone. A virtual “sound room” that is created by spatial audio technology is used in voiscape. We developed a prototype on PCs, in which 3-D graphic is used for supplementing spatial autio. In this prototype, JMF (Java Media Framework) was used for voice capturing and communication, and Java 3D was used for spatial audio and 3-D graphics. Before the development, the author had believed that the basic functions required for the prototype would be realized by connecting these APIs. However, in fact, they cannot be connected directly, so we used OpenAL through the interface of Java 3D. We also encountered problems of sound quality degradation and delay, but they have been almost solved by refining the program by trial and error.

1. はじめに

電話のユーザ・インタフェースは A. G. Bell が 1876 年に発明して以来，現在にいたるまでほとんど改善されることがなかった．それは電話網がかたいネットワークだったため，とくにそれが回線交換網だったために変更できなかったのだとかんがえられる [Kan 03a]．しかし，いまやかたい電話網はやわらかい IP ネットワークによって置換されようとしている．それによって電話のインタフェースを制約していた原因も消滅し，直接のコミュニケーションに匹敵する，人間の聴覚能力をいかした，多者間の自由な会話ができる，常時接続を前提とした新メディアによって電話はとってかわられるとかんがえられる．

このような状況のもとで，著者は電話にかわる音声コミュニケーション・メディアの一案として，3D (3 次元) オーディオによって実現される仮想の“音の部屋”を使用した音声コミュニケーション・メディアの開発をすすめている．作曲家 Murray Schafer が音がつくる風景を soundscape とよんだ [Mur 77] のにならって，このメディアを voiscape (声の風景，声景) とよんでいる．これまでに voiscape がみたすべき要件に関する考察 [Kan 03a] と，ポリシーにもとづいてプライバシー保護や通信量削減を実現する方式 [Kan 03b] と，開発した voiscape プロトタイプの概要 [Kan 03a, Kan 03b] などについて報告した．

このプロトタイプは Java によって実装した．音声の入出力と通信には Java SE (Standard Edition) の拡張 API である JMF (Java Media Framework) [Gor 98] [Fai 00] を使用し，3D グラフィクス表示には Java SE の拡張 API である Java 3D [Sow 00] を使用した．JMF はソース (音源・画像源など)，フィルタ，シンク (再生器や送信器) などの部品をくみあわせるだけで音声・画像などの基本処理ができるメディア処理 API である．また，従来の 3D グラフィクス API のおおくが描画レベルの API であるのに対して，Java 3D は 3D モデルを

記述して描画はシステムにまかせる高水準のインタフェースをそなえた3Dグラフィクス API である.

Java 3D は 3D オーディオ機能もあわせもっているため, 開発前にはこれらの API をつなぐだけで必要な基本機能がほぼ実現できるとかんがえていた. しかし実際にはこれらを直接つなぐことはできず, 機能も不足していたため, オーディオ API として OpenAL [Lok 00] [Cre 01b] を使用し, それを Java につなぐために LWJGL (Light-Weight Java Game Library, 4.3.1 節参照) という Java API を使用し, Java 3Dをとおして OpenAL を使用した.

この報告においては, Java を使用した voiscape のクライアントの実装について, とくに JMF, Java 3D, OpenAL, LWJGL をくみあわせる方法とその結果として生じた音質劣化や遅延をはじめとする問題点とその解決策についてのべる. 3D オーディオ・3D グラフィクスと音声通信・画像通信とのくみあわせは voiscape 特有のものではなく, おおくのマルチメディア応用において必要とされる機能である. また, JMF および Java 3D はとくに試作や実験において有用な API である. したがって, この経験は他のさまざまな応用にいかせるとかんがえられる.

2. Voiscape における会話のモデルとながれ

この章では voiscape における "音の部屋" の概念と会話のながれについてのべる. Voiscape においては, はなれた場所にいるユーザが仮想の "音の部屋" を使用して会話する. そのイメージを図 1 にしめす. ここには複数の "部屋" とよばれる仮想空間があり, そのなかの 1 個を選択できる. ただし, 部屋はかならずしもグラフィクスなどを使用してユーザに視覚的にみせるのでなく, 聴覚的にのみ存在させることも可能である. このメディアをうまく実現させられたとすれば, それを使用することにより, ユーザは部屋内を自由に移動して他のユーザに接近して従来の電話のような 1 対 1 の会話をすることもできるし, 3 人以上あつまっていわゆる井戸端会議や通常の会議をすることもできる. また, 部屋のなかで単独作業をしながら部屋の様子をうかがうこともできる. この空間のなかで各参加者は自由に自律的に移動できる.

クライアントにはあらかじめユーザ名が入力されている. クライアント起動時にこのユーザ名をつかってサーバに自動的にログインする. 3 章で説明するプロトタイプのクライアントのウィンドウを例として図 2 にしめす.

ログインするとサーバから部屋リストが送付される. 図 2 のウィンドウの左側にそのリストが表示されている. ユーザは部屋リストから部屋を選択して入室する. ユーザが部屋を選択して入室すると, 部屋内にだれがいるかがわかる. 部屋内の他のユーザとのあいだで自動的に音声通信が開始され, 仮想空間に位置づけられた他のユーザが 3D オーディオ・3D グラフィクスによって表示される. ヘッドセットを通じて 3D 音声がユーザの両耳につたえられる. 移動して話者にちかづけば

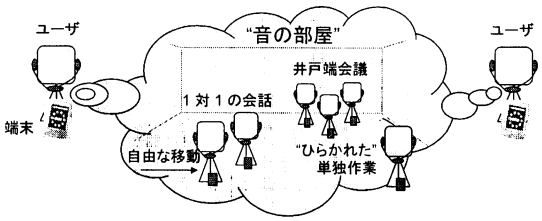

図 1 "音の部屋" のイメージ

声はおおきくきこえるし, 回転すれば声の方向がかわる.

人間は音声だけで話者がいる方向を的確に把握できないことがある. とくに, 聴覚による前後や上下のくべつは 3.1 節でのべる HRTF を使用してもなおあいまいである. そこで, オーディオ表示とあわせて, ユーザの周囲の様子をグラフィクスによって画面に表示する. 図 2 においてはユーザの前方が表示されている. ユーザ自身は表示されず, 部屋は床と壁とによって表現されている. ユーザ間の通信は音声だけにかぎられるので, ここでは他のユーザの画像は表示せず, 直方体と円錐とをくみあわせて他のユーザを表現している. これだけでは他のユーザのむきがわからないが, 直方体の上部にユーザの URI を表示して, それがだれであるかがわかると同時にむきがわかるようにした.

ポインティング・デバイスを使用して, 部屋のなかで自由に移動したり, むきをかえたりすることができる. プロトタイプにおいてはマウスを使用する. この移動や回転は仮想空間内のものであるから, 基本的に実世界における移動や回転とは無関係である. マウスの左ボタンによって前後にドラッグすれば前後に移動でき, 左右にドラッグすれば左右に方向をかえられる. 移動にともなって自動的に他のユーザとの通信を開始したり終了したりし, 声がききとれないようにしたりできる [Kan 93b]. マウスによって部屋内を移動するときは, 移動による位置や方向の変化をただちにそのユーザにフィードバックするため, 移動を検出するごとにそのクライアントのグラフィクス表示を更新する.

3. プロトタイプ

この章においては, 試作したプロトタイプの概要について説明する. 図 3 がその全体構成である. プロトタイプはおおきくわけるとサーバ群と複数のクライアントとで構成されている. クライアントも各サーバも, 実装を容易にするためにできるだけ Java によって記述し, 既存のフリーソフトを利用するようにした. 現在, クライアントは Microsoft 社の Windows XP または Windows 98 を搭載した PC 上で動作させている. サーバの実装については金田 [Kan 93b] がのべているので, ここでは省略する.

クライアントの構造は図 4 のとおりである. すなわち, クライアントは入力デバイスとしてマイクロフォンとマウスとをもち, 出力デバイスとしてイヤフォンまたはヘッド

図 2 クライアント・ウィンドウ

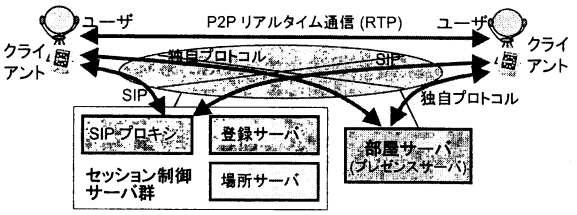

図 3 試作したプロトタイプの全体構成

フォンとディスプレイとをもつ．マイクロフォンからの出力は**音声入力部**によってディジタル信号にエンコードされる．この信号は通信相手が存在するときには**音声通信部**におくる．現在は 8000 Hz でサンプリングし，ITU-T の標準である G.711 u-law 64 kbps の信号として RTP (Real-Time Transport Protocol) [Sch 96] によって他のクライアントに P2P で送信する．他のユーザの音声信号は RTP によって受信し，3 次元オーディオ / グラフィクス表示部におくる．音声の入出力と RTP 送受信には JMF を使用した．

プレゼンスサーバとのあいだのメッセージはこのプロトタイプに独自のプロトコル [Kan 93b] によって**部屋モデラ**が送受信する．部屋モデラは位置指定デバイスからの出力をうけとり，ユーザの部屋内における位置をもとめて，独自プロトコルによってプレゼンスサーバに送信する．プレゼンスサーバからは他のユーザの部屋内の位置を受信する．

3 次元オーディオ / グラフィクス表示部においては，受信した RTP 信号を部屋モデラの情報にしたがって 3D 音場に位置づけ，その信号に対応するユーザのプレゼンスを Java 3D の 3D グラフィクスによって表示する．Java 3D には 3D オーディオ表示の機能も提供されているが，リアルタイム通信とくみあわせて使用できないことがわかったため，独自開発した Java 3D のオーディオ API (図 4 の JA3D) を JMF にはめこんで使用した．JA3D およびその下層の OpenAL, LWJGL に関しては 4 章において詳説する．サウンドカードとしては HRTF (Head Related Transfer Function) [Beg 00] 機能をもつものを使用している．HRTF 機能を使用すれば左右・前後の方向感がある程度えられるが距離感は再現されにくいので，残響を使用して距離感をえている．HRTF 機能についても 4 章において詳説する．

セッション制御部においては SIP プロキシを経由して他のクライアントとのあいだの RTP 通信の開始・終了等を制御する [Kan 93b]．**ポリシー制御部**はセッション制御部の機能を制御するポリシーを保持し，それにもとづいて通信を制御する [Kan 93b]．

4. クライアント実装上の問題点とその解決策

プロトタイプのクライアントを実装する際，最初は Java API だけによって，すなわち JMF と Java 3D だけをくみあわせて必要な機能を実現しようとしたが，この方法はうまくいかなかった．そこで PC サウンド API を使用する方法すなわち OpenAL と LWJGL をあわせて使用する方法によって実現した．この章においてはまず Java API だけによる実装の問題点についてのべ，その後 PC サウンド API を使用する方法とそこで発生した問題点とその解決策についてのべる．

4.1 Java API だけによる実装の問題点

プロトタイプ開発前には JMF と Java 3D とをつなげば必要な基本機能がほぼ実現できるとかんがえていた．しかし，実際にはこれらの API は独立に仕様がきめられていて，うまく連携して動作しないことがわかった．[2] また，Java 3D には 3D 空間中に音源を定位させるとともに複数の音源を配置するための 2 種類のプログラム (以下 3D ミキサとよぶ) が用意されているが，これらはいずれもこのプロトタイプへの使用には適さないことがわかった．これらの問題について，以下くわしく説明する．

まず第 1 の問題点について説明する．Java 3D においては音源を表現するためのクラスが用意されている．点音源のばあいは PointSound クラスを使用する．そのインスタンスを生成するとき，音をあらわすために MediaContainer クラスのオブジェクトを指定する．このオブジェクトは 3 種類の表現形式すなわち汎用のストリームである InputStream クラスのオブジェクトと URL クラスまたは文字列によってあらわされた URL をふくむことができる．しかし，いずれも JMF の RTP によって入力された音声には使用できない．[3] この問題は，もし Java 3D の仕様を変更することができるなら，RTP 形式の URL の指定を許可するだけで解決できる．しかし，それをうまく実装するのは容易でないとかんがえられる．

つぎに第 2 の問題点について説明する．Java 3D には JavaSoundMixer および HeadSpaceMixer という 2 つの音声 3D 化のためのクラスが用意されている．InputStream クラスを使用すればこれらをそのまま使用できるはずだったが，実際には使用できなかった．まず，JavaSoundMixer クラスのミキサを使用すれば RTP によって受信した音を再生できたが，このミキサには音を 3D 化しないという致命的な問題点があることがわかった．また，HeadSpaceMixer は音声をすべてキャッシュしてから再生するため，ストリーム再生には使用できないことがわかった．

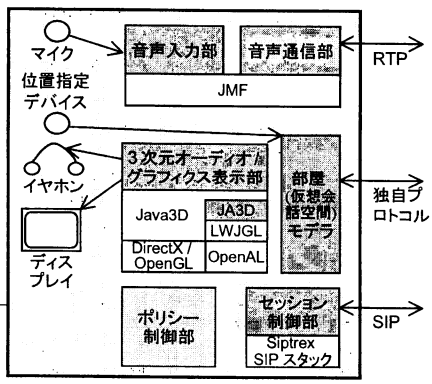

図 4 クライアントの構造[1]

4.2 第 1 の問題点の解決策

上記の第 1 の問題点を解決し

[1] DirectX (Microsoft 社の登録商標) と OpenGL (Silicon Graphics 社の登録商標) はグラフィクス API, OpenAL はオーディオ API である．

[2] 現在では Java 3D のかわりに SUN 社によるオープンソースの JOGL (Java bindings for OpenGL) API を使用すればもっとうまくいくかもしれないが，voiscape のプロトタイピングをはじめた時点では使用できなかった．

[3] すなわち，URL としては静的なファイルの URL が想定されていて，RTP 形式 (rtp:... の形式) の URL はゆるされていないので，RTP のストリームは直接指定できない．また，JMF から通常の方法で出力できる音の形式として InputStream クラスは存在しないので，それによってもつなげない．

てJMFとJava 3Dとを結合するため，2方法を考案した．

[案1] InputStreamを実装する方法: JMFによって受信したRTPデータをInputStreamのデータに変換する，Rendererインタフェースを実装したクラスを開発する．

[案2] RTPからの出力にRTPストリームを使用する方法: 音声をRTP専用の低位のストリーム(RTPSourceStreamクラスのオブジェクト)から直接入力する．

まず案1をこころみた．案1にしたがったJMFとJava 3Dのくみあわせ方法については，ビデオに関する例がSunから提供されている[Sun 03]．[1] このプログラムをオーディオ信号用にかきかえてRTPからの出力に適用した．

この方法においてはsetRendererメソッドによってJMF既定のレンダラ(再生器)を新規開発した専用レンダラに置換する．RTPのデータが到着するとこのレンダラのprocessメソッドがよばれるが，ここでデータをInputStreamインタフェースを実装したAudio3DInputStreamクラスのオブジェクトに変換する．データはこのオブジェクト内でバッファされ，Java 3D内からreadメソッドを使用してよみだされる．この案においてはAudio3DInputStreamのオブジェクトをMediaContainerクラスのオブジェクトに格納し，Java 3Dに通常の方法でわたすことができる．

しかし，案1は結局は放棄することにした．それは，ストリームの再生のために余計なスレッドを介在させることになり，オーバヘッドがおおきく音質の劣化がおおきかったからである．つまり，この方法においてはすくなくともRTP受信用，InputStreamへの変換用，3Dオーディオ再生用という3個のスレッドが必要になるが，このうち第2のスレッドは案2においてはなくすことができる．

つぎに案2をこころみた．この案ではMediaContainerクラスのオブジェクトが，本来はふくむことができないRTP形式のURLをふくむオブジェクトを"ユーザデータ"として(非標準形式で)ふくむようにする(MediaContainerクラスにはこのようなインタフェースが用意されている)．3Dミキサ内でこのURLをとりだしてひらき，PushBufferDataSource型のオブジェクトをとりだす．[2] このオブジェクトのクラスはcom.sun.media.rtp.RTPSourceStreamであり，readメソッドを使用すればデータがとりだせる．このデータをLWJGLのAPI経由でサウンドカードにおくりこめばよい．

案2においてはJMFが提供するインタフェースのなかでは下位のものを使用する．そのため，この方法においてはG.711のデコードは自前でおこなう必要が生じるが，このデコードは単純で計算量もすくないため，あまり問題はない．したがって，現在も案2を使用している．

4.3 第2の問題点の解決策

第2の問題点を解決するため，最初はキャッシングしないHeadSpaceMixerのサブクラスを定義しようとしたが，うまくいかなかった．そこで，OpenALMixerというあらたな3Dミキサを開発した．このミキサにおいてはJava 3Dの高水準APIインタフェースはそのまま使用しつつ，その下位においてはくみこみ実装のかわりに外部のAPIすなわちOSや市販のPCサウンドカード用に用意されたAPIやHRTF機能を使用している．そこで，この節ではまずこれらついて説明してから，プロトタイプにおける選択についてのべる．

4.3.1 DirectSound, OpenALとその拡張API

3Dオーディオはおおくの CPUマザーボードやオーディオカードにおいてサポートされ，ゲームなどに使用されている．しかし，グラフィクスに関してはOpenGLという，複数のプラットフォームにまたがってひろく使用されている標準APIがあるのに対して，オーディオにおいてはこのようにひろく使用されているプラットフォーム独立なAPIが存在しない．Windows PC上のオーディオAPIとしては，MicrosoftのDirectSoundがある．DirectSoundの3Dオーディオの部分がDirectSound 3Dである．

DirectSoundおよびDirectSound 3Dが提供する機能はゲームなどにおいて使用するのに十分ではないとかんがえられてきた．そのため，Windows PCのためのサウンドカードなどの市場をリードする製品を提供してきたCreative Technology社は残響機能や物体による音の反射・回折などをシミュレートするDirectSoundの拡張APIとしてEAX (Environmental Audio Extensions) [Cre 01]を開発した．EAXには基本的な機能を実現するレベル1から高度な機能を実現するレベル3までがあり，さらに付加機能がある．[3]

これらのAPIはいずれもベンダ固有のものであるため，IASIG (Interactive Audio Special Interest Group)という業界団体がつくられて，3Dオーディオの標準機能としてI3DL2 [IAS 99]をさだめ，Loki Entertainment Softwareというゲーム開発ベンダを中心としてI3DL2にもとづくAPIであるOpenALが開発された．[4] EAXはOpenALとくみあわせて使用することもできる．

上記のAPIのうち，Java 3D以外はいずれもJavaから直接使用できない．そこで，OpenGL, OpenALおよびEAXをJavaから利用するために，LWJGL (http://java-game-lib.sourceforge.net/)というAPIがSource ForgeのLWJGLプロジェクトによって開発された．LWJGLによってOpenALをJavaから使用することができる．OpenALによって左右および前後の方向感がえられるし，OpenAL経由でEAXを使用すれば残響や他の環境パラメタも設定できる．

4.3.2 HRTFの実現方法

A3D以外の上記のAPIはHRTFの機能をとりいれていないが，Sensauraというベンダが HRTFをPC上で表現するためのライブラリ(DLL)を提供している．このライブラリを使用すれば，通常のDirectSoundやOpenALのAPIをよびだすだけでHRTFの機能を使用できる．Sensauraのライブラリは特定のオーディオ用DSP (Digital Signal Processor)と連携して動作するため，どのようなハードウェアでも動作するわけではない．Sensaura以外ではQSound社のQ3D技術がHRTF機能をサポートしている．SensauraのHRTF機能がくみこまれたDSPの例としてAnalog Devices社の

[1] このプログラムを使用すれば3Dグラフィクス・オブジェクトの表面にJMFから出力される動画をはりつけられる．ただし，このプログラムでは動画入力にRTPでなくMPEGファイルを使用している．

[2] この方法では3Dミキサをカスタマイズする必要があるので，Java 3Dくみこみのミキサをそのまま使用することはできない．

[3] また，半導体ベンダであるAureal社は同様の機能をもつA3DというAPIを開発している．A3Dは現在も使用されているが，Aureal社が解散したのでサポートされていない．

[4] この会社もすでに解散しているが，OpenALの原始プログラムおよびWindows版とMacintosh版のバイナリ・ファイルはCreative Technology社からひきつづき配布されている．

AD1985[1], Cirrus Logic 社の CS4630, C-Media 社の CMI-8738, ヤマハの YMF-754, RealTek 社の ALC203 などがあり, Q3D を使用したものとして Philips Semiconductors 社 (旧 VLSI 社) の SAA7785 Thunderbird Avenger などがある. [2]

4.3.3 プロトタイプにおける選択

DirectSound 3D を使用すれば現在では左右の方向感はもちろん残響も制御できるので, voiscape が必要とする機能をおよそみたせる. ところが, DirectSound 3D は Microsoft Windows のウィンドウシステムの仕様を前提としているため, Java とは相性がわるいことがわかった. すなわち, DirectSound 3D においては初期化のための関数に対してウィンドウ・ハンドルをわたさなければ動作しないが, Java ではウィンドウ・ハンドルを使用しないので初期化できない. この問題をさけるため, プロトタイプにおいては LWJGL を介して OpenAL を使用することにした.

4.3 節の冒頭においてのべたように, 3D ミキサとしては OpenALMixer を独自に開発して使用している. このミキサを中心とするプログラムを 3 章において JA3D とよんだ. OpenALMixer は JavaSoundMixer や HeadSpaceMixer と同様に AudioDevice3DL2 (I3DL2 標準にしたがうオーディオ・デバイス) というインタフェースを実装している. すなわち, インタフェースは従来のミキサとあわせながら, OpenAL を使用して実装している.

また HRTF に関しては, 前節において列挙した Sensaura のライブラリ使用のチップのいずれかを使用したサウンドカード数枚をためしたが, おもに使用しているのは CMI8738 使用のカードである.

4.4 PC サウンド使用実装上の現象と解決策

前節でのべたようにプロトタイプにおいては JMF, Java 3D と OpenAL, LWJGL, Sensaura の HRTF をくみあわせて使用することにしたが, そこで発生した現象とその解決策についてのべる.

4.4.1 音質劣化と遅延

プロトタイプにおけるもっとも解決困難な問題が音質劣化と遅延だった. 上記のくみあわせを実現した最初の版においては, CODEC として G.711 を使用すればかろうじてききとれる音声が再生されるが, CPU 時間はまだ十分に余裕があるにもかかわらずバッファあふれが生じるようで, 音質はきわめてわるかった. また遅延も最大で 6 秒ほどあった. よりサンプリング周波数のたかい CODEC を使用すると, 音声はまったくききとれなくなる. その原因はわかっていないが, スレッドのスケジューリングが関与している可能性がたかい. [3] プロセスやスレッドの優先度をかえてもこれらの問題は解決できなかった. [4]

遅延に関していえば, 3D オーディオ処理をおこなわなければおおきく遅延することはないようである. 遅延は送信側でも発生し, すべてのバッファがうまると 6 秒に達するのであろう. [5,6]

発生している現象をまとめると, つぎのとおりである.

- 遅延が最大のとき音質は比較的よい. (ジッタが発生してもデータがおちないからであろう.)
- 遅延が最小のときは音とびが発生しやすく, 雑音もおおい. 最悪のばあいはききとることができなくなる. ただし, バッファ・サイズを調整することで音とびはほぼなくすことができた.
- サウンドカードの種類によって遅延が発生しやすいものと発生しにくいものとがある. EAX や HRTF の機能をもたないサウンドカードを使用するかそれらの機能を停止したときは問題がおこりにくい.

今後とりいれるべき方策として静寂検出 (silence detection) がある. 静寂検出とは, 話者が話をしていないことを検出して, その間は音声の送信をとめる方法であり, 音声通信を安定させ遅延をふせぐのに有効だといわれている [Jac 99]. Voiscape においては音声によって話者のプレゼンスをつたえることも意図している. 静寂検出はそれを不利にするとおもわれるが, 他には適当な方法がみあたらない.

4.4.2 JMF と JavaSound に関する他の問題

Java 3D と JMF とのくみあわせによって発生する問題についてはすでにのべたが, JMF 単独および JavaSound に関して下記のような問題点があった.

- **低水準の部品との接続困難性**: JMF はかなり閉鎖的にできている. JMF は部品をくみあわせるだけでオーディオ, ビデオのキャプチャから通信, 再生までのメディア処理ができるため, その RTP 機能なども大学教育などでひろく使用されている. 既存の部品をただくみあわせるだけの処理は非常に容易に実現できる半面, それだけでできない処理をするとき低水準の部品をつなぐのが困難である. すなわち, 既存の部品に低水準の入出力をうけいれる方法がないか, または "プラグイン" のしくみがあってもドキュメントがほとんどない. たとえば, RTP によって受信する音声をあつかうプラグインへの入力がすでにデコードされているのか, パケットヘッダが除去されているのかどうか, などの点が実際に出力をみるまで不明だった. しかし, 実用的な通信のためには JMF くみこみの機能だけでは十分ではないので, 拡張の必要性はたかい.
- **使用可能な CODEC のせまさ**: さまざまな CODEC を指定して動作させてみた結果, 非常にかぎられた CODEC しか動作しなかった. ただし, 動作しないもののなかには JMF 付属の JMStudio というプログラムにおいては動作するものもあるので, 使用法の問題である可能性もある.
- **RTP の遅延のおおきさ**: JMF における RTP の実装は実

[1] これらのチップが使用するオーディオ技術は Analog Devices 社の SoundMax という商標によってしられている.

[2] 以上の DSP やその HRTF サポートが PC へのくみこみを目的としているのに対して, 音楽などのより高音質のソリューションをめざした製品もある. その代表が Wave Arts 社による WaveSurround 技術である. PC 用のソフトウェアも開発されているが, DSP くみこみのオーディオ機器などに採用されている.

[3] プロトタイプにおいては約 70 個のスレッドが生成される. その大半は Java 3D の内部で生成される.

[4] プログラムを分解して複数のプロセスによって実行させれば問題を解決できる可能性があるが, まだためしていない.

[5] 送信側でも発生していることは, 送信側で再生してみてわかった. また, 受信側で遅延発生が分散していることは, 何カ所かで信号をかんたんにモニタしてみてわかった.

[6] この遅延は送信側の PC に CPU 負荷をかけると解消されることがある. すなわち, 負荷をかければ信号 / パケットの発生がおさえられ, バッファにたまった音声データがクリアされて遅延が最小限になる. しかし, 負荷のかけかたによっては逆に遅延が拡大する.

用上かならずしも十分な品質ではなく，とくに遅延が比較的おおきいといわれている．

- **Linux 版における音声キャプチャの問題点**: プロトタイプを Windows から Linux に移植することをこころみた．ところが，Windows 版の Java で動作したキャプチャ機能はそのままでは Linux 上で動作させることができず，Java の利点であるプラットフォーム独立性はいかされなかった．[1]

4.4.3 Java と OS のリアルタイム性

Microsoft 社の OS は Windows NT 以降リアルタイム性にも注意がはらわれているが，今回使用した Windows XP においても対策は十分とはいえない．また，Java はリアルタイム処理に不向きな言語だといわれている．その理由としてガーベジコレクションの問題がある．プロトタイプにおいてはできるだけオブジェクトを再利用してごみがでないようにしているが，それでもガーベジコレクションが原因とみられる音声再生の中断が発生することがある．また，4.4.1 節でのべたようにすべての処理を 1 個のプロセスのなかにつめこんで多数のスレッドを使用していることもリアルタイム性を阻害している可能性がある．

5. 実装法の評価

この章では voiscape プロトタイプの実装において使用した Java による 3D オーディオ・3D グラフィクスと音声通信などとのくみあわせ方法を定性的に評価する．

この方法により，Java 3D 本来のインタフェースの最低限の変更によって OpenAL が使用できるようになった．すなわち，この方法では第 1 に OpenALMixer を JavaSoundMixer や HeadSpaceMixer にかえて使用するだけで OpenAL を使用できる．第 2 に，OpenALMixer を使用すれば，RTP 形式の URL を MediaContainer オブジェクトに非標準形式で格納することによって，Java 3D に RTP ストリームがわたせる．ここで RTP 形式の URL が指定できるようにするのが最善だが，それができない現状ではこの変更はやむをえない．

ただし，現在このインタフェースは十分汎用的にはなっていない．汎用インタフェースにするには，RTP にくわえて OpenALMixer においても JavaSoundMixer や HeadSpaceMixer があつかえるすべての形式のファイルやストリームをあつかえるようにする必要がある．また，現在は他にも OpenALMixer には制約があるので，それをなくすか，またはゆるめる必要がある．

6. 結論

Voiscape のプロトタイプにおいては JMF, Java 3D を中心とし，LWJGL を介して OpenAL によって 3D 音声再生機能を実現した 3D ミキサ OpenALMixer を開発し，使用した．音質劣化や遅延などの問題を容易に解決することができなかったが，試行をかさね，これらの問題を軽減させた．OpenALMixer を改良すれば 3D オーディオ・3D グラフィクスとメディア通信とをくみあわせた多様なアプリケーションの試作や実験のために使用できるであろう．改良するべき点としては，のこされた遅延の発生をなくすこと，さまざまなデータ形式に対応して汎用性をたかめることなどがある．

[1] プロトタイプのキャプチャ機能だけだけでなく，JMF 付属の JMStudio というプログラムのキャプチャ機能も，すくなくとも Java SE 1.4.2, JMF 2.1.1e というくみあわせでは Red Hat Linux 上で動作しないことを確認した．

参考文献

[Beg 00] Begault, D. R., "3-D Sound for Virtual Reality and Multimedia", NASA/TM-2000-XXXX, NASA Ames Research Center, April 2000, http://human-factors.arc.nasa.gov/ihh/spatial/papers/pdfs_db/Begault_-2000_3d_Sound_Multimedia.pdf

[Bol 00] Bollella, G., et al., "The Real-Time Specification for Java", Addison-Wesley, 2000. http://www.rtj.org/rtsj-V1.0.pdf.

[Cre 01a] Creative Technology, "Environmental Audio Extensions: EAX 2.0, Version 1.3", http://www.sei.com/-algorithms/eax20.pdf.

[Cre 01b] Creative Technology, "Creative OpenAL Programmer's Reference Version 1.0", 2001.

[Fai 00] Faiman, N., Giese, D., Rokanuzzaman, A., and Schroeder, M., "A Survey of the Java Media Framework 2.0", CSci 532 : Programming Languages and Paradigms, University of North Dakota, http://www.cs.und.edu/-~mschroed/cs532/survey.doc.

[Gor 98] Gordon, R. and Talley, S., "Essential JMF – Java Media Framework", Prentice Hall PTR, November 1998.

[IAS 99] The Interactive Audio Special Interest Group, "Final IA-SIG 3D Audio Rendering and Evaluation Guidelines Level 2", http://www.iasig.org/wg/closed/3dwg/-3dl2v1a.pdf.

[Jac 99] Jacobs, S., Eleftheriadis, A., and Anastassiou, D., "Silence Detection for Multimedia Communication Systems", *Multimedia Systems*, Vol. 7, pp. 157–164, 1999.

[JCP 03] "JAIN-SIP API Specification Version 1.1", http://jcp.org/aboutJava/communityprocess/final/jsr032/-index2.html, Java Community Process, 2003.

[Kan 93a] 金田 泰, "仮想の '音の部屋' によるコミュニケーション・メディア *Voiscape*", 電子情報通信学会 技術研究報告 (MVE / VR 学会 EVR 研究会), 2003-10-7.

[Kan 93b] 金田 泰, "仮想の '音の部屋' によるコミュニケーション・メディア *Voiscape* におけるポリシーベース・セッション制御", 電子情報通信学会技術研究報告 (IA / IRC / 情報処理学会 QAI 研究会), 2003-10-8.

[Lok 00] Loki Software, "OpenAL Specification and Reference Version 1.0 Draft Edition", 2000.

[Mur 77] Murray Schafer, R., "The Tuning of the World", 訳書: 鳥越 けい子他訳, "世界の調律", 平凡社, 1986.

[NIS] "Project: Internet Telephony / VOIP", http://www-x.antd.nist.gov/proj/iptel/ .

[Roa 02] Roach, A. B., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 2543, IETF, June 2002.

[Ros 02] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E., "SIP: Session Initiation Protocol", RFC 3261, IETF, June 2002.

[Sch 96] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, IETF, January 1996.

[Sow 00] Sowizral, H., Rushforth, K., and Deering, M., "The Java 3D™ API Specification (2nd Edition)", Addison-Wesley, May 2000.

[Sun 03] "Render Live Video on a 3D Surface", *JMF Solutions*, http://java.sun.com/products/java-media/jmf/2.1.1/-solutions/DemoJMFJ3D.html .