Javaを用いたWeb3Dのための3次元グラフィックス及びVRMLライブラリの開発

陳　炳宇[†]　　西田　友是[††]

The Development of 3D Graphics and VRML Libraries for Web3D Platform by Using Java

Bing-Yu CHEN[†] and Tomoyuki NISHITA[††]

あらまし　この論文はインターネット上での Web3D のための3次元グラフィックスに関する新しいプログラミング環境について提案する．インターネット上で動作する3次元グラフィックスのプログラムの開発は，OpenGL のように普及した3次元グラフィックスツールがないので容易ではない．そのため，筆者らは Java のみを用いて jGL と呼ばれる3次元グラフィックスライブラリを開発した．jGL は汎用の3次元グラフィックスライブラリであり，その API（Application Programming Interface）は OpenGL の API と同様に定義される．また，OpenGL と同じ形式で記述できるので，プログラマは簡単に利用できる．最近，携帯電話上で Java プログラムが実行可能になったので，iAPPLI への移植実験も報告する．更に，VRML（Virtual Reality Modeling Language）はインターネット上での標準的な3次元モデルのファイル形式である．インターネット上で3次元モデルを表示するとき，VRML ファイル形式を用いることが多い．そのため，jGL の API を拡張して，Java のみを使って，jVL と呼ばれる VRML ライブラリを開発した．

キーワード　OpenGL，VRML，Web3D，Java，グラフィックスライブラリ

1. まえがき

近年，インターネットを利用するユーザが増えている．その上，インターネット上での3次元グラフィックスの表示に注目が集まっている．しかし，このような3次元グラフィックスのプログラムを開発するとき，インターネット接続されたマシンは同じ種類とは限らないので，異なったマシンで実行可能なシステムの開発は容易ではない．すなわち，これはプラットホーム依存の問題を解決しなければならないことを意味している．現在，Java[1] はこの問題を解決できるが，Java を用いる場合 Java の3次元グラフィックスのサポートが問題となる．

一般に，コンピュータ上で3次元グラフィックスのプログラムを開発するとき，プログラマは OpenGL[2] のような汎用の3次元グラフィックスライブラリを利用できると便利である．しかし，Java を用いる場合 OpenGL のような3次元グラフィックスライブラリは存在しない．更に，そのような3次元グラフィックスライブラリは簡単に習得し利用できる必要がある．そのため，筆者らは OpenGL の API を参考にし jGL と呼ばれる3次元グラフィックスライブラリの API を設計した．この API を用いることで，ユーザが新たな3次元グラフィックスライブラリの API を学習する必要がなくなり，インターネット上で動作する3次元グラフィックスのアプリケーションを開発することが以前より容易になると思われる．

Java3D などのインターネット向けの3次元グラフィックスライブラリは OpenGL のようなライブラリをインストールする必要がある．それと異なり，jGL では Java のみを用いて開発したので，前もってこうしたライブラリをインストールする必要がない．jGL を用いてインターネット上で3次元グラフィックスのアプリケーションを開発するとき，プログラマは Java VM（Virtual Machine）中の3次元のグラフィックス

[†] 東京大学大学院理学系研究科，東京都
Graduate School of Science, The University of Tokyo, 7–3–1 Hongo, Bunkyo-ku, Tokyo, 113–0033 Japan
[††] 東京大学大学院新領域創成科学研究科，東京都
Graduate School of Frontier Sciences, The University of Tokyo, 7–3–1 Hongo, Bunkyo-ku, Tokyo, 113–0033 Japan

レンダリングを意識せずにすむ．この3次元のグラフィックスレンダリングは，jGLが行う．もし，開発したプログラムをウェブ上で公開する場合，このプログラムとjGLを一緒にウェブサーバにおいておく．そして，ユーザはウェブブラウザのインラインアプレットでこのプログラムを使用することができる．また，ユーザはjGLを用いてインターネット上で3次元グラフィックスのアプリケーションを使用するとき，jGLあるいは他のパッケージをインストールする必要がない．必要なすべてのコードが必要に応じてサーバからダウンロードされる．

筆者らは1997年にjGLの初版[3]を公開した．当時，マシンのパフォーマンスが良くなかったので基本的なレンダリングルーチンだけをサポートした．また，初版のjGLのパフォーマンスは，C言語を用いて開発されたMesa-3D[4]と比べて，4分の1程度遅かった．そのため，テクスチャマッピングのような複雑な計算が必要な機能を省略した．現在，ハードウェアは高速化され，インターネット上での3次元グラフィックスの表示を要求することも多くなっている．そのため，jGLの機能向上を計った．しかし，ハードウェアは昔より改善されたが，インターネット利用者の急増により1人当りのネットワークバンド幅は減少したのと等価であるので，jGLの機能を拡張するとき，コードサイズを最小にする必要もある．そのため，jGLのカーネルを記述し直した．

最近，携帯電話（例えば、NTT DoCoMoの503i以降のiAPPLI[5]など）でもJavaのプログラムが動作する．jGLの開発環境はJava Standard Edition（J2SE）であるが，iAPPLIのJava環境はJava Micro Edition（J2ME）に基づいたNTT DoCoMoの特殊なバーチャルマシンであるので，直接実行させることができない．したがって，iAPPLIで動作可能とするために，一部のjGLをiAPPLIへ移植した．

インターネット上で便利なプログラミング環境を構築するためには，3次元グラフィックスライブラリのみが不足している．3次元グラフィックスアプリケーションを開発するときは，3次元モデルを表示する必要がある．インターネット上で標準的な3次元モデルのファイル形式はVRMLである．そのため，jGLのAPIを参考にして，VRMLの仕様書[6]に従い，同じくJavaのみを用いて，jVLと呼ばれるVRMLライブラリを開発した．また，jVLの3次元グラフィックスレンダリングも同様にjGLが行う．

2. 関連システム

インターネット上で，Web3Dプラットホームのためのいろいろな関連システムがある．すなわち，3次元のグラフィックスライブラリ（下記のJava3DとJSparrow）や，3次元物体を表示するためのプログラム（下記のShout3D，blaxxun3D，Cortona JetとXj3D）などである．

Java3D – Java3D[7]はSun Microsystems社のJavaの3次元サポートシステムである．OpenGLあるいはDirectXを用いているので，パフォーマンスは当然よいけれども，プラットホームに依存する問題が起こる．また，Java3DのAPIは独自に定義された形式なので，APIを学習する時間がかかる．更に，Java3Dを用いて開発したプログラムを実行する前に，Java3Dをインストールする必要がある．

JSparrow – JSparrow[8]はPFU社のOpenGLのJavaバインディングソフトウェアである．ローカルマシンのOpenGLを用いているので，Java3Dのようなプラットホームに依存する問題も起こる．また，JSparrowを用いて開発したプログラムを実行する前に，JSparrowをインストールする必要がある．

Shout3D – Shout3D[9]はEyematic Interactive社の商品であり，Javaのみを用いてインターネット上で3次元モデルを表示できる．サポートしている3次元モデルのファイル形式は独自に定義された形式であるが，VRMLとの変換プログラム（1対1の対応ではない）がある．VRMLのように，Shout3Dは3次元モデルのアニメーションをプログラムするために独自に定義されたAPIを提供する．

blaxxun3D – blaxxun3D[10]はBlaxxun Interactive社の商品である．blaxxun3DはShout3Dのように同じくJavaのみを用いてインターネット上で3次元モデルを表示できる．サポートしている3次元モデルのファイル形式はVRMLとX3D（Extensible 3D；次世代のVRMLファイル形式）の草案である．JavaEAI（Java External Authoring Interface）の概念に従って，blaxxun3Dも3次元モデルをプログラムするためのAPIを提供する．

Cortona Jet – Cortona Jet[11]はParallel Graphics社の商品である．Shout3DとblaxxunD3Dのように，Cortona JetもJavaのみを用いてインターネット上で3次元モデルを表示できる．サポートしている3次元モデルのファイル形式はVRMLであるが，

表 1　関連システムの比較
Table 1　The comparison of related systems.

システム	Java のみ	API の普及度	VRML 支援
Java3D	×	△	×
JSparrow	×	○	×
Shout3D	○	×	△
blaxxun3D	○	×	○
Cortona Jet	○	×	○
Xj3D	×	×	○
提案システム	○	○	○

プログラムするための API は提供しない.

Xj3D – Xj3D [12] は Web3D Consortium のオープンソースのシステムである．Xj3D は Java を用いて開発しているが，3 次元グラフィックスレンダリングは Java3D を用いている．サポートしている 3 次元モデルのファイル形式は VRML と X3D の草案である．なお，Xj3D は X3D ファイル形式の正式なテスト用のプラットホームである.

表 1 のように Java3D はプラットホームに依存する問題があり，そしてプログラマは更に多く学習する時間がかかるので，よく知られた API のプラットホームに依存しない 3 次元グラフィックスライブラリを提供することが有用だと思われる．更に，インターネット上でいろいろな 3 次元モデルを表示するプログラムがあるが，ユーザはモデルの動作，光源の変更などをプログラミングすることができない．そのため，インターネット上で 3 次元グラフィックスのプログラムを開発するとき，jGL と jVL を利用すれば，以前より容易になる.

3. jGL の概要

jGL は Java のための 3 次元グラフィックスライブラリであり，前もってインストールしておく必要がない (すべての必要なコードはランタイムにダウンロードされる)．更に，多くの 3 次元グラフィックスのプログラマは OpenGL に慣れているので，jGL を使いやすくするために，OpenGL の仕様書 [13] に従って，jGL の API を OpenGL の API と同様に定義した．そうすることで，プログラマが jGL を用いてインターネット上で 3 次元のグラフィックスプログラムを開発するとき，OpenGL の関数に 1 対 1 に対応する jGL の関数があるので，他のライブラリを学習する必要がない.

3.1 OpenGL vs. jGL

OpenGL の関数は OpenGL ユーティリティライブラリ（GLU）と OpenGL（GL）の二つの種類に分類

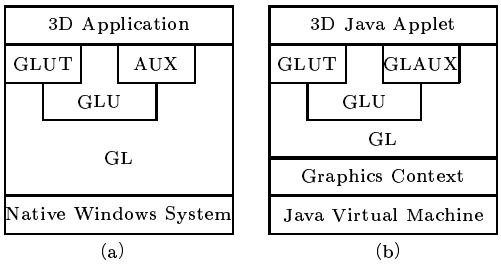


図 1　(a) OpenGL と (b) jGL の階層構造
Fig. 1　The hierarchy of (a)OpenGL and (b)jGL modules.

することができる．その階層構造は図 1 (a) のようになっている．この階層構造を参考にして図 1 (b) のような階層構造をもつように jGL を開発した.

GL は描画するのに必要なプリミティブな 3 次元グラフィックスオペレーションの，座標変換，クリッピングなどを含めて有用である関数をサポートする．GLU は GL の関数を用いてより容易な関数をプログラマに提供する．この二つの主なインタフェースのほかに，GLAUX（あるいは AUX）という OpenGL プログラミングガイド補助関数群がある．GLAUX は OpenGL に本来含まれる関数ではない．しかし，GLAUX は広く使用され，親しまれているため，これも開発した．最近，GLUT を利用するプログラマが増えているので，この部分も開発した.

jGL の GL，GLU と GLUT の実装は主に OpenGL，OpenGL ユーティリティライブラリ [14] と GLUT の仕様書 [15] に基づいている．また，GLAUX ライブラリは OpenGL プログラミングガイド [16] に従って実装される．GL，GLU，GLUT と GLAUX はプログラマのためのインタフェースだけである．それらのもとに，jGL のカーネルとしてグラフィックスコンテクストがある.

3.2 グラフィックスコンテクストの実装

コードサイズを減らす，また jGL のパフォーマンスを保持するあるいは高めるために，Java のクラス継承という特徴を利用して図 2 のようにグラフィックスコンテクストのシステム階層を設計した.

jGL のグラフィックスコンテクストは二つの部分に分類することができる．一つはディスプレイリストのための部分である．すべての GL からの命令は実行せずにレンダリング命令のシーケンスとして保存される．もう一つは実際の描画を行うアルゴリズムの部分である．すべての GL からの命令は直ちに実行される.

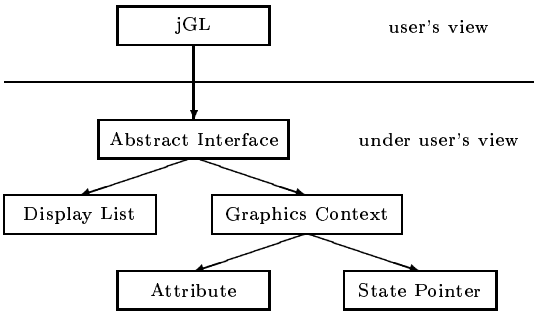


図 2 jGL のグラフィックスコンテクスト
Fig. 2 The graphics context of jGL.

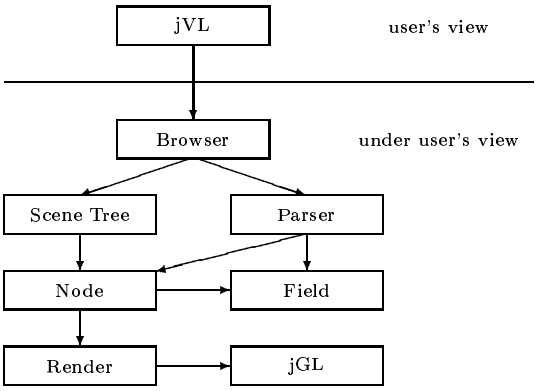


図 3 jVL のシステムの階層構造
Fig. 3 The system hierarchy of jVL.

GL からの命令も二つの種類に分類することができる．一つはグラフィックスコンテクストのステートを変更する命令である．もう一つはグラフィックスコンテクストのステートに基づいた実際のグラフィックス命令である．パフォーマンスを高めるために，描画するアルゴリズムはいろいろな参考文献[17]〜[20]を参照した．

3.3 パフォーマンスを高める課題

計算時間に関するパフォーマンスは3次元グラフィックスとJavaアプリケーションの重要な課題である．更に，jGL はインターネットを介して操作するように設計されているため，インターネットのバンド幅も重要な問題と考えなければならない．そのため，jGL の大きな課題は計算時間に関するパフォーマンスだけではなく，jGL の機能を増加すること，及び，コードサイズを最小にすることも含まれている．

そこで，以下に述べるような四つの点を考慮した．①「 if-then-else 」の構文を避けるために，クラス継承を用いること，②ルーチンの細分を試みること，③分割したルーチンをグループ化すること，④コードサイズを最小にするために，ファンクションオーバライディングの手法を使用することである．

3.4 iAPPLI への移植実験

jGL の開発環境と，携帯電話で動作する iAPPLI というJavaの特殊なバーチャルマシンは違うので，jGL を iAPPLI へ移植するために，iAPPLI の制限を守らなければならない．iAPPLI の主な制限は以下の三つである．①浮動小数点数をサポートしていない．②アプリケーションサイズの制限のため，JAR ファイルのサイズは 10KB 以下である．③描画に必要な関数が iAPPLI 自身のクラスしかない[21]．そのため，iAPPLI 向けの jGL はすべての計算を整数演算に書き直した．サイズを抑えるために，不要な定数とエラーチェックの機能を除去した．そして，iAPPLI 向けの jGL では 30 以上の OpenGL の関数を実装した．それには，3次元の座標変換，3次元の投影，隠面消去，基本的なジオメトリの描画が含まれている．また，ライブラリサイズは 4194 バイトである．

4. jVL の概要

jVL は Java のための VRML ライブラリである．インターネット上での新たなプログラミング環境を提案するために，3次元グラフィックスライブラリ以外に，3次元モデルを表示するライブラリも必要である．そして，jGL の能力をテストするために，VRML の仕様書に基づいて，jGL を用いて，jVL を開発した．jVL も Java のみによって書かれているので，プラットホームに依存せず，前もってインストールする必要もない．このほか，jVL を使いやすくするために，筆者らは jGL の API を参考にして jVL の API を定義した．

4.1 システムの階層構造

VRML のデータ構造は主としてノードとフィールドという二つの種類に分類することができる．3次元モデルは木構造としていくつかのノードと結び付けられる．また，ノードのパラメータはいくつかのフィールドに保存される．このデータ構造に基づいて図 3 のようなシステムの階層構造をもつように jVL を開発した．

ノードとフィールドは3次元モデルを構築するための主なデータ構造である．jVL はプログラマのためのインタフェースだけである．すべてのアルゴリズムは

ブラウザというクラスにおく．３次元モデルのファイルを読むとき，パーサーが構文解析を行って，シーントリーという木構造を生成する．描画する場合，jGL によって，ノードとフィールドで保存されたデータを用いて，画像を生成する．

4.2 パフォーマンスを高める課題

jGL の開発の経験より，以下に述べるような二つの点を考慮した．① jGL と同じくクラス継承とファンクションオーバライディングの手法を用いること，② ディスプレイリストを利用すること．

5. 実験と応用例

jGL では実用上十分な 300 以上の OpenGL の関数を実装した．GL，GLU，GLUT と GLAUX を含んでいる．それらには，２次元及び３次元の座標変換，３次元の投影，デプスバッファ，スムーズシェーディング，ライティング，材質特性の設定，ディスプレイリスト，セレクション，テクスチャマッピング，ミップマッピング，エバリュエータと NURBS が含まれている（アンチエイリアシングはサポートしていない）．

jGL の機能のテスト用に，29 個の例を jGL のウェブページ(注1)で公開している．これらの例は，OpenGL の公式ガイドである OpenGL プログラミングガイドから選んだものである．jGL は Java のみを用いるので，これらの例はそのまますべてのマシンで実行できる．筆者らはいろいろな種類のオペレーティングシステム上でこれらの例をテストした．例えば，Microsoft 社の Windows 98，NT，2000，ME，Sun Microsystems 社の Solaris（Sparc と x86），Silicon Graphics 社の Iris と Linux である．

図 4 は白い正方形を表示する簡単な例の jGL のソースコードである．この例は OpenGL の公式ガイドである OpenGL プログラミングガイドの簡単な例（Listing 1-2, pp. 13, Figure 1-1）と似ている．

jGL のパフォーマンスのテスト用に，材質特性の異なる 24 個のティーポットをレンダリングした．図 5 のような，個々のティーポットは jGL のベジェ曲面の機能（OpenGL のエバリュエータ）を用いて自動的に生成された 12544 個ポリゴンである．このテストプログラムも同じく OpenGL プログラミングガイドから選んだ例（Plate 17）である．測定した結果を表 2 に示す．

(注1)：http://nis-lab.is.s.u-tokyo.ac.jp/∼robin/jGL

```
import jgl.GL;
import jgl.GLApplet;

public class simple extends GLApplet {
    public void display () {
        myGL.glClear (GL.GL_COLOR_BUFFER_BIT);
        myGL.glColor3f (1.0f, 1.0f, 1.0f);
        myGL.glBegin (GL.GL_POLYGON);
            myGL.glVertex3f (0.25f, 0.25f, 0.0f);
            myGL.glVertex3f (0.75f, 0.25f, 0.0f);
            myGL.glVertex3f (0.75f, 0.75f, 0.0f);
            myGL.glVertex3f (0.25f, 0.75f, 0.0f);
        myGL.glEnd ();
        myGL.glFlush ();
    }

    public void init () {
        myUT.glutInitWindowSize (500, 500);
        myUT.glutInitWindowPosition (0, 0);
        myUT.glutCreateWindow (this);
        myGL.glClearColor (0.0f, 0.0f, 0.0f, 0.0f);
        myGL.glMatrixMode (GL.GL_PROJECTION);
        myGL.glLoadIdentity ();
        myGL.glOrtho (0.0f, 1.0f, 0.0f,
                      1.0f, 0.0f, 1.0f);
        myUT.glutDisplayFunc ("display");
        myUT.glutMainLoop ();
    }
}
```

図 4　白い正方形を表示する簡単な例の jGL のソースコード

Fig. 4　The source code of a simple example of jGL to show a white rectangle.

表 2　図 5 のパフォーマンスの測定

Table 2　The performance testing of Fig. 5.

Java 環境	実行時間	プラットホーム
Sun JDK 1.3.1	1347 ms	Intel PentiumIII-1GHz, 512MB memory, NVIDIA GeForce2 GTS Microsoft Windows ME
	9166 ms	Sun Ultra-10 360MHz, 256MB memory, Sun Solaris 7

iAPPLI 向けの jGL を実験するために，図 6 のような動くロボットのアームを描くプログラムを記述した．このテストプログラムは同じく OpenGL プログラミングガイドから選んだ例（Listing 3-7, pp. 114 – 115, Figure 3-22）である．また，携帯電話のボタンを利用するとロボットのアームも本の例と同じく動く．

Java3D と比較するために，図 7（a）のような Java3D のパッケージに HelloUniverse という例として面の色の異なるキューブを描くプログラムを書いた．二つのプログラムは同じマシンでパフォーマンスを測定した．結果は両方ともリアルタイムに表示された．

図 5　材質特性の異なる 24 個のティーポットの jGL による描画結果.
Fig. 5　24 teapots with different materials are rendered with jGL.

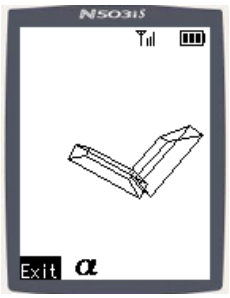
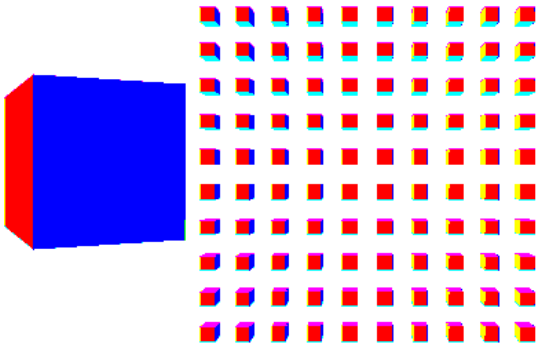


図 6　携帯電話でのロボットのアームの描画.
Fig. 6　A robot arm is rendered on a mobile phone.



(a)　　(b)

図 7　jGL, jVL と Java3D のパフォーマンスの測定に使用したプログラム. (a) 面の色の異なるキューブを描画する簡単な例. (b) (a) のようなキューブを 100 個描画する例
Fig. 7　The program used to measure the performances of jGL, jVL and Java3D. (a) A simple example to render a simple cube with different colors. (b) An example to repeat the same cube as (a) 100 times.

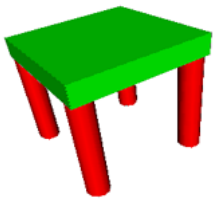


図 8　jVL によるテーブルの描画.
Fig. 8　A table is rendered with jVL.

表 3　図 8 のパフォーマンスの測定
Table 3　The performance testing of Fig. 8.

Java 環境	実行時間	プラットホーム
Sun JDK 1.3.1	15 ms	表 2 の PC
	23 ms	表 2 のワークステーション

そのため，小さいモデルやシーンの場合には，jGL のパフォーマンスは Java3D とほぼ同じであると考えられる.

簡単なモデルでは，パフォーマンスを比較できないので，図 5 のような例を使って，パフォーマンスを測定した．表 2 の PC を用いた場合，Java3D による実行時間は 181 ms である。

開発した jVL では 70% 以上の VRML のノードを実装した．また，ルートとイベント送信メカニズムも含んでいる．jVL のパフォーマンスを評価するにあたり，図 8 のようなテーブルの 3 次元モデルを使用した．このモデルは VRML 仕様書から選んだ例（p. 212, Figure D.3）である．測定した結果を表 3 に示す.

簡単なモデルでは，jVL のパフォーマンスを測定できないので，図 7（b）のような数百個以上の同じキューブを描画する例を用いて，パフォーマンスを測定した．結果を表 4 に示す.

図 9 は jVL を用いて VRML ファイルを表示できるソースコードである．また，VRML ファイルに含まれる物体，光源，センサなどの制御も可能である.

Shout3D または blaxxun3D と比較するために，図 10（b）のような同じ建物の 3 次元モデルを 100 個描画する例を使って，パフォーマンスを測定した.

表 4　jVL のデータ量によるパフォーマンスの測定
Table 4　The performance testing for data size of jVL.

キューブ数	実行時間	プラットホーム
100	29 ms	表 2 の PC
400	64 ms	
900	125 ms	
1600	198 ms	
2500	291 ms	
3600	393 ms	
4900	509 ms	
5600	649 ms	
8100	825 ms	

```
import jvl.VL;
import jgl.GLApplet;

public class viewer extends GLApplet {
    VL myVL = new VL (myGL);

    public void display () {
        myVL.vlRenderWorld ();
    }

    public void init () {
        myUT.glutInitWindowSize (500, 500);
        myUT.glutInitWindowPosition (0, 0);
        myUT.glutCreateWindow (this);
        myVL.vlSetWorldURL (getDocumentBase(),
                            "exampleD.4.wrz");
        myVL.vlInit ();
        myVL.vlViewpoint (getSize ().width,
                          getSize ().height);
        myUT.glutDisplayFunc ("display");
        myUT.glutMainLoop ();
    }
}
```

図 9　jVL を用いて VRML ファイルを表示するソースコード
Fig. 9　The source code for displaying VRML file by using jVL.

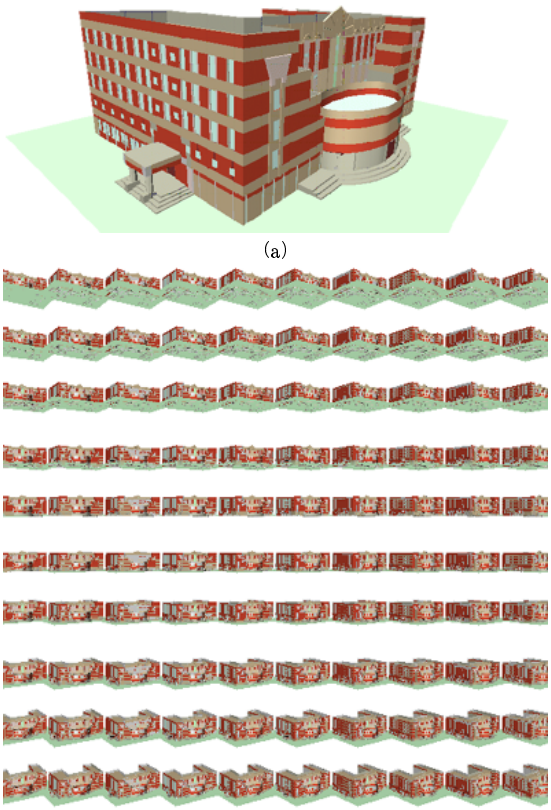
表 5　jVL, Shout3D と blaxxun3D のパフォーマンスの比較
Table 5　The performance comparison of jVL, Shout3D and blaxxun3D.

ライブラリ	実行時間	プラットホーム
Shout3D 1.03	2.25 sec	表 2 の PC
blaxxun3D 1.18	1.31 sec	
jVL 2.0	4.31 sec	

個々のモデルは図 10（a）のような 5273 個のポリゴンと 12 種類の材質特性がある．結果を表 5 に示す．

6. む す び

本研究では Java のみによる jGL と jVL を実装した．また，携帯電話向けの iAPPLI 版の jGL も実験



(a)

(b)

図 10　パフォーマンスを測定するためのモデル．(a) 建物の 3 次元モデルを描画する例．(b)（a）のような建物のモデルを 100 個描画する例
Fig. 10　The model used to compare the performances. (a) The rendering of a 3D building model. (b) An example to repeat the same model as (a) 100 times.

した．筆者らが行った比較によると，OpenGL あるいは DirectX をそのグラフィックスエンジンとして用いる Java3D に対して，jGL のパフォーマンスは単純なモデルに対してはさほど悪くない．したがって，プログラムを実行する前に，ユーザ側が jGL とこのプログラムをインストールする必要がないので，jGL は Web3D プラットホーム上の簡単なモデルを利用するプログラムに非常に適していると思われる．筆者らがウェブサーバ [22] に jGL をアップロードしたとき以来，インターネットの jGL の多くの利用者がいることからも有用性が示された．また，50 人の利用者にアンケート調査をした結果を表 6 に示す．大部分の経験者は「jGL は信頼性がよい」と「jGL の使い方は容易である」と評価したので，このアンケートにより jGL

表 6　jGL のアンケートの調査結果
Table 6　The questionnaire result of jGL.

	最良	良	悪	最悪
信頼性	5	45	0	0
使いやすさ	8	42	0	0

の信頼性と有用性が実証された.

jVL のパフォーマンスは現在まだ十分ではないが，プログラマに対し公開している API があるので，3 次元モデルの動作，光源の変更などをプログラミングすることができる．また，jGL と同様な利点があるので，Web3D プラットホーム上のプログラムが jVL を利用することができるので，プログラムの開発はより容易になったといえる．jGL と jVL の主な欠点はパフォーマンスの問題であるが，それはより速いインタプリタ，コンパイラと CPU の開発によって解決されることが期待できる．また，このシステムは Java のライブラリを提供するだけではなく，PDA，携帯電話などのような，Java ベースのあらゆるブラウザやシステムに組み込むのに役立つ.

文　献

[1] "The Source for Java(TM) Technology," Sun Microsystems, Inc., http://java.sun.com, 2001.

[2] "OpenGL – High Performance 2D/3D Graphics," OpenGL, Org., http://www.opengl.org, 2001.

[3] B.-Y. Chen, T.-J. Yang, and M. Ouhyoung, "JavaGL – A 3D graphics library in Java for Internet browsers," IEEE Trans. Consum. Electron., vol. 43, no. 3, pp. 271 – 278, 1997.

[4] B. Paul, "Mesa 3-D graphics library," http://www.mesa3d.org, 2001.

[5] "All about i-mode," NTT DoCoMo, Inc., http://www.nttdocomo.co.jp/i/java.html, 2001.

[6] R. Carey, G. Bell, and C. Marrin, "ISO/IEC 14772-1: 1997 Virtual Reality Modeling Language (VRML97)," VRML Consortium, Inc., 1997.

[7] "Java 3D(TM) API Home Page," Sun Microsystems, Inc., http://java.sun.com/products/java-media/3D, 2001.

[8] "JSParrow – A Java Binding for OpenGL," PFU, Inc., http://www.pfu.co.jp/jsparrow, 2001.

[9] "Welcome to Shout3D," Eyematic Interfaces, Inc., http://www.shout3d.com, 2001.

[10] "blaxxun3D," blaxxun Interactive, Inc., http://www.blaxxun.com/products/blaxxun3d, 2001.

[11] "Cortona Jet," Parallel Graphics, Inc., http://www.parallelgraphics.com/products/jet, 2001.

[12] "Xj3D Open Source Browser," Web3D Consortium, http://www.web3d.org/TaskGroups/source/xj3d.html, 2001.

[13] M. Segal, and K. Akeley, "The OpenGL Graphics Systems: A Specification (Version 1.2.1)," Silicon Graphics, Inc., 1999.

[14] N. Chin, C. Frazier, P. Ho, Z. Liu, and K. P. Smith, "The OpenGL Graphics Systems Utility Library (Version 1.3)," Silicon Graphics, Inc., 1998.

[15] M. J. Kilgard, "The OpenGL Utility Toolkit (GLUT) Programming Interface API Version 3," Silicon Graphics, Inc., 1996.

[16] J. Neider, T. Davis, and M. Woo, OpenGL Programming Guide,Addison-Wesley, 1993.

[17] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, Computer Graphics Principles and Practice, Addison-Wesley, 1990.

[18] A. S. Glassner, Graphics Gems,Academic Press, Inc., 1990.

[19] J. Arvo, Graphics Gems II, Academic Press, Inc., 1991.

[20] D. Kirk, Graphics Gems III, Academic Press, Inc., 1992.

[21] "i モード対応 Java コンテンツ開発ガイド," NTT DoCoMo, Inc., 2001.

[22] R. B.-Y. Chen, "jGL 3D Graphics Library for Java," http://nis-lab.is.s.u-tokyo.ac.jp/∼robin/jGL, 2001.

（平成 x 年 xx 月 xx 日受付）

陳　炳宇

1995 台湾大学資訊工学科卒．1997 同大大学院資訊工学専攻了．同年，台湾陸軍総部入部．1999 台湾大学資訊工学科助手．2000 東大大学院理学系研究科博士課程入学．工学修士．主として，インターネットグラフィックスに関する研究に従事．情報処理学会，ACM，IEEE 各学生会員.

西田　友是　（正員）

1971 広島大・工卒．1973 同大大学院工学研究科了．同年，マツダ（株）入社．1979 福山大学工学部講師．1984 同助教授．1990 同教授．1998 東京大学理学部教授．1999 同大大学院新領域創成科学研究科教授．工博．主として，コンピュータグラフィックスに関する研究に従事．情報処理学会，ACM，IEEE 各会員.