データベースシステム 121−21
情 報 学 基 礎 58−21
（2000. 5. 26）

構造化文書に基づいた対話的戯曲提示システムの実現

藤野 猛士 † 野宮一生 †
横田一正 ‡ 國島丈生 ‡ 三宅忠明 ‡

† 岡山県立大学大学院 情報系工学研究科
‡ 岡山県立大学 情報工学部

舞台での上演を想定した文学である戯曲を読む場合、その内容を視聴覚的な情報として受け取ることが出来ればより一層理解を深めることが期待出来る。本稿では、そのための手段として戯曲提示システムを提案する。戯曲提示システムは、次のような要素によって構成されている。1) 戯曲のテキスト文書をXMLの記述に基づいて構造化し、ト書きや台詞といった情報を区別する。2) ト書きエディタにより、構造化文書に視聴覚的な情報を付加する。3) 各種メディアの同期をとるため、構造化された戯曲をQUIKプログラムに変換する。4) システムのインタプリタにより、記述された実行順序によって各メディアを再生し、戯曲の上演を行う。5) 演出のための情報付加を行うため、デバッグ機能を持たせる。

本システムの特徴は、XML及びQUIKの記述に基づいたマルチメディア情報の対話的な提示を行うことである。

Implementation of Interactive Drama Presentation System Based on Structured Documents

Takeshi Fujino † Issei Nomiya †
Kazumasa Yokota ‡ Takeo Kunishima ‡ Tadaaki Miyake ‡

† Okayama Prefectural University, Graduate School of Systems Engineering
‡ Okayama Prefectural University, Faculty of Information Science and System Engineering

As dramas are written with assumptions to be played on stages, it is expected that we could understand them more deeply by directing freely them on virtual stages. For the purpose, in this paper, we propose an interactive drama presentation system, which consists of the following steps: 1)Dramas in the format of plain text are automatically transformed into structured documents in XML, where stage directions and speeches are separated. 2)By a direction editor, visual and auditorial information is manually inderted. 3)A structured drama is converted into a QUIK program, which synchronize multimedia informaton. 4)By interpreting the QUIK program, multimedia information for the drama are presented. 5)In debbugging environment, we can direct the stage.

One of the major features of our system is interactive presentation of multimedia information, based on XML and QUIK.

1 はじめに

計算機技術の発展と多方面での利用により、多種多量の情報が計算機に蓄積され利用されるようになってきた。これにより、ビジネスや科学技術分野のみならず、人文科学の分野もその対象となってきている。

岡山県立大学には「デァドラ伝説」という物語の文献が数多く所蔵されている。近年それらの資料の電子化が進められており、そのデータベース化も行われてきた。「デァドラ伝説」とは、ケルト（アイルランド）文学の国民的伝承物語である。ケルトの言葉（ゲール語）には文字が無かったため、物語を伝える手段として口承が用いられていた。そのため、「デァドラ伝説」は時代とともに内容や語彙が変化し、数々の変種が存在している。1860年以来発刊されたものが55種類、その他に口述表記されたものも数多くある。我々はこれまでデータベース化されたそれらの文書を対象にして、構造化、総索引表生成、検索などのプロトタイプシステムを実現し、複数の作品における類似性や語彙変化を分析・研究してきた[1],[2]。

また、「デァドラ伝説」を元にした戯曲も多数存在する。戯曲とは舞台での上演を想定した文学であり、文書内に広がる映像的なシーン描写が大きな意味を持つ。そのため、戯曲を単に文章として読んだだけではそれに含まれる情報の把握として不十分であると考えられる。そこで今回、分析対象を戯曲にまで拡張することを検討し、戯曲「悲しみのデァドラ」[3]の視聴覚的提示をQUIKシステム[4]の応用研究対象として取り上げることを提案する。本稿では、戯曲の視聴覚的提示方法についての設計・実装の結果の報告及び考察を行う。

2 システムの概要

2.1 対象とするメディア

これまでのQUIKでは、情報源として記号で記述したオブジェクトを対象としていた。しかし、様々な問合せへの対応を考えると、静止画や音声と言ったマルチメディア情報を対象として扱うことを想定するのは自然なことである。そこで、QUIKの拡張としてマルチメディア情報を扱うことを提案する。

今回QUIKで対象としているメディアは、以下の7種類である。

i) 静止画: gif, jpg, bmp などの画像ファイル
ii) 動画: mpg, mov, avi などの動画ファイル
iii) オブジェクト: 3D オブジェクト
iv) テキスト: 文字列
v) 移動情報: それぞれのメディアの移動情報
vi) 音声: wav, mid などの音声ファイル
vii) 発話: 文字列

上記のそれぞれのメディアは、各メディア特有の詳細データとして位置情報やファイル格納場所等のデータを、またメディア共通の共有データとして以下に記すようなデータを持っている。

- type
  メディアの種類
- 再生時間
  実際にメディアを表示する時間
- 同期時間
  処理するメディアが複数ある場合に、メディア間の同期をとるための時間

2.2 戯曲の構造化

戯曲の台本を単に電子化したテキスト文書では、ト書きや台詞といった情報を区別することが出来ない。また、本来戯曲の進行は文頭から逐次的に行われていくものであるが、仮に外部から進行を操作したい場合には、その場面毎を識別するための情報が新たに必要になる。

そこで、それぞれの情報を個別に取得するために、文書をXMLの記述に基づいて構造化する。

例えば、図1のような戯曲を考えてみる。

```
（ある日の朝。老女とラバーカム 登場。）
老女 ： まだお戻りになりませんか？
ラバーカム ： そのようね。
（老女 退場。）
```

図 1: テキストで書かれた戯曲

XMLに基づいたタグを構造識別子としてテキスト文書の各構成要素に挿入することで、文書に構文的概念を付加することが出来る(図2参照)。

```
<drama>
<stage>
ある日の朝
</stage>
<stage>
老女とラバーカム 登場
</stage>
<speech_part>
<speech role="老女">まだお戻りになりませんか？</speech>
<speech role="ラバーカム">そのようね</speech>
</speech_part>
<stage>
老女 退場
</stage>
</drama>
```

図 2: XML の記述に基づいて構造化された戯曲

そして、構造化された戯曲のト書きを具体的な画像や音声に置き換え (図 3 参照)、

```
<drama>
<stage>
<background src="http:background">
<role appear="http:老女, http:ラバーカム"/>
</stage>
<speech_part>
<speech role="老女">まだお戻りになりませんか？</speech>
<speech role="ラバーカム">そのようね</speech>
</speech_part>
<stage>
<role disappear="老女"/>
</stage>
</drama>
```

図 3: ト書きを編集した戯曲

各メディアに詳細データと共有データを付加することによって、図 4 のような QUIK プログラムに変換することができる。

```
 1: #main <= #stream(#scene);;
 2: #scene1 <= #outM( #background; #speaker1,
 3:                   #speaker2; #sentence1;
 4:                   #sentence2 );
 5:            #delM( #speaker1 );;
 6: #scene2 <= ··· ;;
 7:         ···
 8: #background/[ type=i, value=http://...../背景.jpg ];;
 9: #speaker1/[ type=i, value=http://...../老女.jpg];;
10: #speaker2/[ type=i, value=http://...../ラバーカム.jpg];;
11: #sentence1/[ type=4, font=老女,
12:              value=まだお戻りになりませんか];;
13: #sentence2/[ type=4, font=ラバーカム,
14:              value=そのようね];;
```

図 4: QUIK プログラム

2.3 戯曲提示システム

ここでは、QUIK でマルチメディア情報を扱うための拡張機能の応用例として、構造化された戯曲の視聴覚的提示を行うシステム [5],[6] について簡潔に説明を行う。図 5 に戯曲提示システムの概要を示す。

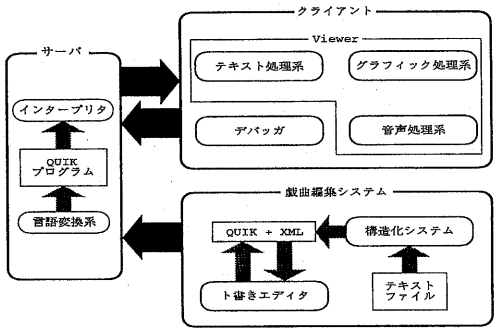

図 5: 戯曲提示システムの概念図

2.3.1 戯曲編集システム

戯曲編集システムは、以下の 2 つの処理系から構成される。

i) 構造化システム
テキストで書かれた戯曲を XML の記述に基づいて構造化を行なう (図 1 →図 2)。

ii) ト書きエディタ
構造化された戯曲のト書きを具体的なメディアに対応させる (図 2 →図 3)。

2.3.2 サーバ

サーバは、以下の 2 つの処理系から構成されている。

i) 言語変換系
構造化された戯曲を各メディアの同期をとるために QUIK プログラムに変換しシナリオを生成する (図 3 →図 4)。

ii) インタープリタ (QUIK)
記述された実行順序によって、各メディアをクライアント側に送信するエンジンとして QUIK を用いる。

2.3.3 クライアント

戯曲提示システムにおけるクライアントは以下に示す一連の動作を繰り返すことによって、劇を進行する。

- メディア情報の取得と解析

- イベントの実行
- QUIK へ次のメディア情報の送信要求

そして、劇の進行の途中でイベントの変更要求が生じた場合、劇の進行を停止してメディア情報の変更を行う。その際作成した新規メディア情報は、保存することによって元のメディア情報の代わりに QUIK へ返信され、イベントの変更が行われる。

クライアントは、以下の 4 つの処理系から構成されている。

i) グラフィック処理系
3D 空間上へのグラフィックの描画を行う。

ii) 音声処理系
発話及び音声再生を行う。

iii) テキスト処理系
字幕の表示を行う。

iv) デバッガ
マルチメディア情報を付加した構造化文書への更なる情報の追加及び編集を行う。

またクライアント側からサーバ側に要求される機能として、以下の 6 つを考えている。

- シーングラフ作成
シーングラフを作成するためのデータをサーバに要求する命令。
- 再生
シナリオを開始する。
- 一時停止
シナリオの実行を一時停止させる。
- 再開
一時停止 によって停止された場所から処理を再開させる。
- 停止
シナリオを完全に終了させる。
- 保存
編集した情報をサーバに保存する。

2.3.4 デバッガ

一般に戯曲文書自身には劇の大筋の事柄が書かれてあるのみで、詳細な事柄に関しては記述されていない。例えば、「人物 A および人物 B 登場」という場面の場合、これでは登場者が人物 A と人物 B であることが分かるのみでその登場方法に関しての情報は何も含まれていない。そのため、

- どちらが先に登場したのか。
- 登場のタイミングはどうなのか。
- どこへ向かって登場したのか。

といった幾つかの情報の欠落が生じる。実際の劇を形成する上でも役者や演出家による独自の付加情報が必要不可欠であることを考慮すると、本システムにおいてもユーザが何らかの手を加えることによって劇を作成していく必要があると考えられる。デバッガは、このような欠落している情報の付加を行うための処理系である。

デバッガは次の 3 つのウィンドウによって構成されている。

i) シーングラフウインドウ
シナリオの実行順序をグラフィックで表示したものである。図 6 にシーングラフウインドウの概念図を示す (図 4 の QUIK プログラムに対応)。ノード (図 6 中のマルで囲まれた部分) の中は、各メディアの名前を示しており、破線で表現されたノードは、現在再生しているノードを表している。

ii) 詳細情報ウインドウ
シーングラフウインドウで示された現在実行しているノードの詳細情報を表示する。ここに表示されるデータを編集することによって、詳細情報を決定していく。

iii) 位置表示ウインドウ
通常、ビューアは 3 次元仮想空間からなる舞台を正面から見ているため、キャラクタなどの位置が分かりにくい。そこでこの位置表示ウインドウに、舞台を真上から見たキャラクタの位置を示すことにする。移動情報の入力の際にも、このウインドウを使用する。

また、デバッガには「再生」の機能があり、シーングラフウインドウの任意のノードからシナリオを開始することが可能である。

3 戯曲の視覚的提示

戯曲を視覚的提示にあたり、最初のアプローチとして挙げられるのは舞台を構築することである。今

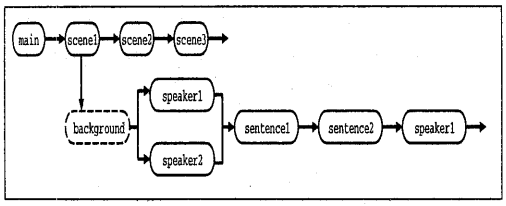

図 6: シーングラフウインドウの概念図

回対象とする戯曲「悲しみのデァドラ」は3つの幕から構成されており、それぞれ舞台となる場所が異なっているため少なくとも3つの舞台を構築する必要がある。

そこで今回、それらの舞台を3次元空間上に構築することを検討した。これによって、例えば「森の中に家があり、家の後ろには山がある。玄関から入っていくと正面には机があり、右の壁には戸が、左の壁には窓がある。」といった舞台の場合、劇の進行に関わらず視点を「玄関に立って正面を見る視点」や「家の外に出て窓から中を覗く視点」等に自由に変更することが可能になる。実際の舞台では、観客は常に一定の視点から劇を観ることしか出来ないため、これは大きな利点となる。

戯曲の構成要素としては、舞台の他に役者や大道具、小道具といったものが挙げられる。これらのオブジェクトも3次元オブジェクトとして作成し、3次元空間上の舞台で出力及び消去といった操作を行うことで戯曲の上演の実現を図る。また戯曲の演出を行う上で、動画や静止画の再生、スポットライトの操作、作成したオブジェクトの移動制御、台詞と同期をとった字幕の表示等も必要になる。

4 戯曲の聴覚的提示

4.1 音声合成による発話処理

戯曲の聴覚的提示を実現するにあたり最も重要となるものは、台詞を必要に応じて発話させるということである。そのための手段としては、あらかじめ台詞を音声データとして録音しておいたものを再生させるといった方法や、文字列として与えられた台詞を音声合成によって発話させるといった方法が考えられる。

前者の方法では、人の声をほとんどそのまま再生できるため本物の演劇に近い臨場感を出すことが期待できるが、対象となるもの（ここでは戯曲「悲しみのデァドラ」）が変わる度に台詞を録音しなければならず、汎用性に欠ける。また、台詞を録音したデータファイル一つ一つの容量が大きく、その数も膨大になるためそれらを管理するのも大変である。

一方、後者の方法では、コンピュータによる音声合成という特性上どうしても機械的な喋りかたになってしまうという欠点はあるが、基本的には必要に応じて台詞を文字列として与えてやるだけで良いので、汎用性の面では優れていると言える。

そこで今回、コンピュータによる音声合成を利用して発話処理を行った。

4.2 場面に応じた効果音の再生

戯曲のト書きの中には、効果音を伴う動作を指定している場合がある。また直接指定されていなくても、人物が歩く場面や、ドアを開けるといった場面には、その動作に適した効果音を再生した方が、視覚的な表現だけをする場合よりも臨場感を与えることが出来る。そこで、こうした発話以外の音声全般の再生も行うことにした。

戯曲で使われる効果音は、再生されるべき位置の固定されるものと位置が変化しつつ再生されるもの、また位置を特定せずに再生されるものの3種類に大別することが出来る。これら位置情報と、再生すべき音声ファイル及びその音量の3つの情報を、音響処理を行う際の入力情報として扱うことにした。

しかし、ト書きの中から効果音が必要な部分だけを自動的に抜き出すのは困難であり、またその場面に適した効果音を自動的に選択するのも同様に困難である。よって構造化文書がQUIKプログラムに変換された段階では、これらの情報は付加されていない。そのため、これらの情報はデバッガ上にて手動で付加する必要がある。

5 実装と評価

2.1で述べたメディア情報は、特殊なクラスに全て保持することにした。戯曲提示システムのサーバは、そのクラスを必要に応じてクライアントに送信する。

5.1 グラフィック処理系

ここでは静止画、動画、オブジェクト、移動情報を処理する。それぞれのメディアに含まれるデータを入力として受け取り、ビューア上に3次元空間を構築する。ビューアの実装には、開発言語としてJava3Dを使用した。今回グラフィック処理系には、以下に記すように劇閲覧用とデバッグ用の2つのビューアを用意した。

5.1.1 劇閲覧ビューア

3次元グラフィックスで作成された劇を閲覧するためのビューアであり、図7にそれを示す。これにより、ユーザは閲覧するだけではなく、仮想世界を自由に移動することができる。

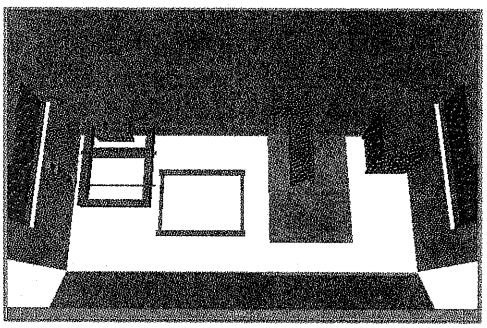

図 7: 劇閲覧ビューア

Java3Dでは仮想世界の物体間の構造を木構造で表現しているため、今回の劇の各場面設定の構造もそれで表わすことができる。ここでこの木構造に大量の物体を加えてしまうと、計算処理が膨大になってしまい、実行速度が低下したりメモリ不足で表示できなくなってしまう。そこで仮想世界内にユーザの位置を感知するセンサーを設け、ユーザの位置により必要な物体を木構造から切断したり連結したりすることにより、計算処理の軽減を図る。

今回劇とはいえ、舞台セットを作成するのではなくその世界全体を広範囲に作成している。例えば第一幕ではラヴァーカムの家が舞台であり、脚本中では外へ出ることはないが、今回作成した仮想世界においてはユーザが自由に移動できるよう家の外の情景に関しても作成している。しかし外へ出た場合、家の中の物体は見えないため表示しなくてもよい。そこで上述したようにその場において必要ないものに関しては仮想世界から排除している。これによりスムーズな動作と多数の物体の配置が可能となる。

5.1.2 デバッグビューア

デバッグビューアは、ユーザが劇中のイベントを変更する際に劇を停止することによって現れる。作成したデバッグビューアを図8に示す。

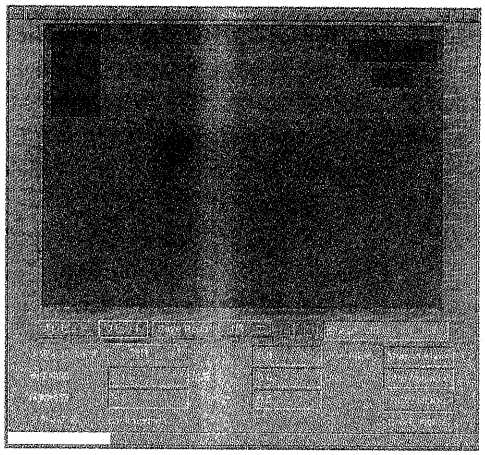

図 8: デバッグビューア

デバッグビューアは次の2つの部分からなる。

- 移動経路記述用地図
  上部のキャンバスがこれに当り、仮想世界における物体の位置を表した地図の役割をする。地図の拡大、縮小も選択ボタンを押すことにより可能である。劇の進行上登場した物体と画像は円状の目印で、その位置を表示する。これにより役者の位置関係が把握できる。また、この目印は通常赤色で表示されているが、それぞれの役者がイベントを行うときには青色に変化する。これにより現在どの役者が演技を行っているのかが分かる。また、物体の移動情報を作成する際には以下の手順で行う。

  i) キャンバス内でマウスドラッグをし、移動経路を描く。

  ii) “Preview” ボタンを押し、動作確認をする。

iii) “Road Save” ボタンを押し、移動経路を保存する。

- メディア情報ビューア
  ここで取得したメディア情報の各種のパラメータを表示し、変更を行う。移動経路以外のデバッグはこれらの値を変化させることによって行うことができる。

デバッグ後はメディア情報を更新するために”Save Node” ボタンを押す。ここで、変更するメディア情報がオブジェクトまたは画像である場合、仮想世界に出現させる物体や画像が変更されている可能性がある。その場合、図 9 に示すようにユーザが指定した物体や画像を表示することにより再確認してもらう。更新されたメディア情報は劇を再開する際にサーバへ返送され、再度そのイベントから実行される。

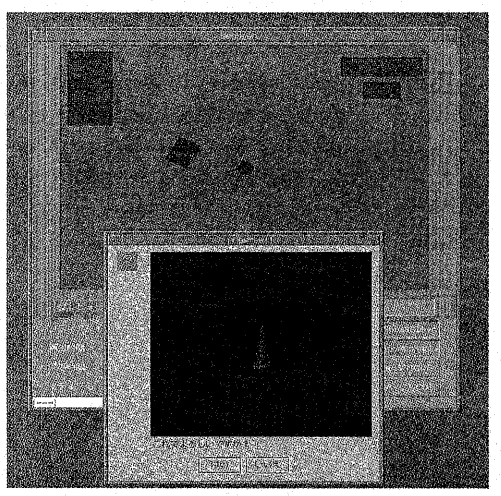

図 9: 物体選択画面

5.2 音声処理系

戯曲提示システムの処理系の一部である音声処理系には、発話メディアを扱う発話処理系と、その他の音声に関するメディアを扱う音響処理系の 2 つの処理系を用意した。

5.2.1 発話処理系

発話処理系を実装するにあたり、IBM ViaVoice SDK1.5、MicroSoft SpeechAPI 及び音声合成ソフトウェア IBM ProTALKER97 を利用した。開発言語には Visual C++を用いた。また、戯曲提示システムが開発言語に Java を用いて実装されているため、システムと発話処理系の仲介となる部分を Java を用いて実装した。

発話メディアは詳細データとして、発話すべき文字列（台詞）、声質、発話の速度、声の高さの 4 種類の情報を持っている。

これらのデータを入力として受け取り、Java プログラムを介して一度ファイルに出力する。台詞情報は read.txt というファイルに、声質情報は data.dat というファイルに、速度情報は speed.dat というファイルにそれぞれ出力される。これらのファイルを発話処理系が入力として受け取り、それに応じて ProTALKER97 の音声合成エンジンを制御することによって、コンピュータのスピーカから音声が発話される。

また、発話メディア内のデータは、デバッガ上で修正することも出来る。そのとき、声質は男性・女性のどちらかを選択、速度は 5 段階の中から選択する。

5.2.2 音響処理系

効果音再生のための音響処理系の実装には、Java3D を使用した。

Java3D は Java,Java2 に比べるとサウンド関連の機能が充実しており、音声データの再生・停止だけでなくループ回数の設定や再生順のランク付け、3D 空間上の距離による音量の変化等も制御することが出来るといった特徴がある。こうした点を考慮して、Java3D を使用するという経緯に至った。

音声メディアは詳細データとして、再生すべき音声ファイルの格納場所（URL）、音量の 2 種類の情報を持っているのだがこれらの情報は、まずデバッガ上にて手動で指定する必要があるため、音響処理系もデバッガ用と上演用の 2 つを用意することになった。

デバッガ用の処理系では、再生すべき音声データファイルの格納場所及び音量の指定し、これらの情報の QUIK プログラムへの付加を行う。そのため

に、音声データファイルを選択するためのリストと音量調節のためのスクロールバー、また音声を確認するための再生・停止ボタンの3つの外部インターフェースを設けた。その際指定できる音声ファイルはあらかじめ用意しておき、特定のフォルダにまとめて格納しておく必要がある。

上演用の処理系では、システムのサーバから送られてきた情報を入力として受け取り、スピーカを通して効果音の再生を行う。

6 おわりに

本稿では、QUIKでマルチメディア情報を扱うための拡張機能の応用例である構造化された戯曲の視聴覚的提示システムの概要を説明し、そのクライアント上にあるグラフィック及び音声発話に関する処理系の設計と実装についての報告を行った。

戯曲の視聴覚的提示によって、戯曲自体からは十分に読み取れなかった視覚的および音響的な空間を提示することができ、新たな発見をすることが期待される。現実の戯曲の上演がそうであるように、たとえ同じ戯曲であったとしても、それは演出によって様々異なったものになる。本システムでは、ト書きエディタとデバッガがその演出に対応しており、利用者がコンピュータと対話しながら試行錯誤や思考実験を行い、様々な上演結果を出力することができる。本システムを、戯曲の演出システムとして使用する場合、

- テキストの変更
- ト書きエディタ
- デバッガ
- 仮説付問い合わせ (仮説推論)

の4つが関連している。

今後の課題としては、以下に記すような機能を実装していくことが挙げられる。

- 汎用的な構造化アルゴリズムの考案
- 構造化文書からの情報抽出の自動化
- コンテンツの整備
- デバッグ機能の強化
  - ノードの追加及び連結
  - 同期時間の指定方法の改善
  - 仮説による演出機能の実験環境の実現
- オーサリングツールの作成
- 音声再生時における音源の位置制御

今後は、上記問題点の解決のみならず、演出システムとしての統合の実現を図る予定である。さらに、文学や演劇の専門家による評価を反映させていきたいと考えている。

謝辞

最後に、日頃から有意義な御助言を頂いた言語・ソフトウェア工学研究室の諸氏ならびに岡山理科大学の劉渤江助教授、大森徹氏に厚く御礼申し上げます。

参考文献

[1] Takeo Kunishima, Kazumasa Yokota, Bojiang Liu, and Tadaaki Miyake, "Towards Integrated Management of Heterogeneous Documents", *Cooperative Databases and Applications '99*, pp.39-51, Springer, Sep., 1999

[2] 三宅忠明、横田一正、"情報処理としての作品解析 — デァドラ伝説の研究から," 英語青年、vol.146,no.1, pp.6-9, April., 2000.

[3] 三宅忠明: "J.M. シング/悲しみのデァドラ" , 大学教育出版,(1999)

[4] BojiangLiu,KazumasaYokota,NobutakaOgata "Specific features of the QUIK mediator system" IEICE Transactions on Information and Systems,vol.82,no.1,January 1999

[5] 緒方啓孝, 野宮一生, 國島丈生, 横田一正 "QUIKにおける戯曲の視覚化" 電気・情報関連学会中国支部大会,122512,October 1999, 広島

[6] 杉本健二、緒方啓孝、的野晃整、平野尚孝、横田一正、國島丈生、"対話的マルチメディア情報提示システム実現のための QUIK の拡張"、情報処理学会データベースシステム研究会、東京、2000年5月25-26日.