グラフィクスとCAD 90－1
（1998. 4. 24）

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

間瀬 健二　　シドニー・フェルス　　江谷 為之
{mase,fels,etani}@mic.atr.co.jp
ATR 知能映像通信研究所

あらまし

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の分野でのプログラミングに適した3Dグラフィックスツールキット、InvenTclを提案する。InvenTclは、Tcl/Tkを拡張することでOpen Inventorにスクリプトベースのプログラムインタフェースをもたせたインタプリタ型の3Dグラフィックス言語である。実装上はOpen Inventorのクラスライブラリを Tcl/Tkに埋め込み、Open Inventorのイベント管理をTclに取り込むことなどによって、3Dシーン記述のための高次のインタフェースを提供することが可能となった。本文ではInvenTclの実装における要点を詳しく述べ、InvenTclによるシーン生成の例題を紹介する。

A VR Scripting Language for Rapid Prototyping of Virtual Environment

Kenji Mase　　Sidney Fels　　Tameyuki Etani
{mase,fels,etani}@mic.atr.co.jp
ATR Media Integration & Communications Research Laboratories

Abstract

InvenTcl is a Tcl/Tk-based interpretive and 3D graphics language that includes the 3D VR modeling library, Open Inventor. Open Inventor is a easy and powerful VR programming toolkit which allows a programmer to describe three-dimensional scenes in an abstract structure of graphics primitives. To create InvenTcl, the Open Inventor toolkit is "wrapped" inside the interpreter Tcl/Tk and its object-oriented extension [incr Tcl]. Thus, the Open Inventor objects are created and manipulated interpretively through a Tcl shell window. Moreover, the 3D objects created can be binded to any Tcl scripts. InvenTcl is a powerful toolkit for prototyping VR world, graphics educations, and agent-oriented programming.

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(3D)グラフィックスを表示したり、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。そして、様々な使いやすい3D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、その多くはコンパイル作業を必要とするため、ラピッドプロトタイピングには適さないという問題がある。このことは、気軽に3次元世界を構築してテストをしたり、既に作った世界にすこしずつ手をいれながら作業をするユーザにとっては不便である。例えば、3Dグラフィックスや VRをこれから勉強しようというユーザは、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。

一方、市販のVRモデラーやオーサリングツールではビジュアルなユーザインタフェースが提供され、インタラクティブな3次元VR世界を構築することは、それほど困難ではなくなってきている。しかしながら、これらのシステムはオブジェクトの振る舞いをモデリングすることはできるが、プログラミングすることは困難である。したがって、例えば自律的な振る舞いをするオブジェクトをデザインしようとすると、全ての場合を想定した振る舞いのモデリングをする必要がある。また、これらのシステムは、とにかく3次元世界を表示したいというユーザには便利であるが、グラフィックスの理論を理解したり習得しようとするユーザには不向きである。

また、VRMLのようにエンジンをプログラミングすることでアニメーションを定義したりすることができるが、プログラミングとモデリングのフェー

ズが切り放されてしまうため、結局、作業が行きつ戻りつとなって、効率が悪い。ところが、マウス操作による直接モードとプログラムによるスクリプトモードの両方の良さを組み合わせたツールは少ない。Open Inventorなども、出来上がったシステムは外部プログラムとのフレキシブルなインタフェースを持たないという欠点があり、スクリプトベースのプログラミング言語としては不十分である。

本文では、InvenTclと呼ぶ、Tcl/Tkを拡張してOpen Inventorへのスクリプトベースのプログラムインタフェースをもつた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]を統合している。Open Inventorのライブラリを Tcl/Tkに埋め込むことによって、3Dシーン記述のための高次のインタフェースを提供することが可能となる。

InvenTclは、Tclに対しては3次元インタフェースを提供し、Open Inventorに対してはインタプリターインタフェースを提供していると考えることができる。すなわち、Tkを使った2次元のインタフェースwidgetを使いつつ、Tclコマンドにより3次元シーンへの直接アクセスが可能となる。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プログラミング言語として適していると考えられる。すなわち、InvenTcl開発の目的は、(i) Open Inventorのインタプリタ版を提供することと、(ii) 3次元Tk Canvas Widgetを提供することにある。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。

- 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との組合わせが容易
- 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

なお、関連研究としてはつぎのようなものがある。まずTkに関して、OpenGLを利用した3次元拡張の試み[4]がある。しかし、これらはオブジェクト指向プログラミングができないという欠点がある。他のインタプリタ言語をベースにしたものに、Modula-3をベースにしているObliqインタプリタとAnim3Dライブラリを使ったObliq-3D[5], Pythonインタプリタと Direct3Dをベースにした Alice[6]、そして最近では独自の形式による TBAG[7]などがある。これらはいずれもオブジェクト指向プログラミングが可能で、かつラピッドプロトタイピングを指向しているという共通点がある。アニメーションの実現方法やベースとしている言語の特徴による違いがあるが、上記の中では、InvenTclはキーフレームベースのアニメーションを行なうAliceに近いといえる。InvenTclとこれらとの最も大きな相違は、ベースにしているTcl/tkおよびOpen Inventorのユーザへの利便につきるといってもよい。他の便利な言語が多く開発されている中で、これらの言語には根強いユーザがいる。特にOpen InventorはOpenGLのように詳細までプログラミングする必要のないVR構築にとって便利なツールであるが、一旦シーンを作ってしまうと外部プログラムとの結合の融通が効かなかったり、シーンの修正に手間がかかるいう欠点があり、高度で拡張性のあるVRシステムを作る言語としては物足りないところがあった。

以下、本稿では、2節でOpen Inventorの変換の中心であるライブラリのラッピングの具体的方法について詳しく述べ、InvenTclの例題を示す。3節では、InvenTclを用いて作成したアプリケーションとして、部屋の仮想ウォークスルーシステムを紹介する。

2 Open Inventorのラッピング

InvenTclを生成するためには、まずOpen Inventor（以下、とくに断らない限りInventorと省略する）のライブラリをラッピングして、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にアクセスできるようにしなければならない。これにはstatic関数も含まれる。具体的にはTclのシェルから入力されたコマンドは一旦ハッシュテーブルで照合され、Inventor側のどのクラスのどのメソッドを呼びだせばよいかの判断が行なわれる。次に、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する必要がある。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いれている。また、Tcl/Tkの良さであるバインディング機能をいかす

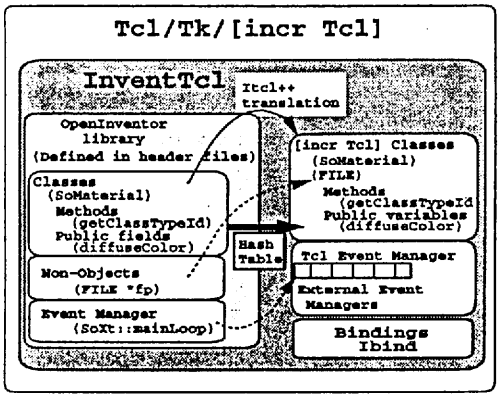

図 1: InvenTcl における Open Inventor, Tcl/Tk, [incr Tcl] の関係を示すブロックダイアグラム

ために、3 次元 object にもバイディングできるようにしている。こうすることで、例えば、3 次元 object をクリックしたときに、それにバインドされた Tcl スクリプトが実行されるようになる。実際にはマウスイベント、キーボードイベントなどのイベントを扱えるようにする必要がある。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する必要がある。例えば、enumerated 型、array 型 (1D, 2D, 3D, nD)、ファイルポインタ、関数ポインタなどである。これらの変換のほとんどは、Inventor のヘッダーファイルの変換で済む。パーザを使って class と method の変換は全部自動でできるが、非 object パラメータやイベントチェックなどはプログラミングが必要である。InvenTcl, Open Inventor, Tcl/Tk および [incr Tcl] の関係を図 1に示す。

2.1 クラスの変換

Inventor のクラス構造を [incr Tcl] に変換するには Itcl++[8] と呼ばれるツールを用いる。Itcl++ は C++ コードの変換を含むツール群で、これを用いて、Tcl から C++ のメソッドや変数にアクセスできるようになる。Itcl++ を使うと、クラスライブラリのヘッダーファイルを [incr Tcl] のクラスに変換でき、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 (継承) 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る。このように 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なわれる。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。以下そのうち代表的なものを述べる。

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。Itcl++ 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

のように使うことができる。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 つ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なうようにした。

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 *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、*ambientColor* フィールドも生成されるので、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、*ambientColor* 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 *ambientColor* のインスタンスを含むことができ、フィールドにアクセスできるようになる。これはパーザーの部分的な修正で可能となった。

2.2 イベント管理

典型的な Inventor のプログラムにおいては、初期化をして、シーングラフを生成し、ユーザのインタラクションを設定したあと、イベント管理をする無限ループである *SoXt:mainLoop* 関数を起動している。InvenTcl を実現するためには、Tcl/Tk 側も Inventor のイベントを全部管理する必要がある

ため、この構造を変更している。それにはInventorのイベント管理をTcl/Tkのイベント管理下におき、Inventorのイベントを Tcl/Tk 側がメインのイベントキューで発見すると、その1つのイベントを処理するInventorのプログラムを起動し、Tcl/Tkに制御を返すようにした。ユーザにとっては、Inventorのプログラムと同じように *SoXt:mainLoop* 関数を発行するだけでよいが、内部ではTclにイベントハンドラを埋め込む処理が行なわれている。

この設計は、すべてのイベント処理を一旦Tclで処理することになるため、現在のコンピュータパワーやアーキテクチャでは高速なインタラクションやアニメーション制御をするようなアプリケーションには不向きである。Tclに戻す必要のないイベントに限って、Inventorだけに閉じて処理することがon/offで設定できるように、改造する計画がある。

2.3 オブジェクトへのバインド

Tclではbindコマンドを使ってTkのオブジェクトに対し、イベントに応じた処理を登録することができる。同様のことをInventorのビューワーで出きるようにするために3Dバインドを用意する。こうして、ビューワーに表示されたオブジェクトをマウスでクリックすると、バインドされたTclスクリプトが実行するようにできる。

実装上はIbindという関数を作った。Ibindは4つの引数をもつコマンドで、それらは、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、およびTclスクリプトである。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、次のように使うことができる。

```
Ibind $cone $root <Button1> \
                {puts "Hello world!"}
```

こうすると、coneオブジェクト上でマウスボタン1を押すと、"Hello world!"という文字が出力される。これを実現するために、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全てのバインドとマッチングをとるようにした。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。

2.4 非オブジェクト構造の変換

Inventorはオブジェクト指向の言語であるが、非オブジェクト構造のものも少なくない。これらは大

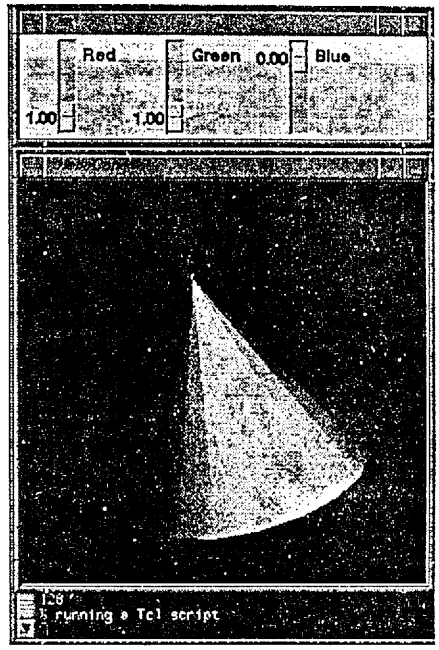

図2: 円錐の例題プログラムの表示画面：Tkで書かれた上部のGUIで円錐の色を制御できる。また、最下部はTclシェルのウィンドウwishで、円錐上でボタン1を押したときのメッセージが表示されている。

抵、メソッドへの入力パラメータとして使われる。これらは、itcl++では生成できないので、マニュアルでプログラミングした。主なものを列挙すると、(i) 数え上げ(enum)型、(ii)配列、(iii)ファイルポインタ、(iv)関数ポインタである。特に、配列型のデータを扱うArrayクラスはincr++が1次元arrayオブジェクトしか扱えないため、拡張する必要があった。現在、Array2D, Array3D, Array4Dのクラスを用意してある。

2.5 例題

以下はInvenTclによるプログラミングの例を示す。この例題では、図2のような円錐1個だけの簡単な3次元シーンを定義して、TclからInventorへのインタラクションと、InventorからTclへのインタラクションを説明する。図3は図2の円錐を表示して、スライダーで色を変えたり、円錐をボタン1でピックすると"running a Tcl script"というメッセージをTclシェルウィンドウに出すプログラムの

```
# 円錐を表示するプログラム

# 1: ウィンドウとシーンデータベースの初期化
set window [SoIt::init "HelloCone" "HelloCone"]
# 2: シーングラフのルートを生成
set root [SoSeparator::Constructor]
$root ref
#  カメラと照明の設定
set camera [SoOrthographicCamera::Constructor]
$root addChild $camera
$root addChild [SoDirectionalLight::Constructor]
# 3: トラックボール、材質の設定
set trackBallManip \
    [SoTrackballManip::Constructor]
$root addChild $trackBallManip
set coneMaterial [SoMaterial::Constructor]
$root addChild $coneMaterial
# 4: 円錐オブジェクトを追加
set cone [SoCone::Constructor]
$root addChild $cone
# ビューワのパラメータ
set ra [SoItRenderArea::Constructor $window]
$ra setTitle "HelloCone"
$camera viewAll $root [$ra getViewportRegion] 1
$ra setSceneGraph $root
$ra show
SoIt::show $window
# 5: Inventor のイベント処理開始
SoIt::mainLoop
# 6: tcl/tk とオブジェクトの結合
set coneColor \
  [lindex [$coneMaterial configure -diffuseColor] 2]
set red 0.5
set green 0.5
set blue 0.5
# 円錐の色とスライダを結合する手続
proc changeColor {val} {
   global coneColor red green blue
   $coneColor set Value2 $red $green $blue
}
# 7: スライダー
toplevel .ex
scale .ex.r -from 0 -to 1 -resolution 0.01 \
   -label Red -variable red -command changeColor
scale .ex.g -from 0 -to 1 -resolution 0.01 \
   -label Green -variable green -command changeColor
scale .ex.b -from 0 -to 1 -resolution 0.01 \
   -label Blue -variable blue -command changeColor
pack .ex.r ex.g .ex.b -side left
# 8: 円錐のところでボタン1を押すとメッセージを表示する
Ibind $root $cone <1> {puts "running a Tcl script"}
```

図 3: InvenTcl のプログラム例：円錐を表示して、スライダーで色を変えるプログラム

具体例である。[1]

この簡単なプログラムで InvenTcl のおもな特徴を説明することができるが、その自由度を文章で説明するのは困難である。次の節では、実際のプログラム開発で利用した例を示して、その説明を試みる。

3 応用

我々はエージェント技術と携帯端末を組み合わせた、展示スペースや博物館などのガイドシステムの開発を進めている [10]。ユーザの位置情報を使った3次元仮想空間を使った物理マップは、ガイドする際に鳥瞰的な情報を提供したり、Augmented Reality の考え方による付加情報の提示のための基本的な空間となる。そのような空間には、オブジェクト指向の属性をもった仮想物体を置いたり、エージェントを表示することが求められるが、展示スペースのガイドなどの応用を考えると、展示内容に対するフレキシビリティが必要である。InvenTcl はそのようなシステムのベース言語に適している。

そこで、InvenTcl のプログラミング能力評価を兼ねて、研究所公開時のガイドシステムとして3次元ウォークスルーとそのビルダーを作成した。図4にそのビルダー兼2次元のウォークスルーインタフェースおよび3次元ウォークスルービューワーを示す。ビルダーで線を引くと InvenTcl は3次元の壁を生成することができる。ビルダーはウォークスルーの視点位置制御のインタフェースにもなっている。また壁には属性エディタがバインドされており、最新のテクスチャ画像を貼付けて、最新の様子を伝えることができるようになっている。また、人形型のオブジェクトは来訪者につきそうパーソナルエージェントを現しており、クリックするとユーザ情報がポップアップされ、ユーザが移動すると場所が変るようになっている。このシステムは、ユーザ追跡センサーと組み合わせ、部屋の要所で情報提供するシステムであるが、特別な 3D グラフィックスの教育を受けていないプログラマが、約1週間で、このような、簡単であるが、具体的なアプリケーションを完成できたという良い例である。このプログラマは実際には図のインタフェースに至るまでいろいろな方法をためし、試行錯誤を繰り返している。出来上がったインタフェースの評価より、試行錯誤を容易にした InvenTcl に注目したい。

[1] 紙面の都合で省略するが、円錐の表示については Inventor Mentor[9] に C++ によるサンプルプログラムがあり、Inventor と InvenTcl のプログラム構造やクラス名の一貫性や、記述の違いを見ることができる。

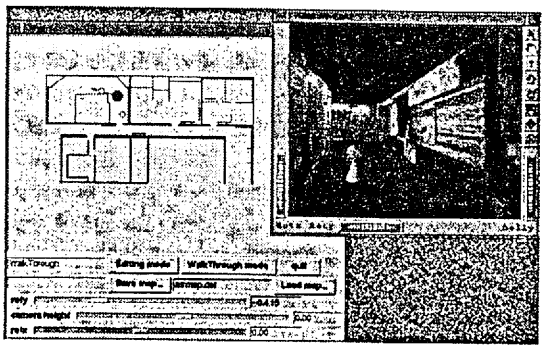

図 4: InvenTcl を使って、ラピッドプロトタイピングした VR シーンの例：壁には属性エディタがバインドされており、最新のテクスチャ画像を貼付けて、ポスターの様子を伝えている。また、人形型のオブジェクトは来訪者につきそうパーソナルエージェントを現しており、クリックするとユーザ情報がポップアップされ、ユーザが移動すると場所が変る。

4 おわりに

InvenTcl は当初、エージェント記述言語として開発をはじめた。3 次元 VR 空間に存在するエージェントの表示と振る舞いを記述するスクリプト言語を目指したのである。現時点では、非常に簡単で、簡素ではあるが、一応エージェントの振る舞いと表現が可能になっている。しかし、アニメーションの表現能力を引き出すにはさらなる工夫が必要である。

まえがきでも述べたように、この種のラッピングは既にいろいろな試みがなされている。既存のグラフィックスライブラリのラッピングは、グラフィックスプログラマの入り口としての InvenTcl のようなインタプリタ型言語を提供することで、将来、ライブラリのエキスパートとなるプログラマへのスムースなプログラミング環境を提供できると信じている。InvenTcl は、Inventor と Tcl/Tk をベースにしているが、この考え方は他の言語（3D 操作をする関係からオブジェクト指向が望ましいが）にも適用できると考える。

現在かなりの部分について実装が完了し、簡単な例題はできるようになった。しかし当初の大きな目的のうち、3D Widget についてはまだ手をつけていない。これについては、今後の大きな課題である。また、擬人化エージェントのプログラミング言語として考えると、エージェントキャラクタのアニメーションを制御するにはリアルタイム性に欠るという欠点がある。これは、Inventor のイベント制御と Tcl のイベント制御の結合が上位の階層でしか実現できなかったことが原因である。これを理想的に回避するにはそれぞれの処理系にまで踏み込む必要があるが、公開された Tcl/Tk に対し Open Inventor 側は困難が付きまとうと思われる。

なお InvenTcl はすでに InvenTcl1.0α 版が完成し試供公開しており、InvenTcl1.0β 版のリリースを 98 年 5 月に予定している。本稿で紹介した機能のうち、overloading や array クラスなどは 1.0β 版からサポートされるものである。

謝辞

本稿の内容は、第二著者の Sidney Fels が中心となって行なった仕事の成果である。初期システムのインプリメンテーションに関わった Armin Bruderlin 氏, Silvio Esser 氏、モジュールのインプリメンテーションと変換作業でご協力をいただいた（株）東洋情報システム川越一宏氏に感謝します。日頃ご指導いただく酒井保良会長、中津良平社長ならびに、ディスカッションして頂く第 2 研究室の皆様に感謝します。

参考文献

[1] Open Inventor Architecture Group: "The Inventor Reference Manual", Addison-Wesley, New York(1994).

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

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

[4] B. Paul: "Togl: Togl allows Open GL or MESA to Render Graphics into a Special Tk Canvas", http://www.ssec.wisc.edu/ brianp/Togl.html.

[5] M. A. Najork and M. Brown: "Obliq-3D: A high-level, fast-turnaround 3D animation system", IEEE Trans. on Visualization and Computer Graphics, pp. 175–193(1995).

[6] R. Pausch et al.: "Alice: A Rapid Prototyping System for 3D Graphics", IEEE CG&A, 15, 3, pp. 8–11(1995).

[7] C. Elliott, G. Schechter, R. Yeung and S. Abi-Ezzi: "TBAG: A High Level Framework for Interactive, Animated 3D Graphics Applications", SIGGRAPH'94 Conference Proceedings(1994).

[8] W. Heidrich and P. Slusallek: "Automatic generation of tcl bindings for C and C++ libraries", Tcl/Tk Workshop(1995).

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

[10] Y. Sumi, T. Etani, S. Fels, N. Simonet, K. Kobayashi and K. Mase: "C-map: Building a context-aware mobile assistant for exhibition tours", The First Kyoto Meeting on Social Interaction and Communityware(1998). to appear.