ハイパフォーマンス
コンピューティング 77－7
（1999. 8. 2）

WindowsNT上でのクロス開発環境を目指す並列化支援ツールの開発

水原隆道◇, 平尾智也◇, 齊藤哲哉◇, 笹倉 万里子†, 城和貴‡, 國枝 義敏◇

◇ 和歌山大学システム工学部
† 岡山大学工学部
‡ 奈良女子大学理学部

現在の自動並列化コンパイラでは，逐次プログラムを効率良く完全に自動並列化することは困難である．より効率の良い並列化を行うためには，利用者がどの部分をどのように並列化するかを，明示的に指示する必要がある．並列化支援可視化システム NaraView では，並列化の度合いを含むプログラム構造，データの依存関係などの情報を視覚化して利用者に提示することが可能である．我々は，NaraView を WindowsNT 上に移植し，従来の機能に加えてソースプログラムとの連係を図る．さらに，ユーザーインターフェイスの改善や，自動並列化コンパイラ Parafrase-2 との連係を強化したシステムを開発し，評価を行う．

Development of a Parallelizing Support System for Cross-platform Programming Environments on a WindowsNT

Takamichi Mizuhara◇, Tomoya Hirao◇, Tetsuya Saito◇,
Mariko Sasakura†, Kazuki Joe‡, Yoshitoshi Kunieda◇

◇ Faculty of Systems Engineering, Wakayama University
† Faculty of Engineering, Okayama University
‡ Faculty of Sciences, Nara Women's University

It is difficult for parallelizing compilers to transform sequential programs to efficiently parallelized programs automatically. To generate efficient parallel programs, programmers should indicate the part of a sequential program which they want to parallelize. NaraView, which is a visualization system for supporting program parallelization, can visualize the overall structure of parallel programs and information of data dependence. We ported the NaraView on a WindowsNT. We keep the former functions of NaraView to associate them with source programs on the new version. Moreover, we improved the user interface, and developed a module which can closely cooperate with a parallelizing compiler, Parafrase-2.

1 はじめに

自動並列化コンパイラの分野では，完全に自動で効率の良い並列化を行うことは難しい．そのため現在の自動並列化コンパイラの多くは，逐次プログラムを並列化する際に，コンパイラが発見できない並列性を，ユーザーがコンパイル時に付加情報としてソースプログラムと共に与えてやることが多い．しかし，ユーザーがプログラム中の複雑なフロー情報やデータの依存関係をソースプログラムから読みとり，新たな並列性を発見することは極めて困難である．そこで，我々は，並列化の対象となる逐次のソースプログラムと，ソースプログラムを詳細に解析して得られた情報を基に視

覚化した画像をユーザーに提示することで，ユーザーが並列化の効果を確認し，その結果を基に再び並列化を行えることができるようなツールとしてNaraViewを開発した[3, 4]．こうすることにより，特に並列プログラミングにおける専門的な知識を多く必要とせず，より性能の高い並列プログラムを開発することが可能となる．このNaraViewを元に，多種多様なアーキテクチャを対象としたクロス開発環境を提供するため，NaraView/NTを開発した．以下，2節ではNaraViewの概要とNaraViewで用いている2種の視覚化法について述べ，3節ではWindowsNTへの移植とNaraView/NTにおけるインターラクティブな操作環境について説明し，4節では，NaraView/NTと他の既存システムとの連係について述べ，最後に5節で全体の評価と結論を述べる．

2 NaraViewの概要

2.1 背景

自動並列化コンパイラは逐次プログラムを詳細に解析し，意味を変えることなく並列化されたプログラムに書き換えるコンパイラである．自動並列化コンパイラを用いることで，過去のプログラム資産を活用し，また，並列化の知識を必要とせず並列システムを利用したプログラム開発が行える．しかし，現在の自動並列化コンパイラでは，完全に自動で効率の良い並列プログラムを生成することは難しい．その原因となるのが，逐次プログラム上のどの部分をどのような並列化手法を使って並列化すれば効率がよいのかという判断や，並列化の鍵となるデータの依存関係の正確な解析が難しさなどが挙げられる．例えば，配列の添字として実行時に動的に決定される要素が含まれている場合や条件分岐が含まれ，意味の解釈が必要となる場合など，コンパイラには簡単に予測できない条件が多い．これらの問題があるため，本当は並列化できる部分に対しても，コンパイラは安全のため並列化を行わない場合がある．

このような問題を解決する支援システムとしてNaraViewは開発された．NaraViewは，UNIX系のOS上で開発された並列化支援視覚化システムであり，ソースプログラムを入力として，並列化のために必要な情報であるプログラムの制御フロー，制御依存，データ依存を後述する視覚化法で表示する．ユーザーが空間上のノードをクリックすることで対応するソースコードの文が表示され，それらの対応を見ることでループの並列化が可能であるかどうかを判断し，どのような並列化を行えばよいかを知ることができる．この視覚化を何らかの形で行う必要があり，我々はソースプログラムレベルでの抽象的な情報を視覚化するProgram Structure View[3]とプログラム中のデータ依存関係を視覚化するData Dependence View[4]を提案し開発した．

2.2 プログラム情報の視覚化

プログラム情報の視覚化では，Fortranのソースプログラムを入力として，プログラムの各ステートメントを，機能別に色付けしたノードとしての物体で表示する．各ノードはそれぞれが制御フローに基づいており，それらの集合でプログラム全体が構成される．1つのノードはソースプログラムの1ステートメントに相当するものである．プログラムの全体の構成・形状を直感的にとらえ，並列化の可能性を読みとれるようにするため，次のように3軸を対応させている．

x： プログラムフロー

y： 並列度

z： ループ構造階層

プログラムフローは，ノード間の実行順序を示す．しかし，NaraViewではノードの実行順序を1つの値で示しているためすべてのフローは表示せず，クリティカルパスのみを表示するようにしている．フローの分岐には，if文のように実行時はどちらか一方しか実行されない条件分岐と，do文のようにループを形成するものとがある．条件分岐については，ステートメント数の多い方を取り上げるが，ソースレベルで扱っているため正確なコスト算出は行っていない．また，ループは階層構造で表現している．並列度は，通常同時に実行可能な命令数で表示される．ループ構造の階層表示は，1つの階層が1つのループに対応するように表示するため並列化されているか否かがループごとに見やすくなる．浅い階層は深い階層よりも外側のループを表す．また，ループノードを実際に構成するノードとの対応関係を明確にするためノード間をアークで結んでいる．

この表示法では，yの値が並列度を表していることから，並列化の観点からはy軸に広がったノードの集合である方がより望ましいといえる．また，逆にy軸への広がりが1のものは逐次プログラムであることを示す．

ガウスの消去法のプログラムを並列化し，この方法により視覚化すると図2.2のようになる．部分的にノードがy軸に広がっており，これが並列化された部分である．

2.3 データ依存関係の視覚化

プログラム中で同じデータを参照している2つのステートメントS1，S2がありS1の方が先に実行されるとする．この2つのステートメントの間には，そのデータに関してデータ依存関係があるという．このとき，S1を依存の始点，S2を依存の終点

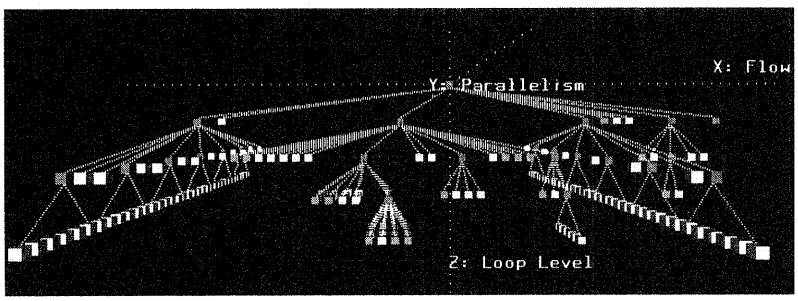

図 1: 並列化ガウスの消去法のプログラム構造

と呼ぶことにする．データ依存関係は，データへの参照が読み込みであるか，書き込みであるかによって４種類に分類できる．このうち，両方とも読み込みである場合には，並列化を妨げる依存関係は発生しないので，以下の３つの場合について考えることにする．

フロー依存：S1-書き込み　S2-読み込み

逆依存　　：S1-読み込み　S2-書き込み

出力依存　：S1-書き込み　S2-書き込み

多重ループを含む一般のループの並列可能性は，ループネスト全体を考えるよりは，各ループごとについて考える方が簡単である，着目したループのすべての参照データに関して，フロー依存，逆依存，出力依存がいずれも無いとき，そのループは並列化可能である．また，フロー依存が存在する場合，ループのイタレーションを変更することによってフロー依存の距離（S1 と S2 の間の依存関係を持たないステートメント数）を変えることが可能である．このような操作を行うことによって，並列化可能な部分が増える場合がある．さらに，逆依存，出力依存が存在する時，変数の複製を作ることによって依存関係は解消でき，並列化可能となる場合もある．また，複製を作らない方法では，逆依存の始点終点の間に同期点を挿入することで，その同期点の間では並列化可能である．よって，同期点をいれるべき部分と，ループのイタレーションとの関係から，実際の並列化可能性が決定する．

そこで，我々はフロー依存，逆依存，出力依存とループのイタレーションとの関係に着目してデータ依存関係の視覚化を行った．

データ依存関係の視覚化を行う Data Dependence View では，プログラム中のデータアクセスを空間上にキューブによって表示している．x-y 平面上に AVD(Array-Variable Disposition) マップというプログラム内で宣言されている配列や変数を配置する．x-y 平面上の１座標に配列の１要素または，１変数が割り当てられることになる．z 軸は時間軸として概念時刻を対応させる．Data Dependence View では，概念時刻としてイタレーションアクセスタイム，ステートメントアクセスタイムを切り替えて表示することができる．アクセスの種類はキューブの色で区別する．読み込みアクセスは青，書き込みアクセスは赤，読み書き同時に発生する場合は紫としている．

データ依存関係をユーザーに明示するために，アクセスを示すキューブ間をポールでつなぐ．ポールは，ライフタイム（フロー依存）ポール，逆依存ポール，出力依存ポールの３種類あり，それぞれ緑，黄，ピンクで表される．また，それぞれの変数が書き込まれた後最初に読み込まれるのがどの時刻かという情報を得るため，最小フロー依存と呼ばれるポールを表示する機能も備えている．

ループにおける依存関係を表示する際には，ループによって生じる繰り返しが，表示されているどの部分に相当するのかを明確に示す必要がある．そこで，アクセスとループ構造の関係を明確にするため，指定した概念時刻の位置に x-y 平面に平行な半透明平面を表示している．これをループグリッドと呼ぶ．

このような視覚化情報から最適化，並列化，デバッグなどに有用な情報を次のように読みとることができる．

- フロー制御ポールから変数の生存期間がわかり，意図しない読み込み，書き込みの存在が発見できる．
- 出力依存ポールが存在した場合，始点の書き込みデータは使われていないことがわかる．
- すべてのポールを表示した状態である $z = i$ の x-y 平面に平行な面を横切らなかったら，その前後に依存関係がないことを意味する．
- AVD マップは変数や配列のメモリ上の配置と考えることでデータの分散の問題に用いることができる．

2.2 節で示した例と同じガウスの消去法のプログラムを並列化し，データ依存関係を視覚化すると図２のようになる．この図中のポールは，フロー制御ポールである．

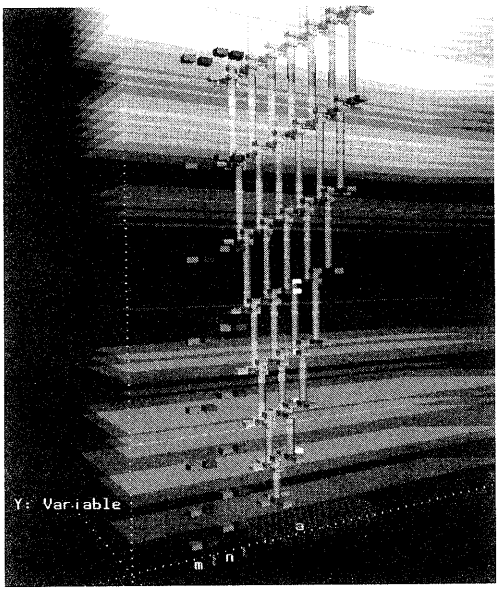

図 2: 並列化ガウスの消去法のデータ依存関係

3 WindowsNT への移植と機能強化

3.1 WindowsNT への移植

当初 NaraView は，UNIX 上で OpenGL を用いて開発されたシステムであった．近年，並列プログラムの実行環境などのダウンサイジングにより，多種多様なシステム上で並列プログラムが動作可能となった．また，低価格コンピュータの普及により携帯型 PC などでもプログラム開発が行われるようになった．これらの背景から，この要求を実現すべく PC 用 OS の代表格ともいえる Windows 環境への NaraView の移植・拡張を行った．また，対象とするシステムも並列コンピュータからワークステーションクラスタなどの分散環境というように多くの範囲に及ぶ．そのため，NaraView/NT には，これらのシステム間でクロス開発を可能とするためのインターフェイスが必要と考えた．

NaraView/NT では，開発済みコードの見直しを含め，コードの最適化，冗長性の排除を行った．また，インタラクティブな使用を目指す支援システムの特性上，視覚化システムの描画速度を向上させた．さらに，内部データ構造の大幅な変更などを行うことによって，複雑な内部データからの視覚化情報の取り出し性能を向上させた．

移植・機能拡張に際しては，Microsoft WindowsNT4.0 を対象として実装を行ったが，Windows98 や Windows95 などを搭載したノート PC でも動作可能となるように注意した．NaraView/NT は，NaraView の基本機能を受け継ぎ，より高機能，高性能なインターフェイスと，前述の Program

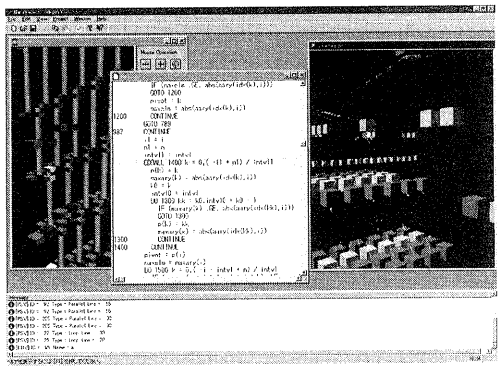

図 3: NaraView/NT の統合画面

Structure View と Data Dependence View の統合，さらにネットワークに接続された他のシステムの自動並列化コンパイラを使用するコンパイラとして指定可能な機能も実装し，任意のターゲットマシンを選択可能なクロス開発環境となっている．(図 3)

3.2 共通インターフェイスの構築

我々は，NaraView をクロス開発環境への移植の際に，実際の使用に耐えうるユーザーインターフェイスが必要であると考え，次のような入出力インターフェイスを装備した．

3.2.1 入力デバイス

入力デバイスとしては，キーボード，マウスでのフルオペレーションの機能以外に，空間の移動には，一般にフライトシミュレータなどで使用されているジョイスティックを利用可能とした．また，フィードバック対応のジョイスティックを使用することにより，情報の存在空間から大きくはずれた場合などジョイスティックを振動させ，空間上で迷うことがないように調整している．キーボード，マウス，ジョイスティックでの空間移動は，視覚化対象オブジェクトを回転させるのではなく，利用者自身が乗り物にでも乗っているかのように空間内を自由に移動できるのが特徴である．つまり，入り組んだノード間の情報を自分が見たい位置，角度から見ることが可能である．

3.2.2 視覚化情報の出力と利用

NaraView/NT の視覚化情報の利用における利便性を向上させるため次のような出力形式を利用可能としている．

- OpenGL によるインタラクティブな出力．

- 視覚化データのスナップショットを画像ファイル（ビットマップ形式）に出力

- 空間データを VRML(Virtual Reality Modeling Language)2.0 仕様 [8] のデータとしてエクスポート可能.

OpenGL による情報の表示では，入力デバイスの項で示したように，キーボード，マウス，ジョイスティックによる操作が可能であり，ノードやキューブの情報をマウスクリックやジョイスティックのボタンを押すことによって取り出すことができる．これにより，ノードの色の意味や情報などを覚える必要が少なく，利用者の負担は小さくなる．画像ファイルとして保存した場合は，OpenGL の操作環境で表示されているスナップショットがそのまま保存されるため，特定の部分だけを取り出して後で利用したり，問題がある場所だけを別の開発者に提示したりといった利用方法が考えられる．さらに，情報の VRML へのエクスポートにより，NaraView/NT が動作しない環境や，ネットワークを利用したリモート環境での利用が Web ブラウザなどとの併用により可能である．

4 他のシステムとの連係と支援環境

本節では，NaraView/NT を使用した統合的な開発環境を構築するための他のシステムとの連係支援環境を提案する．

本稿は，クロス開発環境を目的とするため，ターゲットのアーキテクチャには依存しないユーザーインターフェイスを提供しつつ，並列化の指示などはそのアーキテクチャに依存した形で行わせる必要がある．そのため，NaraView/NT と連係させるためのアーキテクチャ依存のコンパイラを考えない．NaraView/NT は，自動並列化コンパイラとして，Parafrase-2[2] のインターフェイスに対応している．そのため，あるアーキテクチャマシン上で Parafrase-2 が動作している必要がある．

さらに，NaraView/NT と Parafrase-2 とのそれぞれの環境で互いに通信を確立する必要がある．今回は，TCP/IP プロトコルを利用して，LAN やインターネット上でクロス開発が可能なシステムを目的とした．実際にターゲットマシンとして使用したのは，OS に FreeBSD を採用した PC で，その上で更に Parafrase-2 を利用可能にしている．お互いの通信には，TCP/IP のソケットを利用し，FreeBSD 側に NaraView/NT からの通信に応じて Parafrase-2 を起動する簡単なサーバープログラムを搭載した．具体的なシステムのイメージは図 4 を参照されたい．

まずはじめに，逐次プログラムのソースファイルを NaraView/NT 上のテキストエディタで編集可能状態にして，コンパイルコマンドを実行する．すると，NaraView/NT は設定済みの Parafrase-2 が待機中のサーバーに逐次ソースファイルを送信する．ソースファイルを受信したサーバープログラム側では Parafrase-2 を起動しコンパイルを行い，その結果である並列プログラムソースと解析情報（階層タスクグラフ）[1] を NaraView/NT に返信する．NaraView/NT では受信した並列プログラムソースと階層タスクグラフを利用して，視覚化を行い利用者に提示する．利用者は，視覚化された情報を読みとり，ソースファイルと並列化指定を付加したパスファイルを作成し，再びコンパイル指定を行う．これを繰り返し，最終的により効率の良い（並列処理部分の多い）プログラムの生成を行うことができる．最後に，この並列化ソースプログラムをターゲットのマシン用の並列コンパイラに対してコンパイルを行うことによって最終的な並列プログラムの実行コードを生成する．

このように NaraView/NT の環境上では，ターゲットマシンアーキテクチャに対する並列化を，視覚的かつ対話的に行える．このことにより，従来の逐次プログラムを比較的容易に並列化することが可能であり，さらに，新規の並列プログラムの開発においても有効な支援システムとして動作すると思われる．

現在の NaraView/NT は，Parafrase-2 の出力に依存しているため，Parafrase-2 以外の自動並列化コンパイラは使用できないが，将来的には，NaraView に汎用的な自動並列化コンパイラの機能を取り込み，よりインタラクティブ性の高い並列化統合開発環境を目指している．

5 結論

本稿では，WindowsNT 上で，多くのプラットホームに対してのクロス開発環境を支援する並列化支援ツール開発に関する報告を行った．また，並列化支援を目的としてプログラム情報をソースコードレベルで視覚化を行うシステムとして 2 つの手法を説明した．

これらの視覚化情報を用いることにより，どのループの並列化を行えばよいかといった並列プログラミングにおける効率の良いプログラミング環境を提供でき，結果としてより効率の良い並列プログラムの開発が可能となる．また，各変数，配列がどのタイミングでアクセスされるのか，また，そのアクセスは読み込み，書き込みのいずれなのかといった情報などから，それぞれの変数，配列の要素において依存関係の把握が可能となり，ループ内のデータアクセスの状態を特定でき，並列化手法の選択や直感的な評価が可能となる．さらに，視覚化情報に実際のメモリ配置を写像することによって，プログラムを実行した時のメモリアクセスの状況を容易に推測することが可能となり，最適なメモリ配置の決定を支援することも可能である．

我々は，これらの機能を WindowsNT 上に移植し，さらなる機能拡張，最適化を行った NaraView/NT

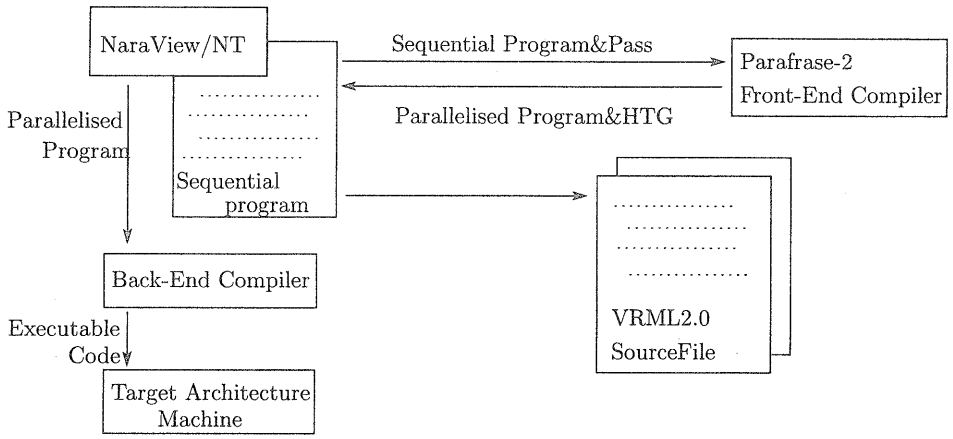

図 4: NaraView/NT と Parafrase-2 の連係

の開発を行った．Windows 上に移植することにより安価なシステムでの並列プログラミングの開発・支援が可能となり，実際に利用するシステムとしての位置づけで，ユーザーインターフェイスの大幅な改善を行った．主なものとしては，視覚化された 3 次元空間上の移動をジョイスティック，マウス，キーボードのいずれのデバイスによっても操作可能であり，ノート PC からワークステーションまで最適な操作方法を選択可能としている．また，OpenGL での描画情報を，WWW(World Wide Web) などでよく知られた，VRML2.0 の仕様に基づいたソースコードや，ビットマップ形式の画像の出力も可能とした．

NaraView/NT では，現在，Source to Source の自動並列化コンパイラ Parafrase-2 との連係が可能であり，ネットワーク上の Parafrase-2 が動作可能なシステムと通信を行える環境があれば，NaraView/NT は，並列プログラミングの統合開発支援環境として利用可能である．

NaraView，NaraView/NT の視覚化情報はターゲットのマシンアーキテクチャに依存しないため，逐次プログラムを多くの並列アーキテクチャシステム上で最適化を行うことが比較的容易に行える．しかし，本稿で述べた視覚化システムだけでは，並列化に対するすべての情報が視覚化できているわけではない．例えば，DO ループの実行回数などがソースプログラム上で変数で与えられている場合，その実行回数を視覚化することは難しい．本システムは，これらの解決法やコンパイラとの情報連係など多くの問題を残しているのでそれを今後の課題としたい．

参考文献

[1] Girkar, M.B., Polychronopoulos, C.D.: The Hierarchical Task Graph as a Universal Intermediate Representation, International Journal of Parallel Programming, Vol. 22, No. 5, pp. 519-551, 1994.

[2] Polychronopoulos, C.D. et al. : Parafrase-2:An environment for parallelizing, partitioning, synchronizing, and scheduling programs on multiprocessors, Proceedings of the 1989 International Conference on Parallel Processing, 1989.

[3] 笹倉万里子, 木和田智子, 城和貴, 荒木啓二郎: 並列化支援視覚化システム NaraView におけるプログラム情報の表示法, 並列処理シンポジウム JSPP'96, pp. 267-274, 1996.

[4] 笹倉万里子, 木和田智子, 城和貴, 荒木啓二郎: 並列化手法選択支援のためのループ内データ依存関係の視覚化法, 並列処理シンポジウム JSPP'97, pp. 297-304, 1996.

[5] 笹倉万里子, 城和貴, 國枝義敏, 荒木啓二郎: 変数オリエンティッドなデータ依存関係モデルの提案, 情報処理学会論文誌 数理モデル化と応用 Vol. 40, No. SIG2(TOM1), pp. 45-54, 1999.

[6] OpenGL Architecture Review Board, Manson Woo, Jackie Neider, Tom Davis: The official Guide to Learning OpenGL, Version 1.1, Addison-Wesley Publishers Japan Ltd., 1997.

[7] OpenGL Architecture Review Board: The official reference document for OpenGL, Release 1, Addison-Wesley Publishers Japan Ltd., 1995.

[8] Jed Hartman, Josie Wernecke: VRML2.0 handbook, Addison-Wesley Publishers Japan Ltd., 1997.