CALL教材の自主開発のために[1)]

キーワード：CALL，教材開発，ドイツ語教育

岩崎克己
広島大学外国語教育研究センター

1．はじめに

平成9年4月1日，教養的教育改革のための学内共同教育研究施設として広島大学に外国語教育研究センターが発足した。センターは，この1年間に，学生の自学自習のためのマルチメディア外国語自習室を開設[2)]するとともに，「CALL教材開発運用プロジェクト」を通じて教材の自主開発に取り組んできた[3)]。本稿では，CALL教材の開発に携わってきたこの1年間の経験にふまえ，自作可能な教材のタイプを実例をあげて簡単に紹介するとともに，CALLのいっそうの普及にとって不可欠な自主教材の作成を支援するための枠組みについて論じる。本稿の構成は以下のようになる。

2．自作可能なCALL教材のさまざまなタイプ

CALL教材の特徴とその有用性については，すでに多くの論者が指摘しており，ここで改めて論じる必要はないであろう。それは，大雑把に言って「インタラクティブ」と「マルチメディア」という2つのキーワードに集約することができる。すなわち，1）ある種のゲーム性の組み込みにより個人学習においても擬似的なコミュニケーション状況をつくれること，2）音声・

画像・テクストなどを組み合わせた情報のやりとりを柔軟に行えることの2点である。以下では，自作可能なCALL教材のタイプにはどのようなものがあるかという観点から，実例を挙げつつその代表的なタイプを類型化してみる[4]。なお，これらの教材の典型的なスクリプト例については，付録として別に挙げることにする。

2．1．伝統的なドリルを教材化したもの

ひとくちにドリル型教材といっても学習の目標や課題(task)を問題にすれば，さまざまな性質のものがある[5]。それらについては，註で示した文献を参照してもらうことにして，ここではそうしたさまざまなドリルが技術的にはどのような形式において実現されるかに絞って考えたい。学習者がコンピュータ上で解答する際に行う操作の違いという観点から見ると，ほとんどのドリル問題は，その学習目標や課題の如何にかかわらず，以下のパターンに分類できる。

1）解答欄や空欄に答えを書かせる書き込み問題
2）マウスのポインタで正解を選んでクリックする四択問題[6]
3）複数の選択肢を並べ，解答欄までマウスで引きずって答えさせる複数解答選択問題
4）マウスを使った並べ替え問題
5）マウスでお互いにマッチするアイテムを結びつけるペアリング問題

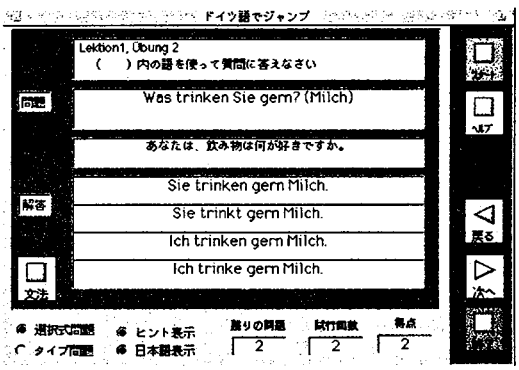
こうしたドリル型教材に対しては，「紙の上のドリルをコンピュータ上に移しただけでほとんど意味が無い」という批判もある。しかし，以下の理由からその有用性は非常に高い。

1）その場で自己採点できるため，学習効率があがる
2）到達度をその都度自己確認できるので，学習の動機付けにとって好ましい
3）ランダムに出題したり一度間違えた問題をもう一度繰り返させたりする機能を組み込めば，繰り返し練習が無理なくできる
4）さまざまな問題形式や解答あるいは回答形式が指定できるので，難易度や養成しようとする能力を限定したいろいろなタイプの問題がつくれる
5）必要な情報を学習者の要求に応じて随時提供できるので，自習の際の学習レベルの自己調整が可能である
6）ドリルに直接音声をつけられる[7]
7）学習者がコンピュータを使って行った操作を記録する機能を組み込むことによって，学習のパターンや誤りの傾向などを分析する際の一次資料となる学習過程の動態的な記録を取ることができる[8]

また自作する際も，HyperCard, SuperCard そして Oracle Media Objects 等に代表されるカード型オーサリングソフトの場合は，ドリル型教材をプログラム化するための基本的なスクリプトのパターンはほとんど確定しておりプロトタイプを作りやすい。上述したパターンのドリル型教材を作成する際に必要となる技法の基本を列挙すると，以下のようになる。個々の技法の一つ一つの具体的なスクリプト例に関心のある読者は，本稿の付録(1)-(7)をそれぞれ参照されたい。

1）乱数による問題のランダムな呼び出し
2）乱数による選択肢のランダムな配列〔四択問題〕
3）間違った問題の繰り返し
4）不要な空白やピリオドなどを取り除く解答の整形〔書き込み問題〕
5）正解が複数ある場合の処理
6）誤答に対応するヒントの登録
7）マウスで動かす答がマウスから手をはなした時点で指定された領域内（典型的には解答欄内）にあるかどうかの判断〔複数解答選択問題やペアリング問題など〕

図1



なお，図1は，筆者が同僚と2人で作成した初級ドイツ語教科書「ドイツ語でジャンプ」（白水社刊）付属の文法ドリル（四択問題モード）である。ここではたとえば上に挙げた技法の2）－6）が使われている。

2．2．画像と音声を直接結びつけることで語彙習得を目指すもの

市販の教材で言えば Rosetta Stone がこの範疇にはいる。方法論的にはGraded Direct Method[9]の延長にあるが，以下に挙げる3つの点で新しく，初級段階での音声を伴った語彙の習得や定着のための教材として実用性が高い。

1）学習の目標を主に語彙の習得に絞った
2）かつて本とせいぜいテープのみであった媒体がコンピュータ上では音声と画像を直接結びつけられるようになった
3）同時に複数の画像を出しその中から正しいものを選ぶという問題の形式によって，インフォメーションギャップ（情報の格差），応答の際の選択可能性，フィードバックというコミュニケーションを成り立たせる基本原理[10]を組み込んだ

図2

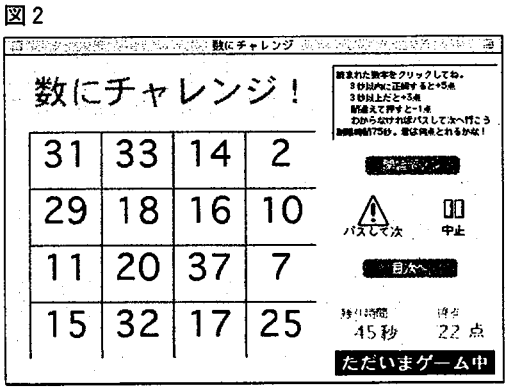


図2は，筆者が自作したゲーム形式の数字学習ソフトの例だが，読み上げられた数字を方陣の中から制限時間内にいくつ当てられるかを競うものである。数字の代わりにものの絵やつづりを使えば，同じ形式で語彙学習ソフトを作ることができる。このタイプの教材は，作るにあたって選択肢の個数や問題の形式にいくつかのバリエーションが考えられる。選択肢は，方陣に収まる数が作りやすく，ポータブルコンピュータのモニターのサイズ[11]等を考慮すると，事実上4・9・16あたりまでが限界である[12]。新しく語彙を学習する場合は，

1度に表示される数はできるだけ少ない方が学習者の負担が少ないが，逆に数が増えた方が，正解を探す際の選択肢の幅が広がるので，そのゲーム的な性格が強まる[13]。また出題の形式に関しても，すべて正解すると次に移るような単調なものから，正解したものを次々に新しいものへと差し替えていく，よりゲーム性の高いものまで考えられる。このタイプも基本的なアルゴリズムが確定しており，音声と画像の素材さえあれば個々の教師が必要とする内容に差し替えられる汎用フォーマットを作るのは比較的容易である。技法の基本は以下の2つである。具体的なスクリプト例については本稿の付録(8)-(9)をそれぞれ参照されたい。

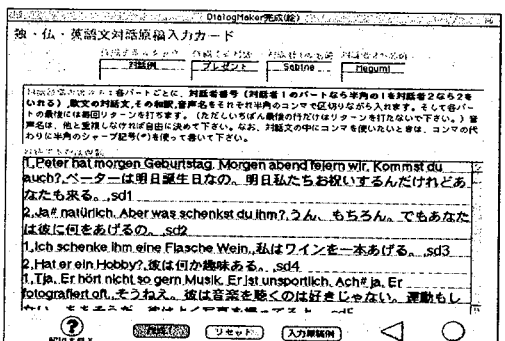
1）あらかじめ方陣として決められたスロットへ，乱数を使って選んだ画像やテクストをセットしていく
2）流した音声とクリックした箇所のデータが一致すればそこにまた新しい画像やテクストを同じくランダムに選んでセットする

一般に，ゲーム的な性格を持たせた方が学習の動機付けにとって好ましく，その場合は，チャレンジした回数や時間に一定の制限を定めて終了させる処理も必要になる。またドイツ語のように文法的な性を持つ言語の名詞の学習の場合，文法的性の情報をたとえば定冠詞1格の形で音声に組み込むのであれば，名詞の前に冠詞の発音を流す処理も必要になる。ゲームの枠組みとして，あらかじめ語場を設定する（家族・身の回りのもの・色・数・乗り物・言語・スポーツ・趣味の動詞・学生生活）ことも考えられる。なお，このタイプの教材に対しては Graded Direct Method に対する批判がそのままあてはまる。学習できる語はイメージ化の容易な内容語（具体的な名詞・主に動作に関わる動詞・形状や様態を叙述する形容詞）とその位置関係（動作や場所に関わる前置詞）などに限られる。

2．3．映像・音声と対話テキストを伴う対話練習型教材

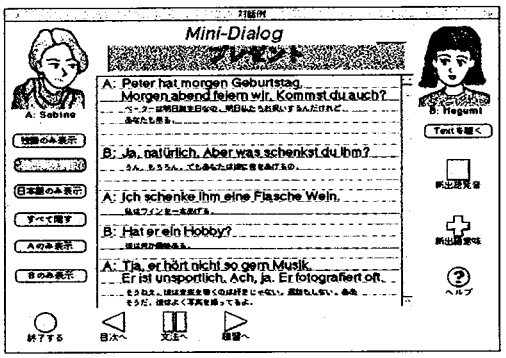
従来のカセットテープによる練習では，特に初級者の場合，音声教材がテキストのどの箇所を読んでいるのかの特定自体が難しく，繰り返し練習する場合も頭出しで苦労するため時間的ロスも大きい。また練習の形態もテープを単に何度も聴くにとどまることが多い。これに対しCALL教材の場合，たとえば映像や音声の流れているテキストの箇所を反転させたり，逆にテキストをクリックすればその箇所の音声や映像を流すという形で，映像や音声とテキストを直接結びつけることできるのでこうした問題は生じない。ただ難点は，対話練習型教材は作るのに手間がかかる点である。図3は，この問題を克服するために筆者が Oracle Media Objects を使って自作した欧文対話教材の自動作成ツール Dialog Maker の対話作成画面である。テキスト原稿とデジタル化された音声さえあれば，これを使うことで比較的簡単に対話練習型教材を作ることができる[14]。なお図4は，Dialog Maker によって実際に自動作成された対話教材の例である。ここでは，ボタン一つで，欧文

図3



の対話テキストと日本語訳のどちらか一方あるいはその両方を出したり隠したりできる。音声の方も，テキスト全体や対話者の一方のパートだけを，あるいはまた特定の行だけを流すこともできる。これらの機能を組み合わせれば一人でも，聞き取り，復唱，パートナー練習，ディクテーションなど多様な練習が可能である。

図4



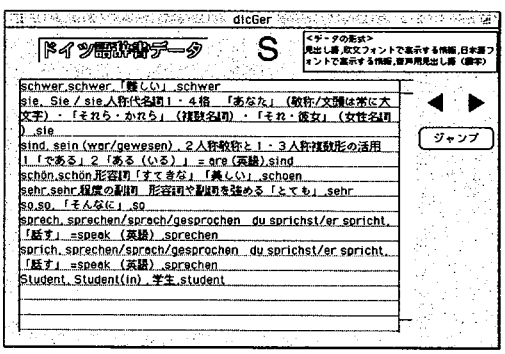
2．4．読解用プログラム

読解用プログラムは，大きく言って読みのテクニックに関わる部分と理解内容を問うための問題の部分に分かれる。問題の部分は，たとえば本文に合う内容を直接答えさせたり，選ばせたり，要約文の一部を穴埋めしたり，Yes-No 形式で内容の理解を問うたりなどいくつかのパターンが考えられる。ただし，これらはいずれも2.1で紹介したドリル型問題のバリエーションであり，同じ技法で処理できる。読みのテクニックに関わるものとしては，頻出語のチェック機能，国際共通語 (universally understood term/Internationalismen) やキーワードなどの手がかりになる語についての情報を与えたり，複雑な構文の基本骨格の部分だけを強調したりなどの機能が考えられる。これらは，あらかじめキーワードや基本構文とその処理法を登録しておけば簡単であるが，個々のデータに依存するので，作る手間がかかる。読みのテクニックに関わるもう一つの問題は，読む速さの制御である。これは，あらかじめ読む速さを指定して，それに応じてテキストを途中から消していったり，時間を制限して読ませ，速さを競わせたり測定したりすることなどがその典型である。これらのスクリプトについては本稿の付録の(10)-(11)を参照されたい。なお，このタイプのものに関してもすでにフリーソフトがいくつかある[15]。

2．5．glossary としての補助的辞書

本格的な辞書は，データの大きさや要求される記述の厳密さ・語彙選択の適切さなどからして個人が簡単に自作できる範囲を超えている。ここで問題にするのは，いくつかのプログラムで汎用に使うことのできる glossary（補助的な小規模辞書）ないしは単語帳のようなものである[16]。たとえば，2.5.や2.6.で紹介した，対話練習型教材や読解用プログラムには，こうした辞書の組み込みは必須である。具体的にはこれらの教材のテキストの中の単語部分をクリックするだけで辞書にアクセスし，意味・用法・音声などを返すようなものである。このために考えなければならないのは，辞書用のデータと辞書引きのための手続きの２つである。たとえば，図５は，2.3.の図４で紹介した対話練習型教材で使うことを目的にして作ったドイツ語簡易辞書のＳの項のデータ例である。辞書引きの手続きに関して言えば，欧文辞書の場合，技術的には以下の処理ができればよい。詳しくは本稿の付録

図5



の(12)を参照されたい。

1）クリックした単語を選択し反転させ，
2）選択した単語の頭文字のアルファベットを手がかりにして辞書スタックの中のその頭文字のページを探しに行き，
3）そのページの各行を検索し，第1アイテムの見出し語に一致するものがあれば第2アイテム以降に書かれた情報にアクセスし，
4）得られた情報を適切なフォント・サイズ・スタイルの辞書情報として表示しつつ，
5）登録された音声を流す。

したがって，英語文のように欧文特殊文字を含まないテキストであれば30行ほどのスクリプトで処理できる。しかし英語以外の言語を扱う場合にはいくつかの問題がある。

まず第1に，HyperCard や SuperCard や Oracle Media Objects などのオーサリングソフトの日本語版では，テキスト処理の際に日本語文字にあたるアスキーコードを含む文字列があると，そこから後ろを自動的に日本語とみなして2バイトで処理するとともに，自動的に単語の切れ目を付けてしまう。このためたとえばドイツ語のウムラウトのような欧文特殊文字を含む単語は，その特殊文字のアスキーコードが日本語文字のそれと一部重なるため，そこが単語の切れ目として認識されてしまう。したがって，特殊文字を越え単語の切れ目の空白に到って初めて単語全体を切り取るような特別な処理が必要になる。

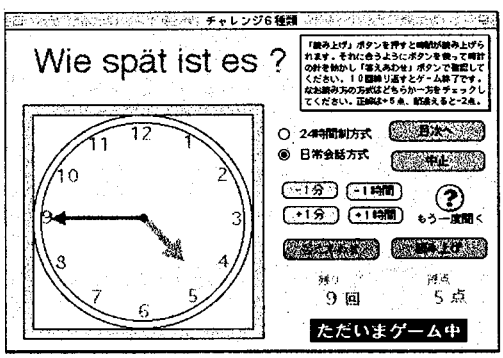
第2の問題は，音声用の見出し語である。複数のシステムで共有するので音声名と単語名を統一しておくのがわかりやすいが，音声名として特殊文字を使えないため，特殊文字を通常のアルファベットに翻字しなければならない（たとえば schön を schoen とするように）。この作業を自動化するのであれば自動翻字の部分も組み込む必要がある。

英語以外の欧文処理の3つ目の問題は，Windows と Macintosh の両方で使えるプログラムを作ろうとする場合，特殊文字のアスキーコードの割り当てがその両者で異なっているため，その変換を考える必要があることである。これは一方のシステムで作ったテクストデータを自動的に別のシステムに適合するテクストデータに変換するツールを作っておくと便利である。

2．6．Total Physical Response 型

言語的な手段による指示や理解内容を非言語的な応答で表す Total Physical Response 型の教材も作成可能である[17]。これらには，たとえば道案内の経路を表示したり，ものを移動させたり，時間についての情報を時計の針を動かすことで示したり，また財布からお金を出したり，家族関係を系図に書き込んだりなどさまざまなバリエーションがある[18]。適切な移動かどうかの判断は，あらかじめ移動先に範囲を指定し，マウスでつかんで動かす対象がマウスから手をはなした時点でその範囲内にあるかどうか

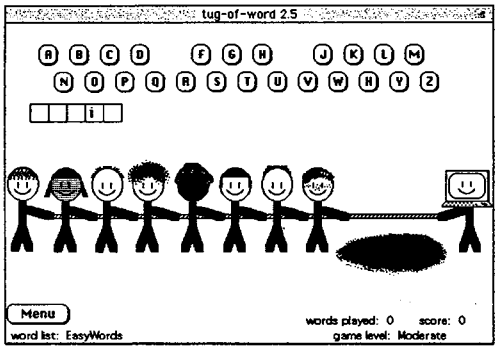
図6



の判断で処理できる。道案内は，経路そのものを問題にする場合は，アニメーションの道筋の記録を考えることになる。時計などの場合は，角度が問題となるので，三角関数を含む計算の部分がいるが，これについては，それ自体独立したモジュールとして組み込んで使えるスクリプトがすでにある[19]。なお，図6は筆者が自作したドイツ語の時間の言い方を学習するスタックである。24時間制と日常での時間の言い方が切り替えられるが，学習者は読み上げられた時間の言い方に合わせ実際に針を動かし，正しく理解したかどうかを自己チェックする。

2．7．その他のゲーム型教材

図7



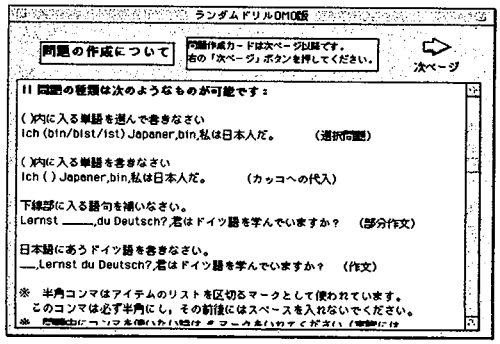
ゲーム型のCALL教材もその多くは，以上に述べてきたいくつかのタイプの組合わせやバリエーションが多い。たとえば，2．2．で述べた語彙習得用の教材も，画像を1種類にして隠し，簡単なドイツ語でヒントを与えながら文字をあてていくという方式にすれば謎々や推理ゲームになる。また画像を隠した状態で何の情報もなしにあてずっぽうであてることに重点を置くといわゆる伝統的なハングマンゲームになる。またヒントに重点を置けばクロスワードパズルにもなる。文字のコマを与え，創造性を問題にすれば，スクラブルなども可能である。スクラブルはかなりの語彙数を含む辞書が必要となるので，自作するのはやや難しいが，ハングマン程度であれば，基本的なアルゴリズムはそれほど難しくない。図7は，Jan Garner 氏がHyperCardで作成したハングマンのフリーソフトtug-of-word ver. 2.5 である。一定時間に答えられないとコンピュータと綱引きをしている子供たちがが一人ずつぬかるみに落ちていく仕掛けとなっている[20]。作りはシンプルだが，単語の部分が簡単に差し替えられるのでそのまま授業でも使うことができる

3．CALL教材の自主開発を支援するために

前節では，個人レベルで自作可能なCALL教材のいくつかのタイプについて挙げてみたが，ここではこうしたCALL教材作成のすそ野を広げ教材の開発そのものを支援していくために重要で，かつ現状でもすぐ実現可能な問題について簡単に触れたい。

3．1．汎用フォーマットの提供

図8



当たり前のことだがCALL推進のために重要なことの一つは，実際に個々の教官が必要とする内容の教材をいかに提供していくかということである。授業用にそろえたり学生に自習用として買わせたりするには市販の教材は高額すぎるし，特に英語以外の言語の場合は数自体も少なく，内容などを検討すると使えるものは数えるほどしかない[21]。したがって，必要な教材を教師自身が自作せざるを得ない。しかしこうした教材のゼロから

の作成をCALLに関心を持つすべての教師に要求する限り，CALLのすそ野は広がらない。またそうした教材作成を補助できる技術スタッフが身近にいることを前提にして議論しても始まらない。このことから，現に教材を作成している者にとって重要なことは，自分たちが使う個別の教材を作るにとどまるだけでなくデータの部分を差し替えることによって，誰でもがそれぞれに必要とする内容の教材を作ることができるような汎用性の高いフォーマットを幅広く提供していくことである。図8は筆者の同僚である吉田光演氏がHyperCardで作成し筆者がWindows上でも動かせるようOracle Media Objectsに移植したドリル型教材「ランダムドリル」の問題作成画面である。問題データと正解の部分を差し替えれば誰でも簡単にドリル型教材がつくれるようになっている。このほかにも前節で紹介したタイプの教材に関してはすでに基本的なパターンが確定しているので，こうした汎用フォーマットを作ることは現時点でも可能である[22)]。

3．2．初心者向け講習会の開催

多少とも複雑なプログラムをすべて自力で開発し，CALL教材を一人で作成できる技術的なノウハウを持った外国語教師の数を増やすことはそれほど簡単ではない。しかし，たとえば先に挙げた汎用フォーマットを利用して自分に必要な内容に差し替える程度のレベルであれば，本人に関心さえあれば，短時間の講習でも充分可能である。特に，将来教師を志望する大学院生は，平均年齢の高い現役教員よりも若く，この分野での技能の吸収も早い。CALLのすそ野を広げるためには，汎用フォーマットの提供とならび，こうした意欲を持った院生や教員を対象とした初心者向けの講習会の定期的な実施は不可欠である。もちろんそのためにはこうしたFD活動を制度的に保証するような人的・財政的な支援も要求していかなければならない。

3．3．フリーウエア教材の提供

最近は，教科書の販売を促進するためそれにコンピュータ教材が添付される例も増えてきている。こうした場合，教材作成に費やす労力を考えるとすべて無償で提供するのに抵抗を感じるのは理解できる。しかし，教科書を買わない限り使わせないと言う姿勢でなく，宣伝のための試供品の提供のように考えて，自作教材はできる限り無償で提供しかつ公開することが望ましい。音声を入れれば規模は大きくなるが，最近の圧縮ソフトを使えば大きなサイズのファイルでも，ある程度小さく細切れにすることは可能であり，それをWeb上で公開し，誰もがダウンロードできるようにしておけば，提供する側の労力も少なくて済む。

3．4．スクリプトの公開

教材そのものや汎用フォーマットの無償提供と並んで重要なのは，CALL教材の内部のソースやスクリプトを公開することである。教材を作成しようとする者にとって一番参考になるのは，他人が作った教材が内部的にどのようなアルゴリズムで動いているのかを研究することである。教材作成者の数を増やしていくためにも，アイデアを公開し，できる限りプロテクトをかけず，内部をのぞけるようにすることが重要であろう。

3．5．各言語ごとの音声・画像素材データベース

教材開発自体をどうやって支援するかという観点でもう一つ重要なのは素材の問題である。ドリル型の教材の場合はテキストだけなのでこれはそれほど問題にならないが，たとえば2．2．で

触れた語彙学習型の場合，音声素材と単語の意味を視覚的に提示するための画像データは不可欠である。著作権フリーの画像素材は売られているが，POP 用やインターネットの素材用など用途が限られており，たとえば外国語学習に必要な基本単語のイラストのセットはあまりない。またかりにあっても数百という画像をそのまま一つの教材でセットとして使い，それを配布しようとすれば，それが著作権や使用権フリーの素材であったとしても必ず問題になるであろう。作成者の側からすれば適切な素材を確保することはプログラミングの技法をマスターすることと並んで重要な問題である[23]。ただし先ほどの語彙学習型教材の例を取れば，どの言語でも初習段階での基本語彙をたとえば，500-600語程度に絞ることは可能であり，またその語彙に対応する画像やイラストもその大部分は重なるはずである。したがって，各言語に共通する基本語彙の画像・イラストと言語ごとのその音声をデータベース化し，教育目的での教材作成のために自由に使えるように提供できれば，教材作成者にとって非常に役立つはずである。提供の方法には，Web 上で公開するか CD-ROM に焼きつけて配布するかの方法が考えられる。科研等に申請して一定のアルバイト料や謝金が得られれば，学生諸君の中にはイラストのうまいものはおり，音声を吹き込んでくれるネィテイブスピーカーを見つけることはそれほど難しくないはずである。

3．6．情報交換・情報共有のための場の設定

基本的な原則は，著作権で縛ることなくデータ・経験・アイデア等を積極的に公開し共有していくことである。情報を持っている本人にはささいなことに思えても，他人から見れば価値の多い情報は多いはずである。こうした情報を発信し共有していくためにもメーリングリスト等を通じた情報交換・情報共有の場を作り，それ維持していくことなどが重要であろう[24]。

4．おわりに

前節では，CALL 教材の作成をどうやって支援していくか，また CALL 教材を実際に自分の授業で使ってみようという教師の輪をどうやって大きくしていくかという観点から現状でもできることをいくつか列挙してみた。最後に，こうした教材を使って実際に学習する学生諸君に直接訴えることの重要性も強調しておきたい。CALL 教室や視聴覚教室等がなかったり，その数が少ないため実際にはほとんど使えない状態であったりしてもビデオモニターさえあれば，ポータブルコンピュータとビデオコンバータおよびコネクタを使うことで授業中での CALL 教材の提示やデモンストレーションは充分可能である。その際重要なことは，単なるデモンストレーションにとどまらず，そのあと個々の学生が実際に教材を使えるような配慮である。たとえば Web 上で公開する場合はその情報を学生にきちんと与えること，フロッピーや CD-ROM の形で配布するならそれを自由に（貸し）与えること，また大学に CALL 自習室等がある場合はその部屋の使い方まで説明し，関心を持っている学生に対して入口の敷居をできるだけ低くしてやることが重要である。自分自身の経験から言っても授業等の場でこうしたケアをすると実際に試してみようとする学生の数は格段に増える。(katsuiwa@ipc.hiroshima-u.ac.jp)

付録　教材のスクリプト例

HyperCard に代表されるカード型オーサリングソフトを使った場合の個々の技法に関わるスクリプト例（部分）を以下に付録として挙げる。なお，半角のスラッシュが左端に 2 つ来ている行は注釈行であって，スクリプトそのものに影響を与えない。

(1)問題文のランダムな選択

基本となる問題文データは，“データ”という名のバックグラウンドフィールドの各行にそれぞれ 1）問題文，2）その日本語訳，3）正解が半角のコンマで区切ったアイテムとして入っているものとする。問題文データの例：

```
Ich habe 3 Jahre Deutsch(          )., 私は 3 年間ドイツ語を勉強しました。,gelernt
Was hast du am Wochenende (          )?, 君は週末何をしたの。,gemacht
Bei der Fete hat er viel Wein (          )., コンパでは彼はワインをたくさん飲みました。,get-
runken
```

ランダムな選択の部分のスクリプト例（部分）：

```
on mouseUp
  global _Answer, _Current
  put the number of lines of bg fld “データ” into _Lines
  -- “データ”の行数（＝問題の個数）を調べ，その数を変数_Lines に入れる。
  put the random of _Lines into _N
  -- 1 から_Lines までの数の中で乱数を発生させ選ばれた任意の数を変数_N に入れる。
  put line _N of bg fld ” データ” into _Current
  -- ” データ”の_N 行目 (＝_N 個目の問題 ) をグローバル変数 _Current に記録。問題データを随時参照するため。
  put item 1 of _Current into cd fld “問題文”
  -- _N 個目の問題の問題文を出題問題表示欄であるカードフィールド “問題文” に入れる。
  put item 2 of _Current into cd fld “日本語”
  -- _N 個目の問題の問題文の日本語訳を出題問題の日本語表示欄であるカードフィールド “日本語” に入れる。
  put item 3 of _Current into _Answer
  -- _N 個目の問題の正解をグローバル変数 _Answer に記録する。当該問題の正解を随時参照するため。
  delete line _N of bg fld ” データ”
  -- 用済みになった_N 行目のデータ（＝_N 個目の問題のデータ）を削除する
end mouseUp
```

(2)四択問題の選択肢のランダムな配列

現在処理中の問題文データは，_Current という名のグローバル変数に 1）問題文，2）その日本語訳，3）選択肢 1（正解），4）選択肢 2，5）選択肢 3，6）選択肢 4 が半角のコンマで区切ったアイテムとして入っているものとする。また，バックグラウンドフィールド” 順序” には，１２３４の４つの数の可能なすべての配列24個がそれぞれ半角のコンマで区切られたアイテムとして各行に 1 個ずつ入っているものとする。したがって，1 から24までの数の中で乱数を発生させバックグラウンドフィールド” 順序” の中のその数の行のデータを選べば 1 から 4 までの数字の任意の配列が得られる。

バックグラウンドフィールド"順序"の例（部分）：

1，2，3，4
1，2，4，3
1，4，2，3
：
4，3，2，1

問題文のデータ例（グローバル変数_Current の値）：

Bei der Fete hat er viel Wein (　　)., コンパでは彼はワインをたくさん飲みました。
,getrunken,getrinken,getrinkt,getrunkt

選択肢のランダムな配列のためのスクリプト例（部分）：

put the random of 24 into _Choice
-- 1から24までの数の中で乱数を発生させ選ばれた任意の数を変数_Choice に入れる
put item (item 1 of line _Choice of bg fld "順序" +2) of _Current into bg fld "答 1"
put item (item 2 of line _Choice of bg fld "順序" +2) of _Current into bg fld "答 2"
put item (item 3 of line _Choice of bg fld "順序" +2) of _Current into bg fld "答 3"
put item (item 4 ofline _Choice of bg fld "順序" +2) of _Current into bg fld "答 4"
--グローバル変数_Current の中の（"順序" の中の_Choice 行目の配列の中の 1-4 個目の要素に 2 を加えた数）番目の要素を選択肢 1-4 用のバックグラウンドフィールド "答 1-4" に入れる。なお 2 を加えるのは問題文と日本語の 2 つが 4 つの選択肢の前にあり選択肢番号とそれの位置の item 番号が 2 つずれるためである。

(3)間違えた問題の繰り返し

たとえば，通常の書き込み問題で以下のような現在処理中の問題文データが，変数_Current に入っていたとすると，回答者の解答が誤りであった場合，個々の問題につき 1 回だけこれらのデータをバックグラウンドフィールド"誤り記録"の最後に加えていく。その結果"誤り記録"には，(1)の問題文データと同じ形式で誤った問題文のデータがたまっていく。すべての問題が終わった段階で学習者に違えた問題に再挑戦するかどうか問い合わせ，もし yes なら"誤り記録"を問題文データを記録するバックグラウンドフィールド"データ"に移し，(1)に挙げた手順を繰り返す。

問題文データ（＝ 変数_Current の値）の例：

Bei der Fete hat er viel Wein (　　)., コンパでは彼はワインをたくさん飲みました。,getrunken

"誤答記録"の例：

Ich habe 3 Jahre Deutsch(　　)., 私は 3 年間ドイツ語を勉強しました。,gelernt
Bei der Fete hat er viel Wein (　　)., コンパでは彼はワインをたくさん飲みました。,getrunken

間違えた問題文のデータを記録する部分のスクリプト例（部分）：

put _Current after bg fld "誤り記録"

もう一度繰り返すかどうかを問うスクリプト例（部分）：

answer "間違った問題をもう 1 回やってみます ?" with "NO" or "YES"
IF it is "YES" then
put bg fld "誤り記録" into bg fld "データ"
put empty into bg fld "誤り記録"
else
put empty into bg fld "誤り記録"

```
exit mouseUp
        end IF
```
〔以下(1)に挙げた操作を繰り返す〕

(4)入力された解答の整形

入力された答を変数_X に取って整形した結果をたとえばバックグラウンドフィールド“答”に返すユーザー定義命令 PRE_SHORI のスクリプトは，たとえば以下のようになる。

```
 on pre_SHORI _X
 lock screen
 REPEAT forever
IF offset(numToChar(13), _X) is 0 then EXIT REPEAT
    else put SPACE into char offset(numToChar(13), _X) of    _X
end REPEAT
--まず入力文の中の改行をすべてスペースに変え，
REPEAT forever
 IF char 1 of _X is SPACE then delete char 1 of _X
 else exit REPEAT
end REPEAT
--最初の部分の余分なスペースをすべて取り除き，
REPEAT with i=1 to number of chars of _X
IF char i of _X is SPACE then
REPEAT forever
   IF char i+1of _X is SPACE then delete char i+1 of _X
   else exit REPEAT
 end REPEAT
end IF
end REPEAT
--次に単語間の余分な空白を取り除いて空白を１とし，
REPEAT with j=1 to number of chars of _X
IF (char j of _X is “.” ) or (char j of _X is “?” ) or (char j of _X is ” ,” ) or (char j of _X is “!” ) then
   IF char j+1of _X is not SPACE then put SPACE before char j+1 of _X
      IF char j-1 of _X is SPACE then delete char j-1 of X
   end IF
   end REPEAT
   --さらにコンマや文末記号の前の空白を削って後の空白１を確保し，
   REPEAT forever
   IF (last char of _X is SPACE) or (last char of _X is RETURN) then delete last char of _X
   else exit REPEAT
   end REPEAT
   --文末の余分な空白や改行を取り除き
   unlock screen
   put _X into cd fld “答”
   --整形した結果をバックグラウンドフィールド“答”に入れる
   end PRE_SHORI
```

(5)正解が複数ある場合の処理

現在処理中の問題文データは，_Current という名のグローバル変数に 1）問題文，2）その日本語訳，3）代表的な正解，4）別解 1，5）別解 2，6）別解 3…別解 n が半角のコンマで区切ったアイテムとして入っているものとする。また学習者が打ち込んだ答はグローバル変数_Kotae に，代表的な正解はグローバル変数_Answer にそれぞれ入っているものとする。ここで，代表的な正解と別解1－n を区別するのは，複数の正解がある場合だけでなく正解が一義的に決まるケースもこのフォーマットで処理するためである。なお，以下のユーザー定義関数 Multiple_Answers は，もし学習者が打ち込んだ答が別解 1－n と一致するなら 1 を一致しなければ 0 を返す。ここに挙げた，Multiple_answers では，問題文データの中で別解が第 4 アイテム以降にある登録されている場合を想定しているが，ここの数字は，データの形によって適宜変えればよい。

問題文のデータ例（変数_Current の値）：

Was studierst du?. Ich studiere hier (　　)，君の専門は何？私の専門は(　　)です。,Jura, Wirtschaftswissenschaften, Literatur, Päagogik, Physik, Medizin

解答が複数ある場合の処理のスクリプト例（部分）：

```
IF _Kotae is _Answer or Multiple_Answers _Current,_Kotae is 1 then
--学習者の打ち込んだ答が，代表的正解と一致するかあるいは別解 1－n と一致するならば…
```

第 4 引数以降にある複数解との一致を調べる関数：

```
function Multiple_Answers _Current,_Kotae
put number of items of _Current into _Length
IF _Length > 3 then
--もし変数_Current の長さが 4 以上（＝別解がある）ならば
 REPEAT with i=4 to _Length
  IF _Kotae is item i of _Current then
   put 1 into _X
   exit REPEAT
    --変数_Current の i 番目のアイテムが学習者の打ち込んだ答と一致するなら変数 _X に 1 を入れ
    --REPEAT 文を抜けよ
  else put 0 into _X
  --変数_Current の i 番目のアイテムが学習者の打ち込んだ答と一致しなければ変数 _X に 0 を入れよ
 end IF
end REPEAT
else  put 0 into _X
--もし変数_Current の長さが 3 以下（＝そもそも別解がない）ならば変数 _X に 0 を入れよ
 end IF
 return X
 --関数全体の値として変数_X を返せ
 end Multiple_Answers
```

(6)誤答に対するアドバイスの表示

現在処理中の問題文データは，_Current という名の変数に 1）問題文，2）その日本語訳，3）選択肢 1（正解），4）選択肢 2（＝誤答 1），5）選択肢 3（＝誤答 2），6）選択肢 4（＝誤答 3），7）ヒント 1，8）ヒント 2，9）ヒント 3 が半角のコンマで区切ったアイテムとして入っているものとする。ここで言うヒント 1－3 には具体的にヒントにアクセスするための番号だけが入れある。この番号は，バックグラウンドフィールド”ヒント”に実際に登録されているヒント文の行番号に対応させておく。

問題文のデータ例（変数_Current の値）：

Bei der Fete hat er viel Wein (　　)., コンパでは彼はワインをたくさん飲みました。
,getrunken,getrinken,getrinkt,getrunkt, 1, 5, 3

バックグラウンドフィールド“ヒント”の例：

1，不規則動詞 trinken の過去分詞は，語幹の母音も変化するよ
2，.....
3，不規則動詞の過去分詞の語末はふつう t ではなく en だよ
4，.....
5，trinken は規則動詞ではなく不規則動詞だよ

誤答に対するアドバイスの表示のスクリプト例（部分）：

```
IF _Kotae is item 4 of _Current then put item 2 of line (4+3) of _Current into bg fld “ヒント表示”
--学習者の答が誤答 1 に一致したらヒント 1 をヒント表示欄に入れよ。
IF _Kotae is item 5 of _Current then put line (5+3) of _Current into bg fld “ヒント表示”
--学習者の答が誤答 2 に一致したらヒント 2 をヒント表示欄に入れよ。
IF _Kotae is item 6 of _Current then put line (6+3) of _Current into bg fld “ヒント表示”
--学習者の答が誤答 3 に一致したらヒント 3 をヒント表示欄に入れよ。
```

(7)マウスを使った移動

以下は，マウスを押してつかんでいる間オブジェクト_X をドラッグすることができ，マウスを放したときそれが，オブジェクト_Y の領域内にあれば，_X を_Y に入れるユーザー定義命令 MOVE_OBJECT の例である。たとえば，入力変数_X には，答の文字が書かれたフィールドが，また入力変数_Y にはその文字を入れる解答欄のフィールドがそれぞれはいる。いま，変数_Y にあたる解答欄を cd fldOE“解答欄”とすると，このユーザー定義命令をカードかバックグラウンドのスクリプトに書いておいて，実際に動かしたいオブジェクトには以下の 3 行のスクリプトだけを書いておけば，それをマウスでつかんで解答欄まで動かすことができるようになる。

```
on mouseDown
MOVE_OBJECT me,cd fld “解答欄”
end mouseDown
```

_X を_Y に移動させるユーザー定義命令：

```
on MOVE_OBJECT _X _Y
put the loc of _X into _StartLoc
--_X の最初の位置を_StartLoc に保管せよ
REPEAT while  the mouse is down
--マウスが押されている間繰り返し
set the location of_X to the mouseLoc
--_X の位置をマウスの位置に合わせよ
end REPEAT
IF the mouse is up then IF location of _X is within rectangle of _Y then put _X after _Y
    --もしマウスから手が放れたとき，_X が_Y の領域内にあれば_X を_Y に入れよ。
else
 beep
 set location of card fld _X to _StartLoc
--そうでなければ警告音をならし_X をもとの位置に戻せ
```

```
end MOVE_OBJECT
```

(8)数字をランダムに方陣に配列する

まずカードフィールド“数データ”の各行に順番に1つづつ数字を入れておく。また，ここではマス目の数を16としそれぞれカードフィールド1-16としておく。さらに各数字の読み上げ音声は，数字の前に半角のNを付けた名前のバックグラウンドサウンドに入れておくことにする。たとえば，13ならバックグラウンドN13のように。以下の例で，カードフィールド”選択”には，母集団の中から，乱数を使って任意に選ばれた16個の数字が入っている。数字を読み上げていく場合は，この“選択”の中の16個の数字からさらに乱数を使ってひとつ選ぶことになる。

数字データの中からランダムに16選んでマス目に入れるためのスクリプト（部分）：

```
put cd fld “数データ” into cd fld “数字”
--母集団の数を減らしていくので，もともとのデータを壊さないようカードフィールド “数字” に
--あらかじめ数字データをコピーする。このコピーした母集団をもとに作業を行う。
REPEAT with i＝1 to 16
-- 16はマス目の数。以下の作業を16回繰り返す。
put the random of (number of lines of cd fld “数字” ) into _ERANDA
--乱数を発生し母集団の全体数中から1つの番号を選ぶ
put line _ERANDA of cd fld “数字” into cd fld i
--選んだ番号の行のデータ（ここではたまたま数字）をマス目 i に入れる
put line _ERANDA of cd fld “数字” & RETURN after cd fld “選択”
--選んだデータを cd fld 選択に1行ずつ記録していく
delete line _ERANDA of cd fld “数字” of cd “ゲーム”
--えらび取ってマス目に書いたデータを母集団の中から消す
end REPEAT
put line (the random of 16) of cd fld “選択” *1 into _X
--1から16の間で乱数を発生させ cd fld 選択の任意の行のデータを選ぶ
play bg sound (N & _X )
--選んだ数字を読み上げる。
put _X into cd fld “正解”
--選んだ数字（＝学習者に捜させるべき正解）をカードフィールド “正解” に記録する。
```

(9)クリックしたマス目の数字が読み上げられた数字と同じかどうかのチェック

以下に挙げるのは，数のマス目のフィールドに書くスクリプトの一部である。なお，_LASTTIME というグローバル変数には，直前に答えた時点での時間が入っている。これをもとに，次の解答までの経過時間を計算する。

```
on mouseUp
global _LASTTIME
IF cd fld “正解” is me then
--このフィールドに書いてある数が読み上げられた数字（＝cd fld 正解）と一致したら
get forecolor of me
--もとの数の色をいったん記録し
set forecolor of me to 181
-- 正解したことを視覚的に確認できるよう数の色をピンクに変え（181は windows パレットでピンク）
play bg sound “good”
--正解なので「当たり」のサウンドをならし
```

```
get the seconds
--現在の時間を調べ
IF it <= _LASTTIME +3 then put cd fld "得点" +5 into cd fld "得点"
--前回調べた時間との差が3秒以内なら得点を5点足す
else put cd fld "得点" +3 into cd fld "得点"
-- 3秒以上なら3点しか足さない
wait 90 ticks
-- 1秒半待って
set forecolor of me to it
--数字をもとの色に戻す
put the random of (number of lines of cd fld "数字") into _ERANDA
--数字データの中から乱数で選んだ任意の行を取り出し,
put line _ERANDA of cd fld "数字" into me
--その行にある数で先ほどの正解した数を上書きする
put line _ERANDA of cd fld "数字" into line the Number of me of cd fld "選択"
--これはn個目のマスなので cd fld "選択" のn行目にもこの数を上書きする
delete line _ERANDA of cd fld "数字"
--この数を cd fld "数字" から削除
(以下略)
```

⑽時間経過を分と秒で表示するタイマー

たとえば,"タイマー"という名前のカードフィールドを作り,以下の2つのスクリプトをそれぞれ開始ボタンとカードに書いておけば,開始ボタンを押して以降の時間経過を表示させることができる。時間表示の基本は,組み込み関数 seconds である時点での時間情報をとってそれをグローバル変数に記録し,次に比較すべき別の時点で再び seconds を使って時間情報をとり,両者の差から時間経過を得ることである。

タイマーを始動する開始ボタンのスクリプト:

```
on mouseUp
  global _LASTTIME
  get the seconds
  put it into _LASTTIME
  end mouseUp
```

時間経過を表示するタイマーの基本スクリプト:

```
on idle global _LASTTIME
IF _LASTTIME is not empty then
 put the seconds __LASTTIME into _X
  put _X div 60 & ":"  & _X mod 60 into _PASSAGE
  put _PASSAGE into cd fld "タイマー"
  end IF
end idle
```

⑾テキストを前から順に消していく

一定の時間経過後に以下のスクリプトを始動すればテキストを前から順に消していくことができる。

```
REPEAT for number of chars of cd fld "Text"
lock screen
```

```
delete first char of cd fld "Text"
unlock screen-- 画面を一時的に lock するのは delete の際のちらつきを防ぐため
wait 3 ticks
end REPEAT
```

⑿辞書引きの基本

"辞書"という名前のスタックにアルファベット記号を名前としたカードを作り，そこにたとえば"辞書項目"という名前のバックグラウンドフィールドを作っておく。たとえば，カードSのバックグラウンドフィールド"辞書項目"の各行には，1）辞書の見出し語，2）欧文フォントで表示する情報，3）日本語フォントで表示する情報，4）サウンド名にするため翻字された音声用見出し語をそれぞれ半角のコンマで区切ったアイテムとして入れておく。そのうえで，テクストの書かれたフィールドに以下のスクリプトを書いておけば，マウスで単語をダブルクリックするだけその単語の辞書を引き，バックグラウンドフィールド"検索結果"に表示することができる。

辞書データの例（＝バックグラウンドフィールド"辞書項目"の値）：

schön, schn, 形容詞「すてきな」「美しい」,schoen
sehr,sehr, 程度の副詞　形容詞や副詞を強める「とても」,sehr
sprech, sprechen/sprach/gesprochen du sprichst/er spricht,「話す」＝speak（英語）,sprechen
sprich, sprechen/sprach/gesprochen du sprichst/er spricht,「話す」＝speak（英語）,sprechen

辞書引きのためのスクリプト例：

```
on mouseUp
  set the lockText of me to false
  wait 30 ticks
  IF the mouseClick is true then READ_UP
  else set the lock Text of me to true
  end mouseUP
  on SEARCH_WORD
  click at the clickLoc
  click at the clickLoc  -- 2 度クリックすることで単語がとれる
  put the selection into _X
  put the selection
  IF ((charToNum(_X) > 96) and (charToNum(_X) < 123)) or ((charToNum(_X) > 64) and
  (charToNum(_X) < 91)) then--アルファベットを明示的に指定し空白/日本語には反応しないようにする。
  put empty into bg fld "検索結果"
  hide bg fld "検索結果"  -- 前の検索結果をクリアして表示窓をいったんしまう
  put first char of _X into _IKISAKI        -- 単語の頭文字を変数 _IKISAKI に入れる
  REPEAT with i=1 to (number of lines of bg fld "辞書項目" of card _IKISAKI of stack "辞書")
  put item 1 of line i of bg fld "辞書項目" of card _IKISAKI of stack "辞書" into _Y
  IF _X contains _Y then        -- 辞書の見出し語が検索語に含まれるならば
    DISPLAY_WORD line i of bg fld "辞書項目" of card _IKISAKI of stack "辞書"
    set lockText of me to true        --このフィールドを再びロックしておく
    exit mouseUp
    end IF
  end REPEAT
  PUT "辞書に登録されていません" into bg fld "検索結果"
  show bg fld "検索結果"
```

```
end IF
set lockText of me to true -- このフィールドを再びロックしておく
end SEARCH_WORD
on DISPLAY_WORD _X
PUT item 2 of _X into line 1 of bg fld "検索結果" -- 基本形/活用などを表示
SET TextFont of line 1 of bg fldbg fld "検索結果" to Helvetica
SET TextSize of line 1 of bg fld bg fld "検索結果" to 18
PUT item 3 of _X into line 2 of bg fld "検索結果" -- 日本語の意味や説明などを表示
SET TextFont of line 2 of bg fldbg fld "検索結果" to Osaka
SET TextSize of line 2 of bg fld bg fld "検索結果" to 12
show bg fld " 検索結果"
IF exists(bg sound item 4 of _X of stack "辞書" ) then play bg sound item 4 of _X of stack "辞書"
--スタック "辞書" に音声用見出しの名前で音声が登録されていればそれを流す
end DISPLAY_WORD
```

参考文献

第3回ドイツ語教授法ゼミナール実行委員会（1997）第3回ドイツ語教授法ゼミナール報告 (Dokumentation des 1. Didaktikseminars für japanische Germanisten 1996). 東京ドイツ文化センター.

Dreyer, H./Schmitt, R. (1985): Lehr- und Übungsbuch der deutschen Grammatik. München: Verlag für Deutsch.

Heaton, J.B. (1990): *Writing English Language Tests.* London and New York:Longman.

岩崎克己/吉田光演（1998）：ドイツ語でジャンプ. 白水社.

Johnson, K./Morrow, K. (eds.)(1981): *Communication in the Classroom.* London: Longman.

Krashen, S.D./Terell, T.D.(1983): *The Natural Approach: Language Acquisition in the Classroom.* Oxford: Pergamon Press.

Levy, M.(1997): *Computer-Assited Language Learning: Context and Conceptualization.* Oxford University Press.

Richards, I.A./Mackey, I.S./Mackey, W.F./Gibson, C. (1953): *German Through Pictures I,II.* Language Research, Inc. (1989)Tokyo: Yohan Publications, Inc.

掌田津耶乃(1996): Oracle Media Objects パーフェクトマニュアル上巻/下巻 Macintosh 版. アスキー出版局.

掌田津耶乃(1996): Oracle Media Objects パーフェクトマニュアル基礎編/応用編 Windows 版. アスキー出版局.

掌田津耶乃(1994)：改訂入門 HyperCard. アスキー出版局.

掌田津耶乃(1994)：改訂実習 HyperCard. アスキー出版局.

掌田津耶乃(1995)：改訂応用 HyperCard. アスキー出版局.

Spier, A.(1981): Mit Spielen Deutsch Lernen. Frankfurt am Main: Cornelson Verlag.

Ur, P./Wright A.(1992): *Five-Minute Activities.* Cambridge University Press.

山内豊(1996)：インターネットを活用した英語授業. NTT 出版.

註

1）本稿ならびに本稿で紹介したCALL教材は，1997年度の広島大学外国語教育研究センタープロジェクトとして採択された「CALL教材開発運用プロジェクト」の一環として執筆および制作された。

2）マルチメディア自習室の概要については、広島大学外国語教育研究センター年報1997参照。

3）本稿で紹介したものを含め広島大学外国語教育研究センターの専任スタッフ・研究員らによって自主開発された教材はWeb上で公開されている (http: //www.ipc.hiroshima-u.ac.jp/~langlabo/ free-soft)。教材作成に用いられたオーサリングソフトは、主に、HyperCardかOracle Media Objects Ver 1.1 である。

4）以下，2.1，2.6，2.7で紹介するタイプは主に「指導の形式」にのみ関わるのに対し，2.2，2.3，2.4，2.5で紹介するそれは「指導の内容」にも関わる。このため，一見するときちんとカテゴライズされていないような印象を与えるかもしれない。しかし，ここでのタイプ分類の基準は，実際に教材を制作しようとする際の技術的な枠組みの違いであって、養成しようとする能力や問題として出される課題の種類を基準としているわけでもなければ，またその実現形式を基準としているわけでもない。

5）ドリル型クローズテストの形式面および内容面でのさまざまな分類については，Heaton, J. B. (1990) pp.34-50 を参照。

6）マルチプルチョイス形式の代表例としてここでは四択問題を挙げたが，Heaton, J. B. (1990) p.28 にはテストとしての有効性と問題作成の際の経済性の観点から選択肢の個数について次のような記述がある。“The optimum number of alternatives, or options, for each multiple-choice item is five in most pubic tests. Although a larger number, say seven would reduce even further the element of chance, it is extremely difficult and often impossible to construct as many as seven good options. Indeed, since it is often very difficult to construct items with even five options, four options are recommended for most classroom tests. Many writers recommend using four options for grammar items, but five for vocabulary and reading.”

7）通常の文法ドリルのような問題でも解答をまず音声で流し，次に文字等で与えるというようなことをすれば，副次的には聞き取り練習の要素を持たせることができる。

8）学習記録の利用法として成績管理がよく挙げられる。しかしドリルでの獲得点数や練習した時間数を成績管理に使うことには筆者は，必ずしも賛成できない。問題に取り組む時間や条件をある程度標準化できる一斉テストなどは別だが，学習過程をコントロールできない状況で，しかも正解かどうかの機械的判断によって数値化された得点を成績管理に使うのは，その信頼性からして問題がある。また成人の学習者を対象とした場合，ドリル型教材で中心的に扱われる文法や語彙の体系的な学習は，外国語能力の養成にとって意味を持つとはいえ，それ自体はけして自己目的とはならない。文法や語彙のドリルで何点取ったかで成績を付けようとする発想そのものが狂っている。ただしこれはコンピュータによる成績管理以前の問題である。

9）たとえばGDMによるドイツ語学習用教材であれば Richards, I. A. / Mackey, I, S. / Mackey, W. F. / Gibson, C. (1953) がある。

10）Johnson, k. / Morrow, k. (1981) 参照。

11）自由に配布するのであれば，画面の小さなコンピュータでもきちんと表示されるようにしておかなければならない。たとえばポータブルコンピュータのモニタサイズぎりぎりの640×480ではタイトルバーなどがあると画面の端が切れてしまうので，教材の表示画面はそれよりひとつ小さい512×348ピクセル程度のサイズに納めるのが無難である。

12）画像として写真を使うなら，大きさからして4個までが理想的で9個にするとやや小さめだが，イラスト化するなら16個でも充分実用になる。教材の表示画面を512×348ピクセルとした場合，それぞれの画像の大きさは，4個の場合は200×160ピクセル，9個は130×105ピクセル，16個なら100×80ピクセル程度になる。

13) この点では，先に挙げた Rosetta stone 自体はここで紹介するタイプの典型とはいえない。選択肢も4つと少なく，全部答えてから次に進むという学習の進め方からしてもドリル的性格が強い。新しい語彙を増やしていくという点では実用性が高いが，全体としてゲーム性に乏しくおもしろくない。

14) 実際，筆者は，初級ドイツ語教科書「ドイツ語でジャンプ」（白水社刊）付属の対話教材をこの DialogMaker を使って作ったが，26個のダイアローグからなる教材作成に要した時間は，約6時間であった。

15) たとえば広島大学総合科学部吉田光演氏作の ReadText 1.0 などがある。

16) 英語の場合，すでにオンラインで使うことができる本格的な辞書があり，大学教育においてはこうしたものは必ずしも必要ではないかもしれない。しかし，英語以外の外国語においては，そうした辞書はまだほとんどなく，また学習者が入門段階の初級者であることも多いため本格的な辞書よりも手軽に使える単語帳型辞書の方が使いでが良いという事情もあって，内容を簡単に差し替えられる汎用型 glossary を作ることの意義は大きい。

17) ちなみに，筆者は，これを勝手に Virtual Physical Response と呼んでいる。

18) これにはまた，コンピュータの画面上で，動作や反応などを示し，それを言語的に記述させる逆方向のアプローチも可能である。

19) マウスでものを動かす操作は本稿の付録(7)で紹介したものと同じである。経路の表示については，オーサリングソフト Oracle Media Objects の場合「パス」というアニメの道筋を指定するオブジェクトがあるのでこれを使うと便利である。アナログ時計の長針と短針の動きについては，掌田（1996）下巻 pp.48-52 参照。

20) これは，Web 上で公開されているフリーソフトである。http://www.shareware.com/ の検索画面で，tug-of-word という検索語をいれて捜すとダウンロード可能な最寄りのサイトがいくつか表示される。

21) 広島大学外国語教育研究センター主催のシンポジウム「マルチメディア時代の外国語教育－CALL の実践・成果・問題点をめぐって－」(1997年12月13日）での三枝裕美氏（京都大学人間学部）の報告によれば，日本で市販されている CALL 教材の各言語ごとの数は，英語，フランス語，ドイツ語，中国語，ロシア語についてそれぞれ，58，8，5，7，2だそうである。

22) たとえば2.3.の図3に挙げた欧文対話教材作成ツール DialogMaker もこうした汎用フォーマットの一つとして構想されたものである。また2.2.の図2に挙げた「数にチャレンジ」も，音声の部分を差し替えればそのままどの言語でも汎用に使えるようになっている。

23) なおニュース等のビデオ映像に関しては，たとえば CNN の場合，正規のライセンスを受けて視聴しているという前提のもとでなら，大学などの教育機関がタイムリーな映像素材を非営利目的で教材作成に使うことが認められている。

24) CALL 関係のメーリングリストは各言語ごとにいくつかあるが，CALL に関心を持つ多くの言語の関係者に開かれたメーリングリストの一つに広島大学外国語教育研究センターにサーバーがおかれた Hi-CALL がある。現在，英語・ドイツ語・フランス語・中国語・スペイン語・日本語などの研究者・教育者を中心に50人（1997年12月20日現在）ほどの登録者がある。具体的な登録方法については，本誌吉田論文の註5参照。

ABSTRACT

In-House Multimedia Teaching Materials Development

Katsumi Iwasaki
Institute for Foreign Language Research
and Education, Hiroshima University

The purpose of this paper is to promote CALL and in-house multimedia development at educational institutions, so that more language teachers can make teaching materials suitable for their classes.

In the first half of this paper some multimedia teaching materials made with the help of authoring software such as HyperCard and Oracle Media Objects are introduced and classified into seven main types as follows:

- CAI drills (traditional cloze tests performed on a computer),
- Vocabulary learning programs with pictures and sound,
- Dialogs with sound and/or movies,
- Programs for reading exercises,
- An on-line glossary,
- Virtual Physical Response with mouse (Total Physical Response realized on a computer),
- Other games.

For each type the development know-how and technical difficulties are delineated, especially from the viewpoint of hands-on development within educational institutions. Some core programming scripts are also added as an appendix along with brief explanations.

In the second half of this paper I propose the following six practical measures for promoting CALL and in-house multimedia development. They are all easy to realize and require minimal financial expenditures.

- Developing and offering so-called "general format programs". This expression refers to the customisable multimedia titles whose data are easy to change by each user;
- A CALL training course for graduate students and young teachers as part of faculty development activities in the university;
- Mutual offering of freeware multimedia titles per WWW;
- Opening the know-how and programming scripts to the public;
- Developing a data-base of copyright-free sound and pictures for nonprofit educational purposes;
- Opportunity for continuous discussing and exchanging of information among CALL specialists, e.g. a CALL mailing list.