2007－CG－126 (6)
2007／2／19

インタラクティブ3Dコンテンツ作成用スクリプト言語とそのプレーヤの開発

上村学[1] 吉岡信夫[2] 大村皓一[3]
[1]大阪工業大学大学院 工学研究科 [2]大阪工業大学 工学部 電子情報通信工学科
[3]宝塚造形芸術大学 メディア・コンテンツ学部 映像造形学科

3D のプレゼンテーションやインタフェースの開発を行うには高度な 3D プログラミングの知識が必要である．本研究ではこれらの 3D コンテンツ開発を手軽に行う手法として，オブジェクト指向のスクリプト言語を利用した 3D コンテンツ開発システム(ICCS)を提案する．このシステムでは 3D オブジェクトモデルにスクリプトで動作やインタラクションを付加することで開発を行う．オブジェクト指向の特性を生かし，ビルトインクラスによって 3D 特有の複雑な計算を考慮することなく，最低限必要な動作を記述するだけで開発を行うことができる．また，コンテンツプレーヤの描画システムに DirectX を利用することで高機能な 3D 表現が利用可能である．

Development of Script Language and its Player for Interactive 3D-Contents

Manabu Kamimura[1], Nobuo Yoshioka[2], Koichi Omura[3]
[1]Graduate School of Engineering, Osaka Institute of Technology
[2]Department of Electronics, Information and Communication Engineering, Osaka Institute of Technology
[3]Department of Media Arts, Image Design, Takarazuka University of Art and Design

To develop an interactive 3D presentation, it is necessary to use detailed knowledge of advanced 3D programming. To make it easy, we propose the Interactive 3D Contents Creation System (ICCS), which uses an object-oriented script language. In this system, a contents creator describes operations and interactions by adding the script to the 3D models. Object-oriented nature of the script eases the development, which only needs minimum description without complex calculations peculiar to 3D. The drawing part of the system, the player, uses DirectX, which enables high performance 3D presentation.

1. はじめに

近年，コンピュータの性能の向上により，PC でも 3DCG を利用したコンテンツを扱うことが容易になっている．また，研究発表や会議などでも 3D アニメーションを利用することによって，データや研究結果などをより分かり易く表示することができ，効率的なプレゼンテーションを行うことが可能である．一方，3D コンテンツを作成するには高度なプログラミング技術の習得が必要であり，その製作コストが問題となっている．

そこで，本研究はプレゼンテーションやインタフェースに利用可能な 3D コンテンツを，3D に関する専門的な知識を必要とせず，手軽に開発する方法を検討する．その具体的な方法として，オブジェクト指向のスクリプト言語を利用してプレゼンテーションなどに利用可能な 3D コンテンツを開発するシステム(Interactive 3D Contents Creation System: ICCS)を提案する．

2. 関連研究

ICCS は外部で作成した 3D オブジェクトモデルに，スクリプトによる記述で動きやインタラクションを付加し，コンテンツの開発を行う．このようなスクリプトを利用したコンテンツ開発[1][2][3]では Flash[4][5]が有名であるが，Flash は 2D コンテンツの作成に主眼が置かれており，3D への拡張はプログラマが奥行きなどの計算を別途記述することが必要であり，高度なプログラミング能力が不可欠である．また，CAD やゲーム開発に用いられる OpenGL や DirectX などの 3DCG 用 API を用いる

方法が考えられる．しかし，これらの方法も高度なプログラミング技術と初期化処理や様々なパラメータの設定などが必要であり，できるだけ手軽に済ませたいプレゼンテーションの作成手法には適していない．本システムでは 3D 表現に必要な計算や設定は全てシステム側で行い，コンテンツの製作者は最低限必要な動きを記述するだけで開発を行うことが可能にすることを目標とした．

3. コンテンツ作成システム

3.1. システムの概要

ICCS の概略図を Fig.1 に示す．ICCS はスクリプトの解析を行い，実行用の仮想マシンコード(VM-Code)を出力する ICCS Compiler と，VM-Code を解析し，実行する ICCS Player で構成される．

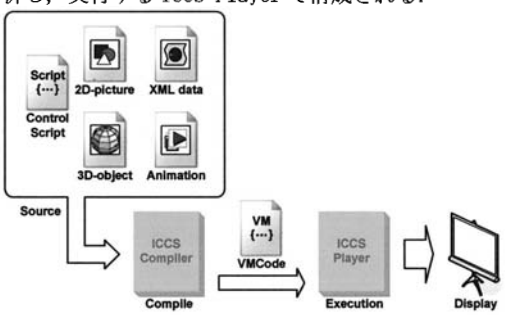

Fig.1 Outline of ICCS

コンテンツの開発者はプレゼンテーションに必要な描画オブジェクトを用意し，それらに対する動作などをスクリプトに記述する．それらを ICCS Compiler に入力することでスクリプトの解析[6][7]を行い，コンテンツ開発を行う．

ICCS Player は描画システムに DirectX[8]を採用しており，描画における複雑な計算はシステム側が自動的に DirectX を通して処理する．これによりコンテンツ開発者は描画時のシェーディングなどの複雑な計算を意識することなく開発を行うことが可能である．

読み込まれる 3D オブジェクトのファイル形式は X ファイル形式であり，他のモデリングソフト(例えば Metasequoia[9])などで作成したものをスクリプト上で読み込ませることで利用可能である．読み込まれた 3D オブジェクトは位置やスケーリングなどの情報を与えることで画面に表示される．

ICCS は 3D コンテンツの開発を主眼においているが，必要に応じて 2D 画像を読み込むことが可能である．これにより 2D 画像を背景に設定したり，スクリプト上から動的にテクスチャとして 3D オブジェクトに貼り付けたりすることも可能である．

文字の出力においても 2D，3D の両方を出力することができる．表示したいテキストをスクリプト上に記述することで，2D，3D のテキストを動的に作成することが可能である．光源についてはスクリプトに光の色や強さを記述することで表現が可能である．また，必要に応じて動画や XML データも読み込むことが可能である．

これらの機能の例を Fig.2 に示す．Fig.2 では 2D テキスト(world)，3D テキスト(hello)，2D 画像(バラ)，そして 3D オブジェクト(球)を表示している．光源は中心付近に配置しており，3D テキスト及びオブジェクトに対してはシェーディングが行われていることがわかる．3D オブジェクトには動画をテクスチャとして貼り付けている．

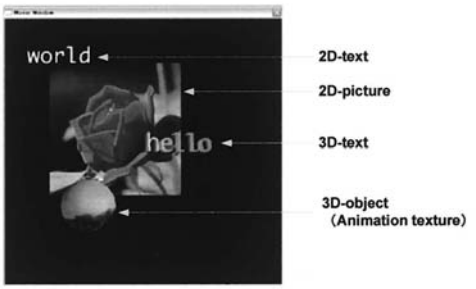

Fig.2 3D-Contents example

3.2. ICCS Script

ICCS は描画オブジェクトの制御にスクリプトを利用している．スクリプトにそのオブジェクトの位置や大きさ，回転角などを記述することでオブジェクトの配置やアクションを設定する．スクリプトの採用により，描画すべきオブジェクトとそのアクションを分離することができ，コンテンツの開発をよりスムーズに行うことが可能である．また，移動などのアクションをベクトルなどのデータとして保持するのではなく，スクリプトにおいて演算として記述することができるため，作成されたコンテンツのデータサイズは非常に小さなものとすることができる．

スクリプトには Java[10]や Ruby[11]を手本とした，オリジナルのオブジェクト指向言語を採用している．オブジェクト指向言語の採用により，2D 画像や 3D オブジェクトを 1 つの(クラスとしての)オブジェクトとして捉えることが可能である．これにより，描画オブジェクトの属性(位置や大きさ，回転角など)をプロパティとして取り扱うことができる．また，描画オブジェクトが行うべきアクションはそのオブジェクトのメソッドとして取り扱うことができる．つまり，描画オブジェクトに対しそのメンバとして特性を記述し，インスタンスを生成することで画面に描画オブジェクトを表示する．

オブジェクト指向スクリプトのもう 1 つの大きなメリットとして継承がある．継承を利用することにより，コンテンツ開発者はシステム側で用意

されているクラス(ビルトインクラス)や既に別のプロジェクトで開発したクラスなどを利用することができる．プレゼンテーションやインタフェースを開発する場合，同じような処理を行うことが多くなるため，これらのプログラムの再利用は有用である．

特にビルトインクラスの利用は，プログラミング経験の少ないコンテンツ開発者にとって手軽に開発を行うことを可能にする非常に有用な手段である．描画オブジェクトは 2D 画像，3D オブジェクトなどに分けられるが，スクリプトではそれぞれビルトインクラスとしてシステム側に実装されている．コンテンツ開発者は作成するオブジェクトに，これらのビルトインクラスの継承を行う．これにより，そのオブジェクトの固有の機能のみを書き加えることで開発を行うことができる．

例えば List.1 のようなクラスを考える．List.1 では MovieClip3D というクラスをスーパークラスに持つ MoveMC というクラスを宣言している．

```
class MoveMC extends MovieClip3D
{
public:
    function Move(Void): Void{…};
    function SetVelocity (x: Number, y: Number): Void{…};
private:
    var vx: Number;
    var vy: Number;
}
```

List.1 Example of inheritance

List.1 の記述により Fig.3 のようなクラスが構築される．MovieClip3D とは 3D オブジェクトとしての基本的な機能を提供するビルトインクラスである．MovieClip3D にはファイルから 3D オブジェクトを読み込んだり(AttachMovie)，テクチャを貼り付けたり(SetTexture)する機能や，位置(position)や回転角(rotation)などの情報を持つ．クラスの作成者は MovieClip3D を継承することで 3D オブジェクトとしての基本的な機能を簡単に MoveMC に組み込むことができる．そして，MoveMC としての固有の機能である位置の移動(Move)や移動距離の設定(SetVelocity)を行うメソッドや，それらで利用する変数(vx, vy)のみを記述するのみで，クラスの構築を行うことが可能である．

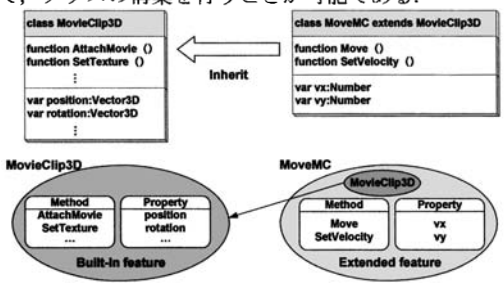

Fig.3 Inheritance of MovieClip3D

3.3. インタラクション

本研究はプレゼンテーションやインタフェースを作成するツールの開発を目的としているため，作成されたコンテンツにはユーザからの入力に対するインタラクションが必要である．ICCS にはマウス入力とキーボード入力に対するインタラクションを実装している．ユーザがマウスやキーボードから入力を行うとそれに対応するイベントメソッドがシステム側から呼ばれる．コンテンツの開発者はこのイベントメソッドに処理を記述することでインタラクションを実現する．

イベントメソッドの例を List.2 に示す．List.2 ではマウスによってオブジェクトがクリックされた場合に，クリックされたオブジェクトに対応するメソッドが呼ばれる"onPress"メッセージを Shpere クラスに実装している．

```
class Shpere extends MovieClip3D
{
public:
    onPress=function (Void): Void
    {
        //  Processing of click interaction
    };
}
```

List.2 Event message definition

このイベントが呼ばれるタイミングを Fig.4 に示す．Fig.4 では球のオブジェクトがインスタンスとして画面上に表示されている．この場合，表示されたオブジェクトをクリックされると，このメソッドが呼ばれる．複数のオブジェクトが重なって表示されている場合にクリックを行えば，最も手前にあるオブジェクトのインスタンスのイベントメソッドが呼び出される．

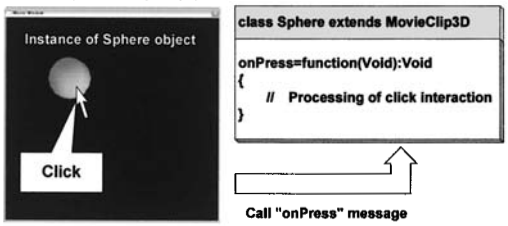

Fig.4 Interaction message

これらのメッセージはビルトインクラスに実装されており，継承によって利用することが可能である．インタラクションの実装をこのようにイベントメソッドとして扱うことで，他のメソッドを作成するのと同じ感覚で扱うことができる．

3.4. XML

ICCS はデータを XML[12]ファイルから読み込みことができる．これによりプログラムとデータを分割することができ，データの変更だけならば XML データファイルを変更するだけで行うことができ

る．また，データの管理を行いやすくなり，プログラムの再利用も簡単に行うことができる．

例えば，プレゼンテーションを開発する際に，〈タイトル〉や〈本文〉，〈画像ファイル名〉などのタグで分けられた XML ファイルからそれぞれデータを取得して表示するプログラムを作成した場合，XML ファイルを変更するだけで，新しいプレゼンテーションを作成することができるようになる．

XML ファイルは XML オブジェクト(XML)に読み込むことで利用可能である．XML は Fig.5 のように木構造でデータを管理しておりタグをたどることでデータを取得する．XML は XML ファイルを読み込むとそれらをハッシュによって整理する．これにより，目的のデータへ高速にアクセスすることが可能になる．

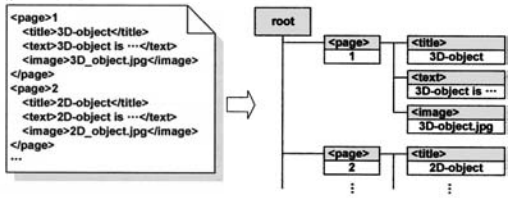

Fig.5 XML data structure

3.5. ICCS Player

ICCS Player の基本的な構成を Fig.6 に示す．ICCS Player は仮想マシン(Virtual Machine)，インスタンス管理部(Instance Manager)，メッセージ管理部(Message Manager)，描画処理部(Draw Processor)から構成される．

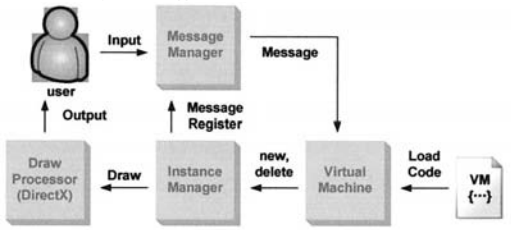

Fig.6 Composition of ICCS Player

(1) 仮想マシン

仮想マシンは一般的なスタックマシンである．仮想マシンコードを 1 つ取り出してコードを解釈，実行する．スクリプトはオブジェクト指向で書かれているので，this ポインタを保持し，this ポインタを利用して実行中のインスタンスを識別する．現在，実行時の型チェックやエラーの検出機能は存在せず，致命的なエラーが存在すればプログラムは停止する．

仮想マシンは仮想マシンコードによってインスタンスの生成(new)や解放(delete)を指示される場合がある．こうした場合，インスタンス管理部に生成するクラスを指示する．

仮想マシンはメッセージ管理部からイベントメッセージの指示を受ける場合がある．こうした場合，対応するインスタンスの対応するメッセージメソッドの実行を開始する．

(2) インスタンス管理部

インスタンス管理部はインスタンスの生成や解放を行う部分である．仮想マシンからインスタンスの生成や解放の指示を受けると，その内容からインスタンスを生成する．生成されたインスタンスが画面に表示するような描画オブジェクトであった場合は描画処理部に描画を指示する．また，インスタンスにイベントメッセージが割り当てられていればメッセージ管理部にメッセージメソッドの登録を指示する．同様にインスタンスの開放が行われれば，描画処理部には描画停止の指示を，メッセージ管理部にはメッセージ登録の解除の指示を行う．

(3) メッセージ管理部

メッセージ管理部はインスタンスごとのユーザからの入力やフレームを管理し，それらに対応して，仮想マシンにインスタンス情報とともにイベントメッセージを通知する．インスタンス管理部からメッセージメソッドの登録の指示を受け，インスタンスの情報と通知するイベントメッセージを登録する．また，インスタンス管理部からメッセージメソッドの登録解除の指示を受ければイベントメッセージの登録を解除する．

(4) 描画処理部

描画処理部は DirectX を利用して描画オブジェクトの描画を行う．インスタンス管理部から描画開始の指示を受け，対応するインスタンスの描画を行う．また，インスタンス管理部から描画停止の指示を受ければ対応するインスタンスの描画を停止する．

描画はフレーム毎に行われる．新しいフレームに入ると MovieClip2D でバックグラウンドに表示するべきインスタンスを Z 値が大きい順に描画を行う．次に MovieClip3D のインスタンスの描画を行う．最後に MovieClip2D で前面に表示するべきインスタンスを Z 値が大きい順に行う．

4. コンテンツ開発の流れ

ここでは「太陽の周りを惑星が回転し，その惑星をクリックすると近づいて説明が表示される」という簡単な 3D コンテンツの例を作成しながら，ICCS によるコンテンツ作成の具体的な手法と利用可能なビルトインメソッドを紹介する．

4.1. 3D オブジェクトの準備

コンテンツ開発の最初のステップはコンテンツに必要な描画オブジェクトを用意することである．今回の例の場合，必要になる描画オブジェクトは，Fig.7 に示すような，太陽の 3D オブジェクトと惑星の 3D オブジェクト，そして惑星の表面にテクスチャするための 2D 画像である．ここで，太陽と惑星のオブジェクトを別のオブジェクトに分けているのは，太陽の 3D オブジェクトには自己照明を強めに設定するためである．光源は太陽の内側に設定するため，自己照明を強く設定することで，太陽が発光して周りの惑星を照らしているかのような印象を与える．

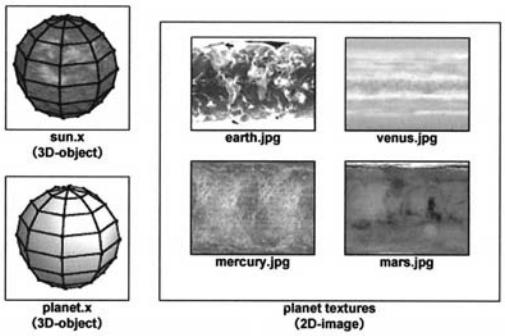

Fig.7 Drawing object used for sample

4.2. クラスの作成

惑星の動きや制御を ICCS Script で記述する．惑星クラス(Planet)は 3D オブジェクトである．したがって 3D オブジェクトの基本クラス"MovieClip3D"を継承すれば，3D オブジェクトの基本的な動作を実装することが可能である．

さらに Planet クラス固有の機能を持たせる．Planet には「3D オブジェクトを読み込み画面に表示する機能」，「フレーム毎に移動する機能」，「クリックすると説明を表示する機能」が必要である．

「3D オブジェクトを読み込み画面に表示する機能」は Planet クラスのコンストラクタで List.3 のように AttachMovie を呼び出せばよい．AttachMovie は MovieClip3D のメソッドであり，3D オブジェクトのファイル名を指定すればそのオブジェクトを読み込み画面に表示する機能を持つ．

```
class Planet extends MovieClip3D
{
public:
   function Planet()
   {
       AttachMovie ("planet.x");
   }
   ...
}
```

List.3 AttachMovie method

「フレーム毎に移動する機能」は List.4 のようにイベントメソッドである onEnterFrame 内に惑星の公転の移動を記述すればよい．惑星の位置は position というプロパティで制御されているのでそれを変化させることで，移動を行う．

```
onEnterFrame=function(Void):Void
{
    position.x = Sin(pos)*radius;
    position.y = Cos(pos)*radius;
...
}
```

List.4 Planet class's onEnterFrame method

「クリックすると説明を表示する機能」はイベントメソッド onPress 内にその内容を記述する．

4.3. XML データの作成

惑星のデータを XML ファイルとして記述する．惑星のデータは Fig.8 のような構造とし，XML ファイルに記述する．このようにデータを XML ファイルとしてプログラムとは別に保存しておくことで，容易に変更が行える．また，スクリプトの作成者とは別の人間がデータを変更する必要がある場合も，XML データを変更するだけであるので，変更する人物がプログラミング経験のない人物でも容易に行うことができる．

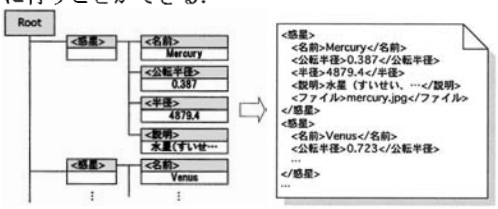

Fig.8 XML data of planets

4.4. Root の作成

ICCS Player は Root クラスのインスタンスを 1 つ作成することで，プログラムを開始する．つまり，Root クラスのコンストラクタが開始メソッドである．したがって，光源の配置や XML から惑星データを取得するなどの初期化処理は Root のコンストラクタに記述する．

XML データのロードは XML クラスを利用する．LoadXML メソッドでデータをファイルからロードし，用意されている探索用メソッドで XML のデータを取得する．

一部のイベントメソッドは Root にも設定することができる．惑星の公転や自転などは各惑星クラスで行うのが適当であるが，カメラが惑星にフォーカスするなどの全体の処理は Root に記述するべきである．したがって，List.5 のようなフレーム毎のイベントメソッドを設定する．List.5 のように全体のカメラ視点の移動は SetCamera でカメラ位置と注視点を設定することで行う．

```
onEnterFrame=function(Void):Void
{
    SetCamera (camera, lookat)
...
}
```

List.5 Root's onEnterFrame method

4.5. 実行

作成した ICCS Script は ICCS Compiler でエラーがないかどうかを確認する．エラーがなければ Fig.9 のようにプログラムが実行される．プログラムの実行時のウィンドウはコンテンツを表示する MovieWindow と Trace 文などのデバッグ用テキストを表示する DebugWindow から構成される．プログラムが実行されると Fig.9(a)のように太陽系の図(惑星は簡単のため水星，金星，地球のみ)が表示され，惑星をクリックすれば Fig.9(b)のように，クリックされた惑星にフォーカスして説明と惑星の名前を表示する．

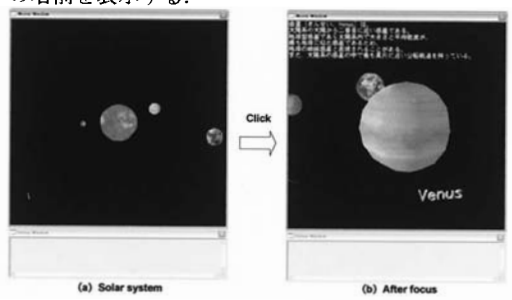

Fig.9 Execution of "Solar System"

このコンテンツのスクリプトの行数はコメント文を含めて 200 行程度である．もし，この行数でこれと同じプログラムを C++で DirectX を利用して作成しようとすれば，ウィンドウを作成し，DirectX を初期化することさえ困難である．また，スクリプト内には 3D 表現に必要なシェーディング処理はもちろん，デバイスの作成や初期化の処理は含まれていない．オブジェクトの作成や移動といったアクションの処理のみを記述するだけでコンテンツの作成を行うことができる．

5. まとめ

本研究ではプログラミング経験の浅いユーザでも簡単なスクリプトの記述を行うだけで 3D のプレゼンテーションやインタフェースを作成するツール，ICCS の開発を行った．ICCS の開発により，研究結果の発表や社内プレゼンテーションにおいても，今まで敬遠されがちであった 3D の要素を取り入れることができる．これにより時間や技術をあまり必要とせずに効果的なプレゼンテーションを作成することが期待できる．

今後の課題としてはビルトインクラスの拡張，言語仕様の調整，デバック機能の拡張，開発支援機能の拡張などが考えられる．

参考文献

[1] 間瀬健二，Sidney Fels，江谷為之，"仮想環境ラピッドプロトタイピングに適した VR スクリプト言語の開発" 情報処理学会研究報告，Vol.98 No32(CG-90) p1-p6，情報処理学会，1998.

[2] 義山琢也，岡田義広，新島耕一，"スクリプトによる 3D CG アニメーション生成のためのコンポーネントベースのプレイヤーの実装"，情報処理学会，第 67 回全国大会論文集，p4-235-p4-236，2005.

[3] 新藤義昭，松田洋，鈴木誠史，"3D-CG Animation のシナリオ記述言語 CPSL と Cyber Teaching Assistant の開発"，情報処理学会論文誌，Vol.43，No.8，p2782-p2795，2002

[4] Jeff Tapper，James Talbot，Robin Haffner，"Object-Oriented Programming with ActionScript2.0"，New Riders，2004.

[5] 大塚勝三，"Flash ActionScript Handbook"，ソフトバンクパブリッシング，2004.

[6] 水野順，"スクリプト言語を作ろう" C Magazine 2000 年 5 月号 p8-p37，ソフトバンクパブリッシング，2000.

[7] 石畑清，疋田輝雄，"コンパイラの理論と実現"，共立出版，1988.

[8] 星正明，"DirectX9 実践プログラミング"，工学社，2003.

[9] http://www.metaseq.net/ (metasequoia について)

[10] James Gosling，Bill Joy，Guy Steele，Gilad Bracha，"Java 言語仕様 第 2 版"，ピアソン・エデュケーション，2000.

[11] http://www.ruby-lang.org/ja/(ruby について)

[12] William R. Stanek，"XML 標準リファレンス"，日経 BP ソフトプレス，2002.