VRラピッドプロトタイピング言語 InvenTcl

Open InventorをTcl/Tkでラッピング

間瀬健二

1 はじめに

グラフィックスエンジンの発達により，汎用ワークステーションやパーソナルコンピュータで，3次元グラフィックスを表示したり，バーチャルリアリティシステムなどのインタラクティブなシステムを制作することが容易になった．さまざまな使いやすい3次元グラフィックスライブラリの提供により，さらにそれが加速されている．しかしながら，ちょっとした思い付きを試そうというときに，いちいちコンパイルしていたのでは面倒だ．ラピッドプロトタイピングには適さないのである．

まず，メニューなどのGUIの設計は，トライアル&エラーで試したいところであるが，そのために毎回コンパイルするのは面倒だし，ときにはメニュー自体は2次元の世界で十分なのに，3次元世界で平面を扱う必要があったりする．3次元グラフィックスやバーチャルリアリティをこれから勉強しようというユーザにとっては，プログラムしたことがどのように表示に影響するかがわかるようになるまで，多くの時間を費やさなければならないという問題を抱えている．3次元グラフィックスに初めてかかわるとき，例題どおりに関数をタイプしてみて，運よく絵が現れたとしても，使った関数の本当の意味はわからないままという経験があるのではないだろうか．

その点，Tcl/Tkのようなインタプリタ言語とWidgetツールキットの組合せは，GUIを使ったプログラミングに適している．なんといっても，インタプリタ型言語はとにかくすぐに結果がチェックできて，私のようにいい加減なプログラマ向きである．しかし，ご存じの通りTcl/Tkは，いまだ2次元の世界にとどまっていて，オブジェクト指向的3次元世界の表示はできない*1．

そこで，グラフィックスライブラリとして使いやすいOpen InventorをTcl/Tkでラッピングした，インタプリタ言語**InvenTcl**を開発することにした．以下，本稿では，ライブラリのラッピングの具体的方法とプログラミング例を示しながら，InvenTclを紹介する．

*1 3Dグラフィックス拡張のリストは http://www.tcltk.com/ にある．

2 InvenTcl

InvenTclは，C++ベースの3次元グラフィックスライブラリであるOpen Inventor（以下，Inventorと略す）を，Tcl/Tkの3次元Widget（これを3D-MegaWidgetと呼んでいる）として取り込もうというプロジェクトである．背景は前述のとおりであるが，もっと直接的なきっかけは別のプロジェクトで，InventorでVRの世界を作りつつ，スクリプトで制御できるモバイルで擬人的なCGエージェントをその世界に表示したいという要求があった．ところが，適当な言語が見あたらない．当時ATRに滞在していた，Sidney Fels（現University of British Columbia）が，InventorをTcl/Tkでラッピングしよう，ということを言いだした．1996年，InvenTclとよぶVRラピッドプロトタイピング言語の発端である．

Open InventorはC++ベースのオブジェクト指向グラフィックスライブラリなので，Tcl/Tkでそのままラッピングしても，Inventorのよさが現れない．そこで，[incr Tcl]というTcl/Tkのオブジェクト指向拡張モジュールを使う．つまり，Inventorの

〈リスト1〉Open Inventor によるシーングラフの定義とビューワの設定例[1]

```
// Hello Cone sample
SoSeparator *root = new SoSeparator;
root->ref();
SoMaterial *myMaterial = new SoMaterial;
myMaterial->diffuseColor.setValue(1.0, 0.0, 0.0);
root->addChild(myMaterial);
root->addChild(new SoCone);
SoXtExaminerViewer *myViewer =
            new SoXtExaminerViewer(myWindow);
myViewer->setSceneGraph(root);
```

〈リスト2〉InvenTcl によるシーングラフの定義とビューワへの割り付け例

```
# Hello Cone sample
set root [new SoSeparator]
$root ref
set myMaterial [new SoMaterial]
$myMaterial -diffuseColor setValue 1.0 0.0 0.0
$root addChild $myMaterial
$root addChild [new SoCone]
ExaminerViewer .v
.v setSceneGraph $root
```

C++ライブラリ[1]，Tcl/Tk[2]，Tcl のオブジェクト指向拡張である [incr Tcl] のライブラリ[3]を統合して，InvenTcl はできている．そうして，3D オブジェクトの生成，表示，アニメーション，インタラクションができる itkwish（[incr Tcl] のシェル）ウィンドウができあがる．

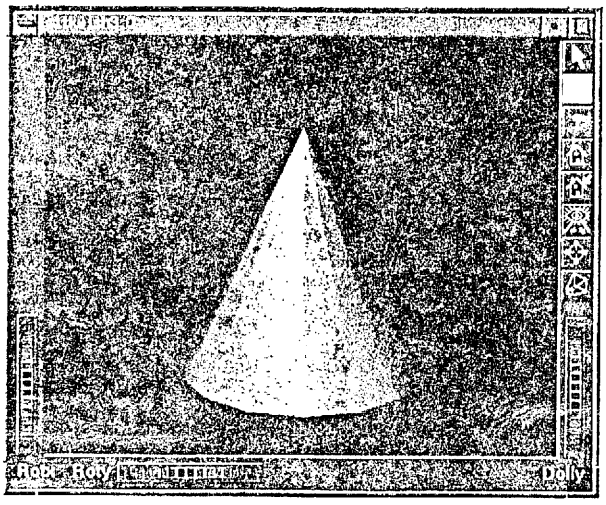
とにかく，InvenTcl のプログラム例を見てみよう．リスト1とリスト2に Open Inventor と InvenTcl のサンプルプログラムを示す．類似したシンタックスが実現できていることがわかるだろう．図1は InvenTcl による表示結果である．ウィンドウ名以外は Inventor の表示そのままである．

さて，InvenTcl は，Tcl に対しては3次元インタフェースを提供し，Inventor に対してはインタプリタインタフェースを提供していると考えることができる．当然，2つの言語体系（シェル言語と C++コンパイラ言語）は，本質的にシンタックスが異なっていて，InvenTcl はラッピングした Tcl/Tk/ [incr Tcl] 寄りであるので，Inventor プログラマから見ると多少連続性に欠けるだろう．また，イベントハンドリングなどは Inventor の流儀と Tcl/Tk の流儀が混在することになる．このへんの設計思想はプログラミングのしやすさに影響するだろうが，いまは両方にできるだけシームレスなシステムを提供することに徹している．例えば，Tcl/Tk には **bind** というバインディングのコマンドがあって，便利である．そこで，InvenTcl では新たに *Ibind* というコマンドを導入して，3D-MegaWidget 中のオブジェクトにバインディングできるように拡張している．これで，Tk を使った2次元のインタフェース widget を使いつつ，Tcl コマンドで3次元シーンへのバインディングによる直接アクセスが可能となる．*Ibind* については後述する．

InvenTcl は，VR システムのプロトタイピングをはじめ，3D グラフィックスの教育，3次元 GUI のプロトタイピング，ビジュアリゼーションなどいろいろな分野での利用が考えられる．C++と Open Inventor でプログラミングすることに比べて，InvenTcl には次のような利点があるだろう．

- シーン中のオブジェクトをスクリプトから操作したり，直接操作が可能
- 3D グラフィックスやアニメーションのプロトタイピングが容易
- 3D シーンと GUI との組合せが容易
- 3D グラフィックスを他のソフトウェアと結合することが容易

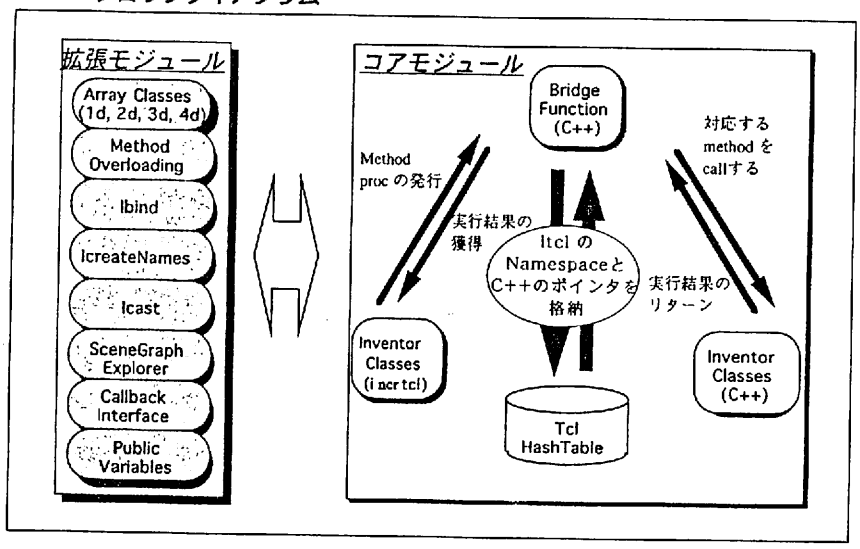
〈図1〉InvenTcl による Hello Cone の表示例



3 Open Inventorのラッピング

InvenTcl を生成するためには，まず Inventor のライブラリをラッピングする．それには，Inventor のクラスを [incr Tcl] のクラスに変換し，Tcl のシェルからすべての object, method, public field, static 関数にアクセスできるようにする必要があった．

〈図2〉InvenTcl における Open Inventor, Tcl/Tk, [incr Tcl] の関係を示すブロックダイアグラム



InvenTcl, Inventor, Tcl/Tk および [incr Tcl] の関係を図2に示す．ここで，ブリッジ関数が [incr Tcl] と Inventor のクラスをつなぐ役目をし，[incr Tcl] ではクラスのインスタンスは名前（ネームスペース）で保持し，Inventor ではポインタで保持する．両者を同期させるために，ハッシュテーブルを作成してこれらネームスペースとポインタを格納している．

なお，見かけ上 C++版と同じように使えることが望ましいので，object の階層構造と名前に一貫性を持たせるようにしてある．そのうえで Inventor のイベントチェックをするループを，Tcl/Tk のイベントチェックのループの中に入れている．また先に述べたように，Tcl/Tk のよさであるバインディング機能を生かすために，3次元 object にもバイディングできるようにするなど機能を拡張している．さらに，実行時の引数チェックや引き渡しをするために，Inventor の object 以外のパラメータも [incr Tcl] の引数として渡されるようにした．例えば，enumerate 型，array 型 (1D, 2D, 3D, 4D)，ファイルポインタ，関数ポインタなどである．

3.1 クラスの変換

ありがたいことに，コードの変換のほとんどは，Inventor のヘッダファイルの変換で済む．まず，Itcl++[4] とよばれるツールを用いて，Inventor のクラス構造を [incr Tcl] に変換する．こうして，インタプリタからオブジェクトのインスタンスを生成し，メソッドを呼び出すことができるようになる．クラスのインヘリタンスもそのまま維持されるので，クラス構造がそのままコピーできている．このItcl++という非常に便利なツールにより，クラスの変換は量的にはかなりの部分が自動的に行われた．しかしながら，ネイティブの Itcl++では自動処理できない問題がいくつかある．typedef 宣言されていない構造体，enumerate 型，クラス中の field 変数などがそうである．Inventor には存在しないはずの method が現れることもあった．その対応には Itcl++のパーザに手を加えた．

◆メソッドのオーバーローディング

まず，Inventor ではオブジェクト指向言語の特徴として，メソッドのオーバーローディングが多用されている．ところが，Itcl++の変換はオーバーローディングに対応していないので，複数種の引数をもつことのできるメソッドが別のメソッドとして定義されてしまうことになる．例えば，C++で定義されているメソッド *setValue* は，

```
setValue array
setValue r g b
```

のように使いたい．しかし，Itcl++でこれらを変換すると，

```
setValue1 array
setValue2 r g b
```

という具合いになってしまう．これを Inventor と同じく，*setValue* という1つのインタフェースで呼び出せるようにするため，Itcl++が生成した，ID 付きのメソッドに対して，もう1層殻をかぶせて，引数のチェックを行うようにした．

◆パブリック変数

次に，public 変数の生成の問題がある．Inventor には public 変数メンバを含むクラスがあるが，Itcl++はメンバを読むことができない．例えば，*SoMaterial* というノードには *diffuseColor* や *ambientColor* など複数の public フィールドがある．C++では *SoMaterial* が生成されると，*diffuseColor* フィールドなども自動的に生成されるので，ポインタでアクセスできるようになる．そこで，Tcl インタプリタからこのフィールドがアクセスできるようにするためには，*diffuseColor* のためのオブジェクトを生成し，[incr Tcl] の public 変数として関連づける．こうして，オブジェクト *SoMaterial* の public 変数はオブジェクト *diffuseColor* のインスタンスを含むことができ，リスト2に示すように，*setValue* コマンドでフィールドにアクセスできるようになった．

◆オブジェクトへのバインド

前述のとおり，Tcl の bind コマンドに対応する，3次元バインド *Ibind*

〈リスト3〉Ibind による Cone オブジェクトへのバインド例
(Cone をクリックするとメッセージがタイプされる)

```
Ibind $root $cone <Key-a> {puts "Key-a pressed!"}
```

を新規に作成した．シンタックスは下記の通りで，オブジェクト名，オブジェクトを含むシーングラフのトップノード名，ユーザイベントシーケンス，および Tcl スクリプトを引数とする．

Ibind *top-node object-node sequence script*

例えば，*cone* という名前のオブジェクトが *root* というシーングラフにあるとき，リスト3のように書ける．*sequence* の〈Key-a〉は，キーボードの〈Key-a〉キー押下イベントである．こうすると，Cone オブジェクト上で〈Key-a〉のキーを押すと，"Key-a pressed !" という文字が itkwish+のシェルウィンドウに出力される．引数の Tcl スクリプトは任意であるから，このバインドはユーザインタフェースのプロトタイピングに非常に有効である．

これを実現するため，シーングラフのトップに generic なコールバックノードを追加して，イベントが起きたときにシーングラフ中の定義されているすべてのバインドとマッチングをとるようにした．実装には32ビットのイベントマスクを用意し，マウスボタンの押下，移動，通常キー入力，Alt/Control/Shift などのモディファイアキーなどの入力に対応している．また，event 情報を渡すキーワードも Tcl の bind で使われるキーワードのサブセットとして，

%b，%x，%y，%A，%K，%N，%W

をサポートしている．

3.2 3D-MegaWidget の実装

Inventor Viewer の 3D-MegaWidget 化には少し手間がかかった．まず，InvenTcl Ver.1.0 では，Tk の描画ウィンドウと Inventor の描画ウィンドウを別々に表示して，基本機能を確認した．しかし，2つのウィンドウがあるということは，別々のウィンドウマネージャ間の通信の問題がある．

そこで，3D-MegaWidget 化のために，Ver. 2.x では，やむなく Tcl/Tk のコアに修正を加えてみることにした．このときは，canvas Widget にラッピングする方針で行った．Inventor の Xwindow を初期化して，Inventor のルートウィンドウを作成し，そのなかに Inventor ビューワのインスタンスと Tk のトップレベルウィンドウ情報を登録するようにした．そして，canvas Widget を拡張するように実装した．しかし，これは失敗だった．Tk の pack コマンドが希望通りの振る舞いをしてくれない．また，Inventor と Tcl/Tk のイベント管理が複雑になってしまったし，canvas では Inventor の機能を十分に表現しきれないことがわかった．何より，コアに手を入れたのは，今後の Tcl/Tk/ [incr Tcl] のバージョンアップに追従できない可能性を残すことになるからである*2．

最後に行きついたのは，現 Ver.3 で採用している，Tk のウィンドウに

*2 [incr Tcl] 2.2 がコア修正版だったことも，この方針をとった理由．3.0 からパッケージになったので，われわれも方針を変更した．

産業図書

プランと状況的行為
人間-機械コミュニケーションの可能性
L.A.サッチマン 佐伯 胖 監訳 2600円
人間の行為はすべて「状況に埋め込まれている」ということを，行動分析や「人間と機械とのコミュニケーション」のプロセスの詳細な分析から解明する。ヒューマン・インターフェース研究者の必読書。

デジタル知識社会の構図
電子出版・電子図書館・情報社会
合庭 惇 2000円
電子出版や電子図書館などのメディアの登場は，情報環境に大きな変化をもたらし，社会の高度情報化を促進している。文化装置のデジタル化の実態を探りながら情報社会の今後を考察する。

マシンの園
人工生命叙説
C.エメッカ 佐倉 統 他訳 2800円
人工生命とは何か。我々はまだその真価を理解していない。デンマークの理論生物学者が哲学的な背景にまで踏み込んで、新たな科学の衝撃を論じる。人工生命研究のこの10年間の復習に、来るべき科学革命の予習に、必読の書。

ドラキュラの遺言
ソフトウェアなど存在しない
F.キットラー 原 克 他訳 3400円
近代の表象システムをメディア環境から再考する著者の最新論考。「人間」という知的枠組を離脱した地点からラカン、アナログ、デジタルメディアをキーワードに人文科学の言説を問い直す。

シリーズ/情報科学の数学
現代暗号
岡本龍明・山本博資 3800円
インターネットの急速な普及と商用化にともない、暗号論が広く実用に使われるようになってきた。本書はこの暗号論を最近の成果や動向を踏まえたうえで、基礎から応用までを解説する。

脳の計算理論
川人光男 5500円
脳研究のための新しいアプローチ、計算理論について、入門から最先端の研究まで網羅する。ニューロンと神経回路のモデル、運動制御、小脳の学習、視覚の計算理論などについて解説する。

※価格は税別

〒102-0072 東京都千代田区飯田橋2-11-3
TEL.03-3261-7821 FAX.03-3239-2178

Inventor の Viewer クラスを，Tk の1つの「新しい widget」としてラッピングして取り込む方法である．Ver.2.x が Viewer インスタンスを canvas Widget に詰め込もうとしたのとは大きな違いがある．

◆ MegaWidget 実装手順

① Inventor の各 Viewer クラスを Tk コマンド化：ラッピングしたクラス
- SoXtRenderArea
- SoXtWalkViewer
- SoXtExaminerViewer
- SoXtPlaneViewer
- SoXtFlyViewer

② Inventor Viewer のポインタを獲得して，それを新 Widget に登録

③新 Widget のイベントハンドラの作成

この実装により，通常の Tk の pack コマンドによって，Tk の Widget の1つとして Inventor View が取り扱われるようになった．結果として，図3，リスト4の例に示すように，InvenTcl 上で複数の Inventor Viewer と Tk Widget が自由に配置できるようになった．

4 実行環境

InvenTcl は現在 Ver.3.0b であり，実行確認環境は下記の通りである．なお，Tcl/Tk は Ver.7.x からパッケージシステムが導入され，コア修正版だった［incr Tcl］も Ver.3.0 から外部パッケージとして提供されている．InvenTcl もこれに追従し，3.0b 版としてパッケージモジュールとしての提供が可能となっている．

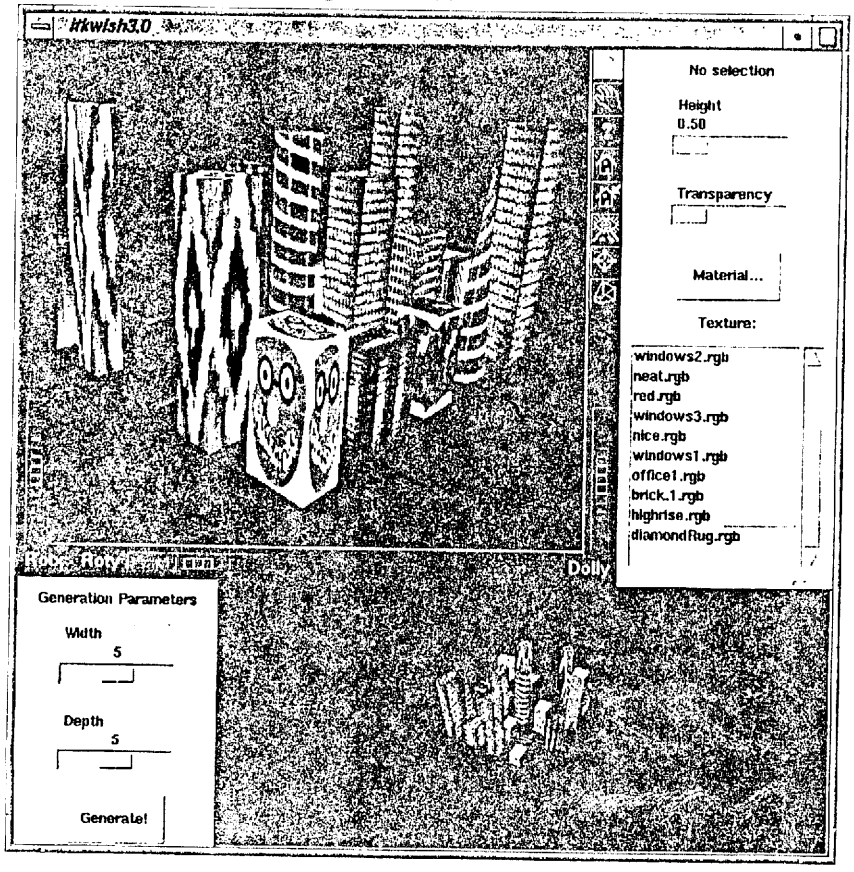
◆ InvenTcl 3.0 の確認済み実行環境

［ハードウェア］
- SGI Indigo2
- O2（IRIX 6.2，6.3，6.5）
- メモリ 128 MB 以上

［ソフトウェア］
- Open Inventor Ver.2.1.2 以上
- Tcl/Tk Ver.8.0
- incr Tcl Ver.3.0

InvenTcl の起動には，itkwish からパッケージをよぶだけでよい（リスト5）．なお，1.0版に比べて，コンストラクタコマンドとして新たに *new* などを導入して（リスト2，リスト4参照），3.0b 版ではかなり Inventor とのシンタックスの親和性が改善された．InvenTcl を使った例題やアプリケーションには，Inventor Mentor にある例題や，簡単なガイドエージェントと結合したウォークスルーシステム[5]などがある．

〈図3〉 Citydemo：MegaWidget の実現例



5 おわりに

当初困難と思われた 3D-MegaWidget と，高速表示を維持するイベント制御が，Inventor および Tcl/Tk のコアに触れずに実現でき，Tcl/Tk/［incr Tcl］のパッケージとして提供することが可能となった．このパッケージ化と MegaWidget の実装により，ユーザの利便性（インストールおよびプログラミング）が高まったと考える．InvenTcl は 1998 年1月に 1.0α 版を試供公開して以来，更新版の提供が遅れていたが，本稿で紹介した全機能をサポートする 3.0b 版の無償公開を現在検討中である．一度使ってみて感想をいただければ幸いである．

なお，InvenTcl で複雑なシーンをモデリングするのは，たとえスクリプト言語になっても，やはり骨の折れる作業である．きっとモデラが欲しくなるだろう．InvenTcl をベースにしたモデラを開発することも可能であるが，Alias，SoftImage，3D

〈リスト4〉Citydemo.tcl（図3）を表示するためのプログラム（抜粋）

```
# Mega Widget sample (citydemo.tcl)

# トップ 領域の widgets:
#   City を概観して、オブジェクトを選択できる
ExaminerViewer
#   [new],[new3] はインスタンスのコンストラクタ
ExaminerViewer .t.v
set ra [.t.v getRenderArea]
$ra setBackgroundColor [new3 SbColor 0.04 0.17 0.32]

  # パラメータ設定用 widgets
  frame .t.f
  ...
  pack .t.f.label .t.f.height .t.f.color .t.f.edit \
            .t.f.texlabel .t.f.tex -side top -ipady 2m

# ExaminerViwer widget と Tk widget のパッキング
pack .t.v .t.f -side left -fill both \
             -expand true -ipadx 2m -ipady 2m

# ボトム領域の widgets
#
frame .b.f
ExaminerViewer .b.cam
[.b.cam getRenderArea] setDecoration 0
.b.cam configure -height
...
pack .b.f .b.cam -side left -fill both -expand true

# 全体表示
pack .t .b -side top -expand true -fill both

# セレクションコールバック生成とコールバック関数の登録
#
set root [new SoSelection]
$ra setGLRenderAction [new SoBoxHighlightRenderAction]
$ra redrawOnSelectionChange $root

$root addSelectionCallback selCB NULL
$root addDeselectionCallback deselCB NULL
```

〈リスト5〉InvenTcl の起動

```
% itkwish3.0
itkwish> package require invenTcl
-------Initializing invenTcl 3.0b-------
Binding low-level functions..... Done
Loading [incr Tcl] auto index..... Done
Starting Inventor Viewer..... Done
-------Initialization Successful--------
itkwish> source citydemo.tcl
itkwish> ...
```

StudioMax などすでに優れた CG モデラが多数世の中に存在しているので，これらを使って作成した iv ファイルを使えるようにするほうが手っ取り早いと考えている．InvenTcl は，Inventor 1.0（すなわち VRML 1.0）レベルのスクリプトファイルを読み込むことはすでに実装できている．*ICreateName* を使えば，iv で定義されているオブジェクトに，インタラクティブに InvenTcl のオブジェクト名を付ける機能があり，すでに適用範囲はかなり広い．VRML 2.0 対応インタフェースやプロシージャ（コンストラクタなど）のオーバーローディングが今後の課題である．また，InvenTcl は IRIX 版のみであるが，Linux および NT へのポーティングを検討している（http://www.mic.atr.co.jp/dept2/inventcl 参照）．

謝辞 InvenTcl の開発には，これまで ATR に滞在した Sidney Fels，江谷為之，Armin Bruderlin，Silvio Esser の各氏がかかわっている．貢献に感謝します．また，インプリメンテーションでご協力をいただいている（株）東洋情報システムの川越一宏氏に感謝します．

参考文献

[1] J. Wernecke : "The Inventor Mentor", Addison-Wesley, New York, 1994.

[2] J. K. Ousterhout : "Tcl and the Tk Toolkit", Addison-Wesley, New York, 1994.

[3] M. McLennan : "[incr Tcl] : Object-oriented programming in Tcl", 1st Tcl/Tk Workshop, CA, University of Berkeley, 1993.

[4] W. Heidrich and P. Slusallek : "Automatic Generation of Tcl Bindings for C and C++ Libraries", Tcl/Tk Workshop, 1995

[5] S. Fels and K. Mase : "InvenTcl : A Fast Prototyping Environment for 3D Graphics and Multimedia Applications" 1st International Conference on Advanced Multimedia Content Processing (AMCP'98), pp.163-178, 1998.

●間瀬健二
（ませ・けんじ）
ATR 知能映像通信研究所
mase@mic.atr.co.jp