インタラクティブストーリー記述言語を基にした「会話する」擬人化エージェントの開発

The Development of "Talk" Personified Agent Based On Interactive Story Description Language

出渕亮一朗
Ryoichiro Debuchi

株式会社アトム
atom Co., Ltd.

Abstract: We propose Interactive Story Description Language base on XML (StoryGraph) that controls "Talk" personified agents (Talking Heads) Based on it, we introduce the example of developing "Talk" virtual characters by real time 3D computer graphics recognizing the voice.

１．はじめに

対話型エージェントの研究開発において、アウトプットとしてコンピューター画面に視覚表示される擬人化エージェント（バーチャルキャラクター）を応用する方法ある。
文字や音声のみで応答する擬人化エージェントよりも、ヒューマンインターフェースとしてより親近感、没入感、興味の誘引等を与えることが期待されるため、注目されている分野である。

Nobert Wienerのサイバネティックス(Cybernetics)理論によると、「生物が外界からの情報を感覚器を通じて獲得し、大脳中枢で処理し、効果器、すなわち手足の筋肉系の行動になって再び外界に働きかける過程は、機械のシステムと同じ次元で議論できる。」のであるが、対話型エージェントとしての技術では、ユーザーからのエージェントへの入力として、マウス等のデバイスによる位置入力、キーボード等のデバイスによる文字テキスト入力、あるいは、マイクロフォンからの入力音声の音声認識等があり、エージェントからユーザーへの出力として、画像/アニメーション表示、文字テキスト出力、あるいは、合成/録音音声出力等があげられる。

「大脳中枢での処理」にあたる過程は、コンピューターの計算処理により行われるはずであるが、この入力と出力を結びつける「知能」部分を記述する一種のプログラミング言語を考案する手法がある。

この記述言語として、例えば、W3Cで提案されているVoiceXML[1]がある。VoiceXMLとは、インタラクティブな音声応答サービスの構築を容易にするためにXMLベースの技術を応用し、音声認識・音声合成機能を持ったブラウザを介した表現を行うためのレイアウトや構造を定義する共通的な記述言語である。VoiceXMLは音声認識とそれを元にした応答やコントロールにフォーカスされた言語であり、ここで注目している、バーチャルな「身体」を持ったエージェントを外側から制御するモジュールのひとつあたる。

視覚表示部分にあたるバーチャルキャラクターの中でも、「トーキングヘッド」(Talking Heads)[2]と俗に呼ばれているものは、特に顔を中心とした人間の上半身をメインとした「会話する」キャラクターであり、音声と口唇の同期(リップシンク)技術を重要としている。

基本的なアイデアの原型は、80年代半ばに英国で制作されたＴＶ番組シリーズ「Max Headroom」[3]である。この分野は現在までに QEDSoft [4]等、数多くの研究開発例、応用例、実用化例がある。

ストーリーを持ったキャラクターアニメーションを記述する言語として、NHK放送技術研究所、TVML研究開発チームによるTVML[5] がある。TVMLは「TV番組記述言語」として考案され、テキスト文章をもとにキャラクターに合成音声で「会話」を行わせたり、「歩く」等のキャラクターの行動の定義、あるいは、スタジオ舞台設定といった視覚効果の記述や、サウンドエフェクトの指定等をすべて簡単な定義コマンドにより記述しようと試みたものである。

TVMLはTV番組や映画のように、一方通行の時間軸に従ったストーリーを再生することを目的とした言語であり、インタラクティブ性を与える場合は外部からの制御プログラム等を必要とする。

音声認識、音声出力、キャラクター表示を統合したシステムとして現在までに開発されているものとして、まず、1997正式リリースのMicrosoft Agent[6]があげられる。

音声認識・合成による対話機能とキャラクターアニメーションによる動作の表示機能を含む。また、ActiveX対応のスクリプトで、アニメーションをプログラム制御する。

Galatea プロジェクト[7]は、「擬人化音声対話エージェント」

出渕　亮一朗　株式会社アトム　女子美術大学非常勤講師
〒160-0022　東京都新宿区新宿 3-1-13 京王新宿追分ビル 6F
Tel: 03-5312-2811, Fax: 03-5312-2812,
E-mail: debuchi@atom.co.jp

開発キットである。音声対話と顔画像に重点を置き、オープンな環境を提供することを目的とする。

音声認識、音声合成、写真から3Dモデルを生成する顔画像合成、ユーザーとエージェントの対話の流れを記述する対話制御の各モジュールを持ち、これらの機能を「Galatea アーキテクチャ」のもとに統合して動作させる。

筆者は、2000年度に取り組んでいた音声認識する擬人化エージェントの具体的な研究開発を行うにあたって、入力部と出力部を結びつけるインタラクティブな処理を記述する側面と、出力としてのバーチャルキャラクターのアニメーションや視聴覚効果を記述する側面の両方を兼ね備えたコンテンツ記述スクリプトの必要性から、インタラクティブストーリー記述言語、StoryGraphを考案した。従来のストーリーを記述するデータ形式では、TVMLのように直線構造で記述されたデータを逐次辿るか、ある種のエキスパートシステムのように階層構造化されたデータをトップダウンで辿るのが基本である。

３番目の構造として、インタラクティブな「分岐ストーリー」を記述できるように、「ネットワーク」的、あるいは「相互リンク」的な構造を持たせたデータ形式が考えられる。StoryGraphは、分岐（インタラクティブ）ストーリーを記述するために定義されたXMLデータフォーマットである。（図２）

ユーザーとエージェントとのインタラクションを実現するために、相互リンクされた単位ノードを基本とするスクリプトプログラミングを行い、またこれに、リアルタイム3D コンピューターグラフィックスでのアウトプットを前提とした、バーチャルキャラクター（擬人化エージェント）操作、及び、描画画面操作コマンドを連結している。

StoryGraphの基本構造として、ストーリーの構成単位としてのノード(StoryNode)を考え、その相互リンクでインタラクティブストーリーを記述する。

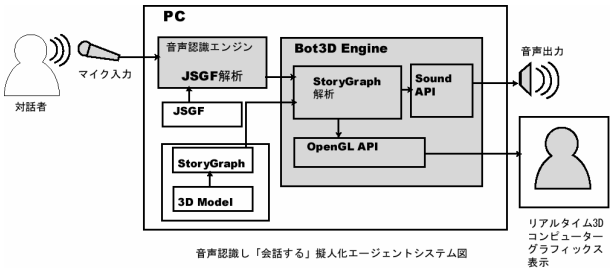
ストーリー実体として、ひとつのノードには「文章」を含めることができる。文章はテキストベースの時間構造を持ち、キャラクターの会話に伴う口唇の動き（リップシンク）の他、キャラクターの視覚言語アニメーション（3.4で後述）、あるいは、様々なイベントアニメーションもテキストベースでアニメーションタイミングが指定される。

入力系と出力系もノードごとに設定される。入力系は音声認識等を想定したインターフェースからの入力に反応する。出力系には上記アニメーションのようなコンピューターグラフィックス表示を前提とした、キャラクターや表示世界（ワールド）に関するビジュアル操作のコマンド群を持つ。（このコマンド群を以下、モデル&グローバルファンクションと呼ぶ。）また、会話音声出力やサウンド出力も含む。

さらに、ノードごとに局所的スクリプティングが記述され、現在辿っているノードのスクリプトが、その都度評価される。

こういった、インタラクティブストーリー記述言語の応用例として、

・マルチユーザーワールドやオンラインゲームでの無人キャラクター(bots, non player characters / NPC)、
・インターネットサイトのナビゲーター(talking heads)、
・AI機能を備えた会話応答エージェント(chat-bots)、
・音声認識して会話するバーチャルキャラクター、
・ロボットコントロール 等があげられる。



（図１）音声認識し「会話する」擬人化エージェントシステム図

２．StoryGraphの歴史

StoryGraph 1.0　筆者が 2000-2001年に擬人化エージェントの研究に取りかかった際に考案したものである。

基本構成として、ストーリーノード/ノード間の分岐接続/パラメータ設定とスクリプト/音声ファイル定義/文章定義 /モデル&グローバルファンクションと、StoryGraphの基本アイデアはすべて含んでいた。但し、この当時は、まだXML設計ではなく、VRML(Virtual Reality Modeling Language)に習ったファイル形式となっていた。

StoryGraph 2.0　2002年に採択された、経済産業省 平成１４年度「次世代コンテンツ支援事業」による研究開発、「3D『トーキングヘッド』のための組み込み描画エンジンの開発」に伴った、StoryGraphフォーマットの見直しと再設計による。

すべての関連ファイルをXML定義に変更したほか、音声合成や音声認識を応用した音声ファイルの音素解析を前提とした、文章定義ファイルにおけるリップシンクの詳細定義、コンピューターグラフィックス描画関連のモデル&グローバルファンクションの詳細定義、メインキャラクター以外のサブキャラクターや、小物、舞台装置等のモデルのロードと設定/操作定義を含んでいる。

StoryGraph 3.0　現在、定義構想の段階のものであり、具体的な開発物への実装はまだなされていない。

大きな変更点として、2.0までは、「文章」定義ファイルを外部に置き、StoryGraph内ではその中のタグ名称を参照する設計となっていたものを、StoryGraphのノード内に直接記述できるようにすることである。

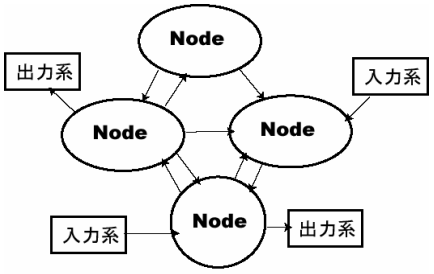
また、文章定義ファイルは、2.0段階では、音素と音素期間情報の中に、視覚言語情報のタグを埋め込む形式であったものを、3.0では日本語テキストの中の任意の位置に直接埋め込むことができるようにする。

３．StoryGraphの基本概念

３．１ ストーリーノード（StoryNode）

StoryGraphでは、ストーリーとはその展開における、それぞれの小ステージ（ストーリーノード：StoryNode）の組み合わせで成立すると考える。各ストーリーノードでは、キャラクターのひとまとまりの会話がリップシンクと音声を伴って再生され、 同時にキャラクターの持つ視覚言語情報のアニメーション再生、その他、小道具、舞台背景などストーリーを組み立てるために必要な視聴覚表現に対して様々な事象が起こるとする。

また、ユーザーからの音声認識インターフェース等を経由した入力も、ノード毎に評価されるとする。



（図２）ストーリーグラフの概念図

３．２ 分岐

単純なストーリーでは、ストーリーノードが一次元的に連続される。ファンクションの種類によっては、例えばサウンドエフェクト再生と背景色のフェードインのように、スタートタイミングが重なり合い、並列進行することはある。

しかし、ユーザーとエージェントのインタラクションを考慮したインタラクティブストーリーにおいては、ストーリーノードが互いに複雑にリンクしあい、ネットワーク状のデータを形成すると考えられる。StoryGraphではこのネットワーク形成に必要なストーリーノード分岐を次の四つに分類している。

1)ダイレクト接続　── 最も単純な場合。次のノードへのリンクが、惟一に定まる接続。

2)自動確率分岐　── ある確率分布に従って、自動的に複数の異なるノードへ分岐する。

3)入力条件分岐 ── キー入力や音声認識インターフェース等ユーザー入力の条件に従って分岐する。

4)パラメーター条件分岐　── データ内部構造に持つ、スクリプト変数の条件に従って分岐する。

３．３ スクリプト

ストーリーノードには、局所的なスクリプトを持たせることができるとする。スクリプト言語は概念的には限定しない。たとえば、JavaScript等でのインプリメントを想定している。

パラメーター（変数）はStoryGraph全体でグローバルに設定することも、ストーリーノードのスクリプト内でローカルに設定することも可能である。現時点で経由しているストーリーノードに記述されたスクリプトが、現時点に評価/実行されるとする。

３．４ 視覚言語と音声/リップシンクの統合

現実の人間が言語を話すということは音声のみ使用しているのではなく実際は、1) 音声、2) 口唇と舌の動き、3) 会話動作、4) 感情表現の組み合わされたものであるはずだ。

1)は聴覚により認知されるものである。2)は視覚により無意識的に認知されるものであり、いわゆるリップシンク(lipsync)として、発音音素と口形の同期した対応付けを意味する。3)、4)は言語によらないものであり、いわゆるノンバーバル言語である。3)は「うなずき」など、頭/首の動き、さらには身振り手振りのジェスチャーまで含む。4)は目蓋、眉、頬などの顔面の筋肉の動きによる感情表現であり、視線や瞬きの速度や頻度を含めることもできる。

1)と2)の関係（リップシンク）についてはさまざまな研究がなされている。録音音声を使用する場合は、音声ファイルを音声処理技術を元に解析して、音声の音素と対応する発生期間を取得しこれをリップシンクに応用する技術がある。

また、合成音声を使用する場合は、内部的に保持している再生テキストに対応した音素と対応する発生期間を取得し応用する方法が有効である。

3)、4)ノンバーバル言語の研究に関しては未熟である。

バーチャルキャラクターにこのアニメーションを付加したい場合、一般的には3DCGアプリケーションにて文章ごとにデザイナーがすべて動作アニメーションを制作するか、モーションキャプチャーシステムによって、実際のアクターの動きを記録して使用する方法が利用される。

しかし、いずれも手間とコストのかかるものである。あるいは、単純なバーチャルキャラクターへの実装方法として、用意したノンバーバル言語アニメーションをランダムに挿入するという方式がよく取られている。

しかし、「肯定」の意味の「うなずき動作」、「疑問」の意味の「首傾げ動作」、 「否定」の意味の「首振り動作」、「好ましくない意思表示」の「眉しかめ」等など、会話動作、感情表現は、単にランダムに会話中で行われるものではなく、言語の形態素と同じく意味のある会話構成要素のひとつであるはずである。つまり、視覚言語として、必要な時に相応しいタイミングで会話中に挿入されるべきものである。

こうした視覚言語の要素化に関しては、"visume"(視覚素)という言葉が使われることもある。これは、"phoneme"（音素)に対応した造語である。

ここで提案するのものは、視覚言語を音声言語とまったく同列に扱い、テキストによって記述する手法である。

具体的には、言語形態素のように、例えば 、「否定」の意味で首を振る動作には、<motion action="no"/>を使用し、「疑問」の意味で首を傾げる動作には、<motion action="?" />、眉をひ

そめる感情表現には<emotion morph="frown">を使用するとする。表情を基本顔型に戻す、<resetEmotion>タグも定義する。
また、各タグ要素に対して時間長や誇張度などの属性を設けることにより、バリエーションを作り出すことも可能である。

例えば、これらの視覚言語タグを使用した例文はこんな感じとなる。全体をくくっている、<Chat>はひとつの「会話文章」を定義するタグである。

<Chat><emotion morph="frown" >ああ、嫌だ嫌だ。<motion action="no"/>
どうにかして勉強しないで学校の成績がよくなる工夫はないのかしら。
<motion action="?"/><resetEmotion></Chat>

これらの動作/感情記号は単にテキストに対応した時間位置での動作の始まりのきっかけを示す。リップシンク、動作、感情表現はそれぞれ独立したシステムであるため並列に動作する。

通常のアニメーション作成方法では時間の流れはフレームに制御される。しかし、この定義方法では文章テキストが時間を制御している。合成音声を使用する場合は、使用している合成音声エンジンのランタイムで視覚言語の再生タイミングが決まることとなる。録音音声を使用する場合は、もう少し複雑となる。(後述)

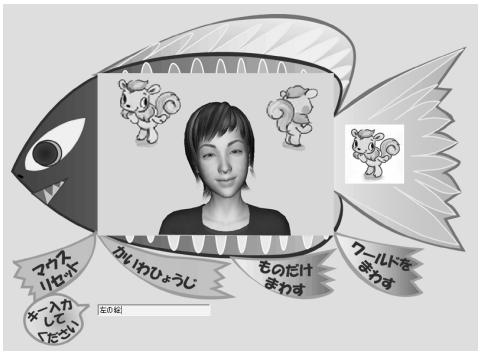
これらの視覚言語を表現するコンピューターグラフィックにおける技術は、リップシンクは口形のターゲットモーフィング技術、感情表現は、眉、目、頬、口等顔のパーツごとのターゲットモーフィング、動作はボーンアニメーション技術により実現されるはずである。

４．StoryGraphのサンプル

StoryGraphデータフォーマットの詳細に関しては、紙面に制限があるため、ここではふれないとする。

以下、StoryGraph 3.0記述にてサンプルを示す。(図３)このサンプルはウェブブラウザ表示のエージェントを想定している。

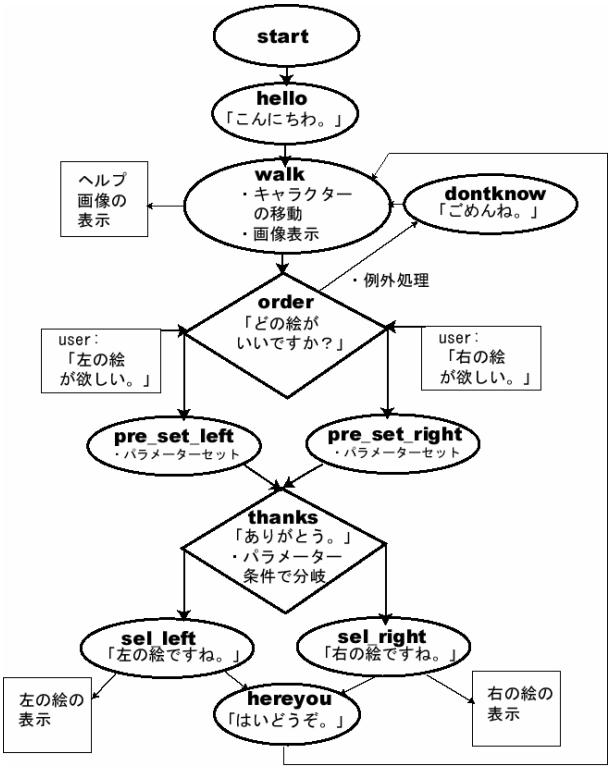
動作は、まず背景に2枚の画像を表示しキャラクターに説明させる。ユーザーにどちらか欲しい絵をマウスクリック、あるいは、キー入力で選ばせ、選んだ絵を画像GIFファイルにてウェブブラウザ内に表示するといったものである。(図４)



(図３) サンプルコンテンツのイメージ図

```
### サンプルのStoryGraph ####
<?xml version="1.0" encoding="Shift_JIS"?>
<StoryGraph version="3.0">
<SpecialNode>//特殊ノードの設定
        <exceptNode value="start" />
        <unknownNode  value="start" />
        <startNode  value="start" />
</SpecialNode>
<StoryAttribute>//アトリビュート設定
        <loadModel url="kaori.xml" name="kaori" />
        <loadBackImage  url="pictures.xml" name="pictures" />
        <grammerFile  url="illust_memo.xml" />
        <parameter name="select" value="no" />
</StoryAttribute>
<StoryNode name="start" > //ここから開始する
        <type  value="DIRECT" />
        <duration value="1.0" />
        <link value="hello" />
</StoryNode>
<StoryNode name="hello" > //「こんにちは」としゃべらせる
        <Chat><motion action="yes"/>こんにちは</Chat>
        <type  value="DIRECT" />
        <duration value="1.0" />
        <link value="walk" />
</StoryNode>
//質問と同時に絵を表示し、また、キャラクターを移動させる
<StoryNode name="walk" ><Chat>どの絵がいいですか<motion
action="?"/></Chat>
        <type  value="DIRECT" />
        <duration value="3.0" />
        <backImage  name="pictures" switch="ON" fade="ON"
duration="1.0"/> //背景画像のフェードイン表示
        <moveToModel model="kaori" tx="0" ty="-5" tz="-20"
duration="1.0" /> //キャラクターを移動する
        <link value="order" />
</StoryNode>
//「右」または「左」などとユーザーに答えさせ入力分岐する
<StoryNode name="order" >
        <type  value="INPUT" />
        <duration value="10.0" />
        <onInput value="TRUE" />
        <except value="order" />
        <unknown value="dontknow" />
        <eventOut value="help.gif" />  //ヘルプ表示
        <Select>//ユーザー入力分岐の設定
        <input node="pre_sel_left" tag="left" />
        <input node="pre_sel_right" tag="right" />
```

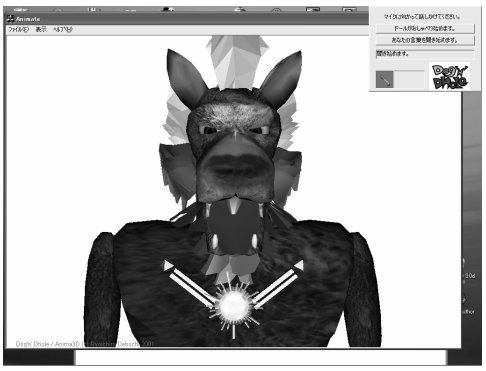
```
            </Select>
</StoryNode>
<StoryNode name="pre_sel_left" >  //selectパラメーターのセット
            <type value="DIRECT" />
            <duration value="0.1" />
            <set value="select=left" />
            <link value="thanks" />
</StoryNode>
<StoryNode name="pre_sel_right" >   //selectパラメーターのセット
── "pre_sel_left"ノードと同様──
</StoryNode>
//「ありがとう」と言った後、条件分岐させる
<StoryNode name="thanks" >
            <Chat>ありがとう<emotion morph="smile" />×</Chat>
            <type value="SWITCH" />
            <duration value="2" />
            <Case>  //パラメーター条件分岐の設定
            <switch node="sel_right" script="select=right" />
            <switch node="sel_left" script="select=left" />
            </Case>
</StoryNode>
<StoryNode name="sel_left" >
            <Chat>左の絵ですね<motion action="?"/>×</Chat>
            <type value="DIRECT" />
            <duration value="2" />
            <eventOut value="left.gif"/>  //左の絵の表示
            <link value="hereyou" />
</StoryNode>
<StoryNode name="sel_right" >
            ──"sel_left"ノードと同様──
</StoryNode>
//入力が例外の場合、「ごめんね。」としゃべらせる
<StoryNode name="dontknow" />
            <Chat><emotion morph="frawn"/>ごめんね<motion
action="no"/>×</Chat>
            <type  value="DIRECT" />
            <duration value="2" />
            <link value="order" /> //ユーザーからの入力待ちに戻る
</StoryNode>
<StoryNode name="hereyou" >
            <Chat><motion action="yes"/>はいどうぞ</Chat>
            <type value="DIRECT" />
            <duration value="1.5" />
            <link value="order" />
</StoryNode>
</StoryGraph>
```



（図４）サンプルStoryGraphのフローチャート

ユーザーからのキー入力はキーワードマッチング等のルーチンを経てタグ化され、<StoryNode>に受け渡される。<eventOut>タグでの出力はHTML内に記述されたJavaScriptに受け渡され、指定ファイル画像を表示する。

５．SignGraphの応用例



（図５）

音声認識して会話するバーチャルキャラクター　「Dog'n' Dhole」 2000-2001　デジタルイメージ展 2001 ワシントン銀座ギャラリー出品他（図５）

形態：Windows スタンドアローンアプリケーション

表示：Microsoft Visual C++ 6.0 と OpenGLで開発したリアルタイム 3Dコンピューターグラフィックス

音声認識：AmiVoice SDK version 3.0 (July.24, 2000 released) / Advanced Media, Inc.

音声出力：声優により録音されたWAVEファイル再生

このアプリケーション/コンテンツはＰＣで再生され、観客によるマイクロフォン入力音声を音声認識して反応し、リアルタイムコンピューターグラフィックス表示される擬人化エージェントが録音音声による「会話」を行ったり、背景色やライティング位置といったグラフィックスの変更を行うものである。（図１）

まず、エージェントが簡単な問いかけを発し、それへの返答により、あらかじめ用意された回答から選択して答えを返す、というフローが、SignGraph 1.0により記述プログラミングされている。

このアプリケーションで使用したアドバンストメディア製の音声認識エンジンは、Speech Recognition Grammar Specification (SRGS)として、Java Speech Grammar Format (JSGF) を採用していた。直接的にはJSGFからの出力がSignGraphへのユーザー入力となる。

（図６）

Bot3D Engineデモコンテンツサイト　「AI(アイ)とAL(アル)の診察室」 2002-2003 [8] 経済産業省　平成１４年度「次世代コンテンツ支援事業」/ 株式会社アトム（図６）

形態：ウェブサイト

表示：Microsoft Internet Explore 用ActiveX コントロールとしてOpenGLで開発された、リアルタイム 3Dコンピューターグラフィックス

入力：マウスボタン選択、または、キーボードによる日本語テキスト入力

音声出力：声優により録音されたWAVEファイル再生

このアプリケーションはウェブサイトとして閲覧される心理テストコンテンツである。「トーキングヘッド」擬人化エージェントが心理テストのナビゲーターとなり「会話」する。

閲覧者はエージェントの問いに対する２～３の回答からマウスボタンクリックまたは、キーボード入力で答えを選択する。SignGraph 2.0により記述されたフローを辿り、最後に診断結果がブラウザ内に表示される。入力テキストを解釈するSRGSにあたるものは独自仕様のフォーマットを使い、その解析ルーチンがアプリケーション内に組み込まれている。

６．終わりに

今後の課題として、SignGraph 3.0の解析エンジンの実装開発がある。特に録音音声ファイルを出力音声として使用する場合は、特殊な音声処理のルーチンを必要とする。

<Chat>タグでは日本語テキストの任意位置に感情/動作タグを挿入する設計であるのだが、リップシンクアニメーション、音声出力、感情/動作タグを同期再生させるためには、次の処理が必要となるはずである。

1)音声ファイルの音声処理を行い、音素と各音素期間を取り出す。
2)日本語テキストの形態素解析を行い、最終的にはモーラに分解する。
3)テキスト列との比較により、感情/動作タグのモーラ間の位置を決定する。
4)音声ファイル解析で得られた音素列と日本語テキスト解析で得られたモーラ列を比較し、マッチングを行う。
5)音声ファイル解析の音素列での感情/動作タグ位置を決定する。

リップシンクアニメーションは、音声ファイル解析での音素列に口形のモデル（モーフターゲット）を割り当て、各音素期間で形状補間することで実装している。同時に、感情/動作タグの実行を開始するトリガータイミングも決定できるので、それを元に感情表現のためのターゲットモーフアニメーション、あるいは、動作アニメーションのためのボーンモーフアニメーションを実行する。

また、SignGraphで使用できるスクリプト言語は定義上は任意のものを使用してよいとしているが、現在開発済みのSignGraph解析エンジンは変数代入、比較といった単純なプログラミング機能のみ対応している。たとえば、JavaScript等、本格的なプログラミング言語での具体的な実装を必要としている。

文献、参考リンク、及び注釈

[1] Voice Extensible Markup Language (VoiceXML) Version 2.0 W3C Recommendation 16 March 2004
http://www.w3.org/TR/2004/REC-voicexml20-20040316/
[2] "talking heads"という英語は、本来は「代弁者」、「報道官」といった意味である。
[3] Max Headroom 80年代半ばに英国、「チャンネル４」で作られたＴＶ番組シリーズ。バーチャルキャラクターは実際にはヘビーにメークされたMatt Frewerにより演じられた。
[4] QEDSoft　http://www.qedsoft.com/
[5] TVML　http://www.nhk.or.jp/strl/tvml/index.html
[6] Microsoft(R) Agent
http://www.microsoft.com/msagent/default.asp
[7] 「擬人化音声対話エージェント基本ソフトウェアの開発」
http://hil.t.u-tokyo.ac.jp/~galatea/bib/Sagayama2003IPA04paper.pdf
[8] Bot3D Engine　http://www.atom.co.jp/bot/