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

2006－MUS－64（3）
2006／2／23

Max/MSP における JoGL の実装

濱野　峻行
国立音楽大学
hamano@kcm-sd.ac.jp

概要: Max/MSP version 4.5 でリリースされた mxj オブジェクトにより、Max 内で Java 言語を扱えるようになった。この mxj オブジェクトを利用し、JoGL による画像生成・処理プログラミングを Max パッチ内で実現させた。本稿ではその実例を紹介し、今後の可能性について考察する。

Applying the JoGL technology in Max/MSP

Takayuki Hamano
Kunitachi College of Music
hamano@kcm-sd.ac.jp

Abstract: New Max object “mxj” was released in the Max/MSP version 4.5. Now we can include the Java programming in Max patch. Using this new feature I have implemented JoGL programming environment, which is Java bindings for OpenGL. In this paper, I would like to introduce applying the JoGL technology in Max/MSP.

1. はじめに

現在、インタラクティブ・マルチメディア作品で用いられるリアルタイム映像処理・生成ツールとしては、主に DIPS、Gem、Jitter、Quartz Composer などが挙げられる。いずれもユーザビリティの点では優れているが、それらの機能には必然的に限界がある。その限界を超える一つの手段として、筆者はここに Max/MSP 上で mxj オブジェクトと JoGL を用いた映像プログラミングを紹介する。JoGL を用いることで、最先端のコンピュータグラフィックス技術を Max/MSP 内で活用することができるようになる。

2. mxj について

2.1 mxj とは

Max/MSP version 4.5 のリリースでは様々な新たなオブジェクト・機能が追加された。mxj オブジェクトもその一つである。mxj は Max から Java Virtual Machine を呼び出し、Java 言語のクラスファイルを実行するためのものである。このオブジェクトにメッセージ"viewsource"を送ると Java ソースコード編集用のエディターが別ウィンドウに表示され、コードの編集、さらにコンパイルが可能になる(図 1)。なお mxj オブジェクトの他に mxj~というオブジェクトもあるが、このオブジェクトでは音声信号処理プログラミングをも扱うことができる。

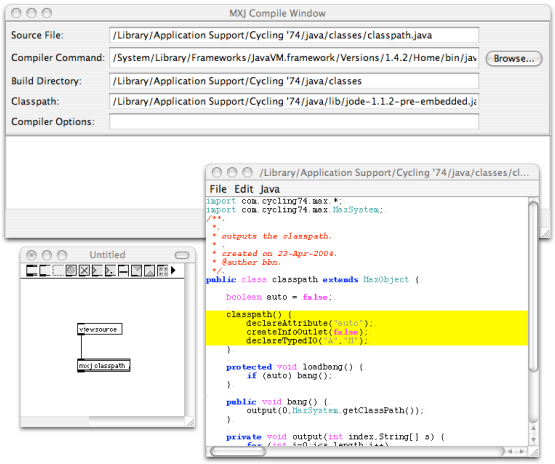

図 1　Max/MSP の mxj オブジェクト

2.2 Java 言語の有用性

Max/MSP で Java 言語を扱うことの意義を、以下に述べる。

(1) Max プログラミングとコーディング

Max はオブジェクトをパッチコードで繋ぐことで視覚的にプログラミングができる GUI プログラミング環境である。ソースコードを書かずにマウスでプログラムを組めるが、配列を扱いにくく、複雑な条件分岐を組み合わせるとデバッグが煩雑になるなどの弱点がある。それらを克服するため Max には多様なオブジェクト・機能が次々に追加されてきたのだが、使いこなすには独特のテクニックの習得が必要となる。そこで、mxj による Java プログラミングを Max パッチ内で併用することにより、弱点の相互補完を実現できる。またユーザーは随時、自分の得意とする環境を選択することができる。

(2) JavaScript と Java 言語

Max オブジェクト js は、JavaScript を実行させるためのものである。JavaScript はスクリプト言語の一種でわずかなプログラミングの知識でも書くことが可能であり、簡単な演算を行うのには適している。しかし、手軽な反面、扱えるデータ型・命令が数や文字列に関する基本的なものしかないことと、スクリプト言語であるため動作速度が遅いことが欠点である。それに対し Java 言語は JavaScript ほどの簡便さはないものの、オブジェクト指向の言語であり他の高級言語と比べても簡潔かつ明快に構成されている。

(3) エクスターナルオブジェクトと mxj

通常 Max のエクスターナルオブジェクトのプログラムは C 言語で書かれる。従って、エクスターナルオブジェクトを書くには C 言語に関する熟練した知識が必要となり、一般ユーザーには敷居の高いものである。一方、mxj では、エクスターナルオブジェクトを書く上で必要とされる多くの記述が自動的にサポートされているため、エクスターナルオブジェクトを作成するのに比べて手軽にカスタムオブジェクトを作ることができる。

3. JoGL

3.1 JoGL とは

JoGL（Java Bindings for OpenGL）は、3 次元グラフィックスライブラリ OpenGL を Java 言語で扱えるようにしたものである。OpenGL はシリコン・グラフフィクス社によって開発された API である。

JoGL は AWT や JFC/Swing と統合されることにより、Java での OpenGL の表示を実現している。Windows や Macintosh はもちろんのこと、Red Hat Linux や Solaris などのプラットフォームでも利用できる。OpenGL 2.0 の機能に全てアクセスすることが可能なので、OpenGL Shading Language (GLSL)も使用可能である。また、NVIDIA 製 GPU に対応した API である Cg 言語も実装されている。この GLSL と Cg 言語によって、現在コンピュータグラフィックスの世界で注目されているシェーダープログラミングも扱うことができる。

3.2 mxj 拡張による独自環境 MJGL

JoGL をシステムに追加することによって通常 mxj オブジェクトは JoGL を使用することができる。しかし、この環境では非常に長いソースコードを毎回、記述しなくてはなら

ず利便性に優れない。そこで、筆者はMax/MSP 上でより容易に JoGL テクノロジーにアクセスできるようにするため、mxj を拡張し、GUI を搭載した MJGL というオリジナルのプログラミング環境を作成した(図2)。MJGL は、テキストエディタに JoGL のための機能を追加したプログラムである。

MJGL では、mxj オブジェクトを作成し、第 1 アーギュメントに"MJGL"、第 2 アーギュメントに扱う Java クラス名を与える。これにより MJGL の GUI が起動し、Java のソースファイルが読み込まれ、コーディング・編集用のテキストエリアが表示される。同時に、JoGL の簡易化されたメソッドがヘルプと共にウィンドウの左側に表示され、ユーザーはマウス操作によっても該当するメソッドをソースコードに追加することができる。Compile&Reload ボタンを押すと、ソースコードが保存・コンパイルされて自動的にクラスファイルが再実行される。このクラスファイルの動的リロードには、Java 言語のリフレクション(Reflection)のテクニックを用いた。また、mxj オブジェクトのインレットからの数値入力、アウトレットへの出力にも対応させた。

以下にこの MJGL 環境を利用して書かれたソースコード例を示す。

```
public class myCode extends MJGL {
  public void init() {}
  public void bang() {
    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();
    gluPerspective( 45, 1.333f, 0.01, 100 );
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    glColor4f( 1f, 0f, 0f );
    glTranslatef( 0f, 0f, inlet[ 0 ] );
    glutWireTetrahedron();
  }
}
```

このソースコードは、赤い正四面体を表示するプログラムである。mxj の第 1 インレットに bang が送られてきたときに再描画するようにプログラミングされている。また、第 1 インレットに数値を送ることで、正四面体の位置を制御できるようにしている。このように MJGL では、OpenGL の命令をほぼそのまま記述することが可能である。

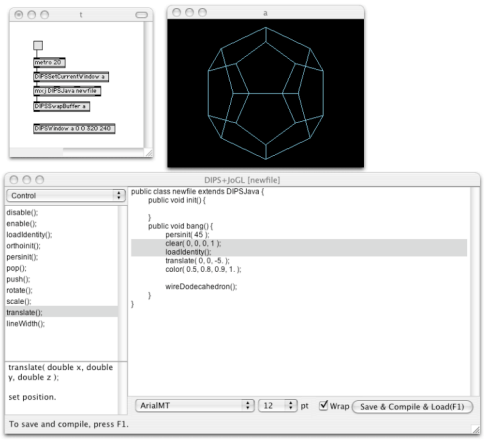

図 2　アプリケーション MJGL の GUI

3.3 MJGL の活用例

先のオリジナル・アプリケーション MJGL を用いて、JoGL とその他の Java テクノロジーを統合した使用例を以下に示す。

(1) Java2D による文字列描画

もともと OpenGL 自体には文字列描画の機能はなく、今までは各種プラットフォームに依存した独自のライブラリを利用するしかなかった。MJGL では Java2D を併用することにより、プラットフォームに依存することなく OpenGL で生成される画面上に文字列を描画できるようにした。Java2D は 2 次元グラフィック描画のための Java の API 群であり、その独自の機能によりフォントのアンチエイリアスやアウトラインの抽出も可能である。

今回は Java2D のオフスクリーン描画領域で文字列を描画し、そのピクセル情報に基づいてテクスチャを生成するという方法をとった。これにより、描画した文字列を 3 次元上で扱うことができるようにした。

(2) JMF によるムービーファイル作成

JMF(Java Media Framework API)は、タイムベースでオーディオやビデオなどを扱うための Java テクノロジーである。この機能を用いて、JoGL の描画画面を QuickTime の MOV 形式ムービーに記録できるようにした(図 3)。保存したムービーを再利用することで、表現の拡張を図ることができる。現時点では一旦 Jpeg ファイルに保存してから MOV 形式に一括変換するという手法をとっているが、今後、直接 MOV 形式にリアルタイム変換できるよう開発を進めている。

(3) DIPS3 との併用

DIPS3 (Digital Image Processing with Sound)は Max/MSP のエクスターナル・オブジェクトとして実装されてるリアルタイム映像処理・生成環境である。DIPS3 の DIPSWindow オブジェクトにより OpenGL 描画ウィンドウを作成し、そこに JoGL で書かれた画像を描画することに成功した。これにより DIPS に未実装の OpenGL 命令を実行したり、DIPS3 プログラミングで困難を伴う部分をアシストすることが可能になる。

4. まとめ

今回、この開発を行う上で、いくつかの問題点が浮かび上がってきた。

(1) mxj 自体の不備

通常の Java 環境では動作するソースコードにも関わらず mxj に移植するとフリーズするといった事象がたびたび発生した。また mxj には Max エクスターナルオブジェクトで言うところの free 関数に該当するものがない。すなわち Max オブジェクトを消去したりパッチを閉じたりしたときに発生するイベントを検出することができない。このように mxj 自体にはまだ不十分な点がいくつか見られた。バージョンアップ、アップデートでの改善を期待したいところである。

(2) Java の実行速度

Java の動作速度に関しては、今回の開発では特に問題を感じなかった。今後 JIT コンパイラの研究が進み、起動時に Java バイトコードを一挙に最適化・ネイティブコードに変換して実行することができるようになれば、より複雑な JoGL プログラミングでも実行速度に関する問題は起こらないであろう。

(3) 今後の可能性

2006 年夏頃には、Java の次期バージョンである Java SE 6(開発コード Mustang)がリリースされる予定である。そこでは JoGL と Java2D が本格的に統合されることになっている。この統合により、OpenGL の画面上に Java2D で図形や文字を描画することが容易となる。JoGL はこれからますます一般化されていくものと考えられる。MJGL も Java SE 6 に随時対応させていくと同時に、新機能を実装させ、拡張を試みていく予定である。

参考文献

[1]OpenGL Architecture Review Board, Maison Woo, Jackie Neider, Tom Davis: “The Official Guide to Learning OpenGL”

[2]Sun Microsystems, Inc., “Java Media Framework API(JMF)” [http://java.sun.com/products/java-media/jmf/]

[3]Miyama,C, Rai,T, Matsuda,S, Ando,D: “Introduction of DIPS Programming Technique”, in Proceddings of the International Computer Music Conference 2003.

[4]Matsuda,S, Miyama,C, Ando,D, Rai,T: "DIPS for Linux and Mac OS X", in Proceddings of the International Computer Music Conference 2002.

[5]Matsuda,S, Rai,T: "DIPS : the real-time digital image processing objects for Max environment", in Proceddings of the International Computer Music Conference 2000.