IntelligentPad における演劇機能の実現†

赤石美奈†† 田中 譲††

北海道大学で開発されている IntelligentPad システムでは，あらゆるメディアをコンピュータ上の一枚の紙（パッド）とみなし，メディアをパッドというダイナミック・オブジェクトとして扱う．各メディアに対応し，各パッドが定義され，各パッドはそれぞれ固有の機能を持つ．さらに，パッドの機能はパッドを貼り合わせることで機能の合成ができる．合成されたパッドもまた一枚のパッドとして扱える．これらのパッドを制御するパッドとしてステージパッドを新たに開発した．ステージパッドは劇のメタファを用いている．劇は舞台，楽屋，役者，台本，観客などで構成される．舞台はステージパッド，楽屋はドレッシングルームパッド，役者は制御対象となるパッド，台本はエディタ用のパッドで実現される．これらのパッドの合成により劇を実現する．劇の構成要素はすべてパッドであり，部品として交換・再利用できる．また，劇自身もパッドであり，劇（パッド）を合成することで劇中劇も実現される．本論文ではパッドによる劇の構成と，ステージパッドの動作機構について解説するとともに，それらの応用例について述べる．

1. はじめに

近年，コンピュータの普及は様々な分野に及んでおり，様々な人に利用されている．それに伴い，ソフトウェアに対するユーザの要求も多様化している．例えば，CAI (Computer Aided Instruction) のように，教師が授業にコンピュータを用いる場合，既製のソフトのみで各教師の要求を十分に満たすことは難しい．効果的な授業を行うためには，教師自らの経験や考えに基づいたソフトウェアが必要となる．このためには，教師自身がソフトを修正，開発することも必要となる．しかし，C, Lisp, Smalltalk 等のプログラミング言語を用いてプログラムを組むことは，それを専門としない非プログラマにとっては困難を伴う．このため，様々なオーサリング・システムが開発されている．

Macintosh の HyperCard[1),2)] は，その一つである．HyperCard には，カード，フィールド，ボタンのオブジェクトが用意されており，各オブジェクトの動作は，それぞれのスクリプトに記述される．ユーザは，これらのオブジェクトを用い，スクリプトを書き，容易にオーサリングを行うことができる．しかし，新たなオブジェクトの導入や，スタック自身を部品として別のスタックを構成することはできない．(Macintosh のソフトに起動をかけたり，別のスタックにリンクを張ることはできる．) 将来，必要となる部品すべてをシステムがあらかじめ用意しておくのは不可能である．このため，新たなオブジェクトの導入を許すオープンなシステムが望まれる．

北海道大学で開発されている IntelligentPad[3)~8)] システムは，様々なメディアをコンピュータ上の一枚の紙（パッド）として表現し，複数のメディアを統一的に扱う．各メディアに対応して各種のパッドが定義される．テキストを扱うパッド，画像を扱うパッド，音声を扱うパッドなど，あらゆるメディアをパッドとして扱う．システムにより提供される基本的な部品をプリミティブ・パッドと呼ぶ．また，パッドは，『貼る』というメタファにより機能の合成ができる．合成パッドも一枚のパッドとして定義され，部品として利用される．貼り合わされたパッド間のインタフェースは標準化されており，個々のユーザの要求に応じて様々なパッドを自由に組み合わせる（貼り合わせる）ことができる．新たなアプリケーションの開発も，すべてを新たに開発するのではなく，それまでのシステムに欠けていたプリミティブな機能のみをパッドとして実現し，これと既存のパッドを合成することにより遂行できる．これはプログラミングの負担を著しく軽減する．

これまでの IntelligentPad システムには，複数のパッドを制御して，一連の仕事をさせる仕組みが欠けていた．このため，複数のパッドを制御するためのパッドとして新たにステージパッド[9)]と呼ぶプリミティブ・パッドを開発した．これにより，IntelligentPad システムを用いたオーサリングが容易に行える．本論文では，ステージパッドの構成，および動作機構について述べる．また，パッドを用いたオーサリングをいくつか述べる．開発は，Smalltalk-80[10),11)] を用

† Implementing Drama Performance in IntelligentPad by MINA AKAISHI and YUZURU TANAKA (Electrical Engineering Department, Faculty of Engineering, Hokkaido University).

†† 北海道大学工学部電気工学科

いて行った．

以下の章でステージパッドに関して詳しく述べていく．3章でスクリプトについて述べ，4章でステージパッドの動作機構について述べる．5章ではステージパッドの合成について述べ，6章では楽屋について述べる．

2. ステージパッドの概要

ステージパッドは，劇のメタファを用いている．劇のメタファを用いた試みはこれまでにもいくつかなされている．そのひとつに視覚的プログラミング環境を提供する Rehearsal[12] システムがある．このシステムでは，Smalltalk のオブジェクトを “performer” とみなす．“performer” は “stage” 上にあり，互いに “cue” を送り合い動作する，これは，Smalltalk のオブジェクトを視覚化したもので，ビジュアルなプログラミング環境を提供するものである．

ステージパッドは，パッドを制御するために劇のメタファを取り入れた．ステージパッドの特徴は，劇を構成する要素がすべてパッドという形態に統一されていることである．これにより，劇の構成要素は，すべて部品化される．さらに，劇自身も部品化される．

劇は，舞台，楽屋，役者，台本，観客等により構成される．パッドによる劇において，舞台はステージパッドであり，ステージパッドはドレッシングルームパッドという楽屋を持つ．役者は，制御対象となる任意のパッド（以後アクタパッドと呼ぶ）である．アクタパッドは，ステージパッド上か，ドレッシングルームパッド上になければならない．また，台本の記述には，テキスト・エディタパッドを用いる．テキスト・エディタパッドには，役者の動作が記述されており，その内容は，パッドの貼り合わせを通じてステージパッドに伝達される．

ステージパッドは，マウス入力やキーボード入力などのユーザイベントやパッドの動作をきっかけに，台本に書かれた指示を各役者に伝える．それぞれのパッドはその指示に基づき動作を開始する．このようにして，ステージパッド上ではアクタパッドが劇を繰り広げる．ユーザは単なる観客ではなく，スクリプトを書くことや，ユーザイベントを発生させることにより，劇の流れを変化させることができる．

図1はステージパッドを用いた簡単なアプリケーションである．図1では，ステージパッド上にゴリラの絵のついたアニメーションパッドと，カウンタが貼られている．アニメーションパッドは，複数枚の絵を保持し，それを順に次々と切り換えて繰り返し表示する機能を持つ．カウンタパッドは，コマンドを送ることでカウントアップや，カウントダウンを行う機能を持つ．これにコマンドを送るためのボタンパッドや，カウントを表示するための表示パッドを貼り付け，カウンタとしている．スクリプトには，『‘ゴリラ’ を動かすと，‘ゴリラ’ は，左右に動き続ける』，『‘ゴリラ’ をクリックすると，‘ゴリラ’ は削除され，‘カウントアップボタン’ をクリックする』という内容の指示が書かれている．‘ゴリラ’ を動かしたり，クリックをするとスクリプトの内容が実行される．これで，簡単なゴリラハント・ゲームが実現される．図2では，ステージパッド上にキーボードパッドが貼られている．キーボードパッドは，送られたコマンドに応じた音を出す機能を持つ．これにボタンパッドを貼りキーボードとしたものである．スクリプトには，『‘ド’ をクリックされたら，‘ミ’ と ‘ソ’ をクリックする』，『‘シ’ をクリックされたら，‘レ’ と ‘ソ’ をクリックする』という内容の指示が書かれている．これも，‘ド’ をク

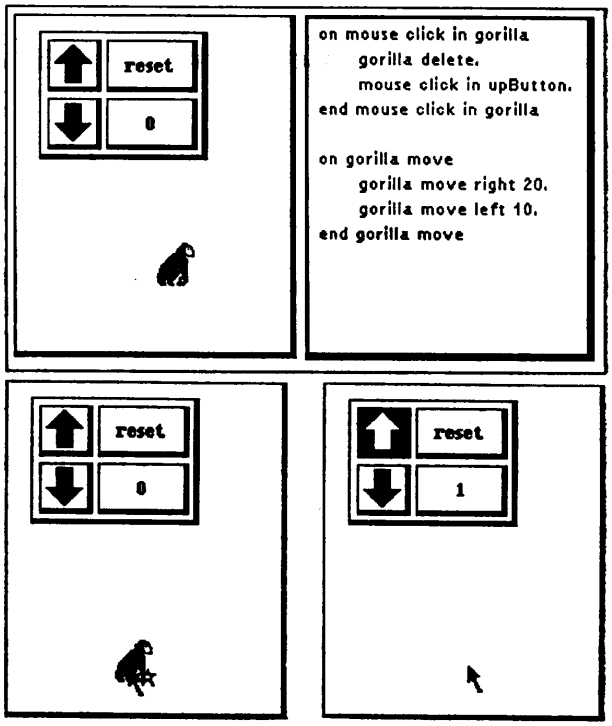

図1 ステージパッドによるゴリラ・ハント・ゲーム
Fig. 1 The gorilla hunting game implemented with a StagcPad.

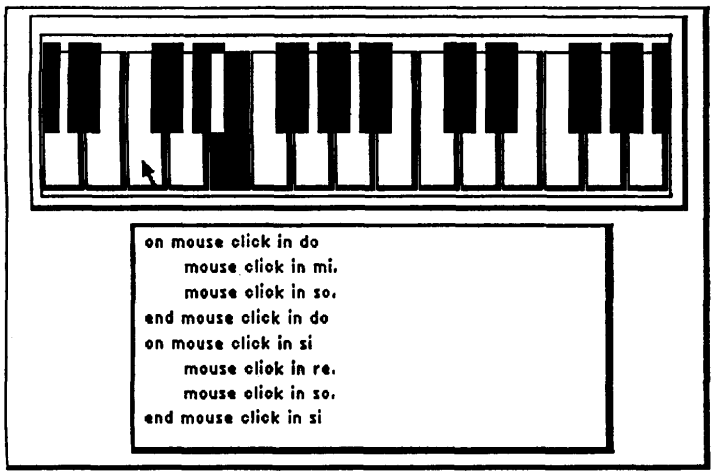

図 2 ステージパッドによるキーボードの自動演奏
Fig. 2 The automatic play of a music keyboard implemented with a StagePad.

リックされたり，'シ' をクリックされたことをきっかけとしてスクリプトが実行される．

3. スクリプトとキャスティングリスト

劇を構成する要素のひとつに，スクリプトがある．スクリプトは，ステージパッド上の各パッドにユーザの意図した動作をさせるための手続や制約を記述したものである．ステージパッド上の各パッドはこのスクリプトの記述に従い動作する．また，スクリプト中では役名を用いて間接的にパッドを指定するため，役名とその役を実際に演じるパッドとの間の関係を定義しなければならない．キャスティングリストは，スクリプト中で指定される役名と実際のパッドとの対応関係を与えるものである．

以下で，スクリプトとキャスティングリストについて詳しく述べる．

3.1 スクリプト

ステージパッドのスクリプトは，これを演じるアクタパッドとしてどのパッドを用いるかということからは独立して書かれる．スクリプトを独立させることにより，任意のパッドをアクタパッドとして用いることができる．また，あるステージを想定して書かれたスクリプトを，別のステージでもそのまま書き直さずに利用できる．スクリプトとアクタパッドを独立させることにより，スクリプトの部品化，アクタパッドの部品化がなされている．

ステージパッドではスクリプトの独立性を保つために，次のような工夫をしている．まず，スクリプト中でのパッドの参照はロールネーム（役名：パッドのスクリプト中での呼び名）を用いて行うこととし，ロールネームとパッドとの対応関係を与えるものとしてキャスティングリストを導入した．また，スクリプト中で用いられるコマンドは，ユーザがパッドに対して行える基本操作のみに限定した．特定のパッドのみが理解できるコマンドを使用すると，あるロールネームに割り当てることのできるパッドの種類を制限することになり，スクリプトの独立性が保たれなくなる．基本操作のためのコマンドは，すべてのパッドで理解される．また，基本操作以外のコマンドを送る場合は，そのコマンドを送るためのボタンパッドを貼り，ステージパッドがそれをクリック（基本操作）することにより，コマンドを送ることができる．

ステージパッドはイベント・ドリブン方式で動作する．あるイベントをトリガとして，次に起こるべき一連の動作を各パッドに指示し劇を進めていく．スクリプトには，トリガとなるイベントごとに，その後に続く動作の系列が記される．

3.2 スクリプトの記述

（1） 記述形式

スクリプトは，『あるイベントが生じた時に，各アクタは以下の命令を実行せよ』という形式で記述される．**on** の後に続けて，そのスクリプトを実行するきっかけとなるイベント名を記述する．その次に各アクタへの指示を順に記述していく．スクリプトの終わりは **end** と，そのスクリプトを実行するきっかけとなったイベント名を記す．

on<イベント名>
<actor1>command1.
<actor2>command2.
<actor3>command3.
.....
end<イベント名>

（2） スロット値参照コマンド

各パッドの primary slot と呼ぶスロットの値を参照できる．(primary slot とは，パッドの持つ先頭スロットの値)

（3） パッドのアクション（行動）コマンド

以下のコマンドを用いてパッドにアクションを行わ

せる.

○move to<destination>パッドの移動
○copy<location> パッドの複製
○delete パッドの削除
○hide パッドの透明化
○show パッドの表示
○open パッドのアイコンを開く
○close パッドのアイコン化
○resize to<new size> パッドのサイズ変更
○paste on<actor> パッドの貼付
○assoc<new slot> スロット接続の変更
○mclick マウスクリック時の動作
○mdrag to<destination>
マウスドラッグ時の動作
○mkey<key> キー入力時の動作

(5) パッド間の制約(パッドの位置に関する制約)

パッド間に制約を設けることができる. 制約を付けられたパッドは常にその制約を満たすように動作を規制される.

図3では, PadA は PadB との距離をxに保つように制約が付けられている. この制約のもとで, PadA が動かされた場合, PadA は制約を満たす位置に移動する. また, PadB が動かされた場合にも, PadA は制約を満たす位置に移動する. PadA は常に制約を満たすように動作する.

○<actor1>is<x>from<actor2>
actor1 は actor2 から距離xを保つ
○<actor1>is touching<actor2>
actor1 は actor2 に接する
○<actor1>is<x>overlapping<actor2>
actor1 は actor2 に距離x重なる

(6) 制御文

パッドの一連の動作を記述する際, 以下の制御文を用いることができる.

○**if**<条件>**then**<action>**else**<action>**end if**
○**while**<条件><action>**end while**
○**until**<条件><action>**end until**
○**repeat**<n> <action>**end repeat**

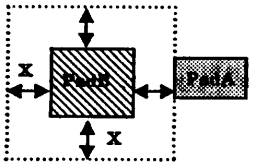

図3 パッド間の位置に関する制約
Fig. 3 Location constraint.

3.3 キャスティングリスト

スクリプト中でのパッドの**参照**は, ロールネームでなされている. ステージパッドがアクタパッドを操作するためには, ロールネームとパッドとの対応関係が定義されていなければならない. 両者を結び付けるのがキャスティングリストである. キャスティングリストは, ロールネームとオペレーションポイントのリストから成る連想リストである.

CastingList
((RoleName$_1$ (OP_{11} OP_{12}……))
(RoleName$_2$ (OP_{21} OP_{22} OP_{23}…))
……)

まず, オペレーションポイントについて説明する. ユーザがマウスを用いてパッドに対する操作 (move, copy 等) を行う場合, 操作の対象となるパッドの領域内にマウスを移動する. このときのマウスの位置を操作点と呼ぶことにする. オペレーションポイントは, ステージパッドがアクタパッドを操作する場合の操作点である. オペレーションポイントは, 自分が付随するパッドと位置情報を保持している. このオペレーションポイントをあらかじめパッドに割り当てておく (オペレーションポイントを割り当てられたパッドはアクタパッドとなる). その結果, ユーザがマウスを用いて行う操作と同等の操作を, ステージパッドからオペレーションポイントを通じて, アクタパッドに指示することができる. ユーザによるパッドの操作とステージパッドによる操作を統一的に扱えるようにオペレーションポイントを用いるのである.

キャスティングリストで, オペレーションポイントを用いることにより, パッド (アクタパッド) の独立性を保つことができる. ユーザによるパッドの操作とステージパッドによる操作は統一的に扱えるため, ステージパッドにプリミティブ・パッドおよび合成パッドが操られる際, それぞれのパッドの持つ機能は損なわれない. 例えば図2のように, キーボードがステージパッド上にある場合, このパッドはキーボードとしての機能をそのまま保持し, かつ (オペレーションポイントが割り当てられている場合) ステージパッドの指示にも従う. キーボードは, ステージパッド上にある時にも, ステージパッドから剝がされている時にも, キーを押されるとそれに応じた音を出すという動作を

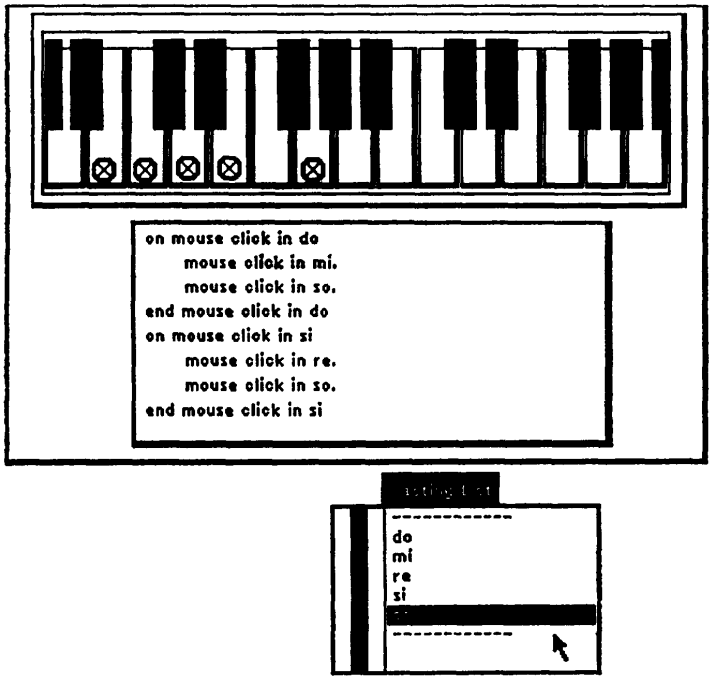

図 4 キャスティングリストとオペレーションポイント
Fig. 4 A CastingList and some OperationPoints.

することに変わりはない．図4は，図2の場合のキャスティングリストとオペレーションポイントを示す．ステージパッドは，オペレーションポイントを用いることにより，IntelligentPad システムで作られたすべてのパッドを，個々の機能を失わずにそのまま利用することができるという利点を持つ．

キャスティングリストを用いてロールネームとパッドを結び付けることにより，パッドはスクリプトに記述されている自分の役割を演じることが可能となる．ある役を様々なパッドに割り当てたり，ひとつのパッドにいくつもの役を割り当てることで，同一のスクリプトをもつステージパッドに異なる劇を展開させることもできる．割当てを変えることで様々な劇を実現する．

3.4 アクタパッドの参照

図6は，図5の貼り合わせを例にとり，スクリプトがどのようにパッドへ伝わるかを示している．

スクリプト中において，ロールネームを用いてパッドを指定する場合，以下の記述形式がある．

（1） 〈RoleName〉

ロールネームに割り当てられているオペレーションポイントに付随するパッドを示す．

(図6のスクリプトの1行目 Jiro move は，キャスティングリストの {Jiro : OP_2} により OP_2 に伝えられる．OP_2 は，自分の付随するパッドである PadB にメッセージ (move) を送り，PadB が move を実行する．)

（2） Pad at @<RoleName>

ロールネームに割り当てられているオペレーションポイントの位置情報により，その位置にある一番上（表面に現れる）のパッドを示す．

(図6のスクリプトの2行目 Pad at @Jiro move は，キャスティングリスト {Jiro : OP_2} により OP_2 に伝えられる．OP_2 は，自分の位置情報から，その位置にあるパッドのうち，一番表面にあるものを探索する．その結果，PadA にメッセージを送り，PadA が move を実行する．)

（3） @<RoleName>

ロールネームに割り当てられているオペレーションポイントの位置情報を示す．

(図6のスクリプトの3行目 @Jiro move は，キャスティングリスト {Jiro : OP_2} により OP_2 に伝えられ

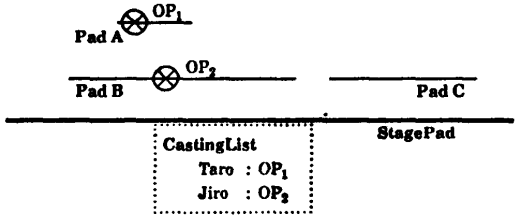

図 5 ステージパッドとキャスティングリスト
Fig. 5 A StagePad and a CastingList.

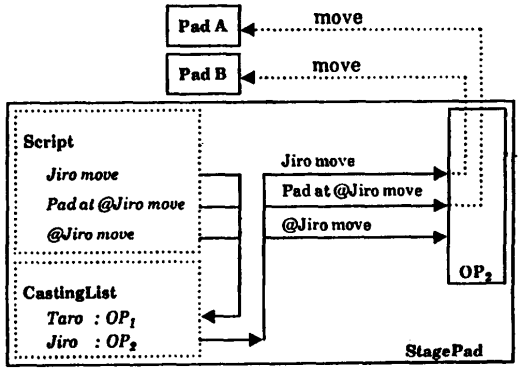

図 6 スクリプトによるアクタの参照
Fig. 6 References to the ActorPads.

る．OP_2 は自ら move を実行する．）

（1）の記述は特定のパッドを常に参照でき，（2）の記述は位置情報からパッドを参照することができる．（3）の記述はオペレーションポイントの位置情報を参照する．

4. ステージパッドの動作機構

ステージパッドの動作機構は，スクリプトを読み込み，それを解析する部分，イベントを検知し，解釈する部分，スクリプトに従いアクタパッドを動作させる部分に分けられる（図7参照）．

4.1 スクリプトの解析

スクリプトの解析は，ScriptParser によって行う．その結果，スクリプトは，連想リストの形式でステージパッド内部に保持される．この連想リストは，スクリプトを実行するきっかけとなるイベントの名前をキーとし，そのイベントに伴う一連の動作のリストを値に持つ．さらに，スクリプトを解析すると同時に，その中からロールネームを切り出し，キャスティングリストを生成する．

4.2 イベントの解釈

ステージパッドでは，二種類のイベントを扱う．一つはマウスのクリック，ドラッグ等のユーザイベントである．もう一つは，パッドの動作により発生するパッドイベントである．パッドイベントの種類はパッドの基本操作と同じである．

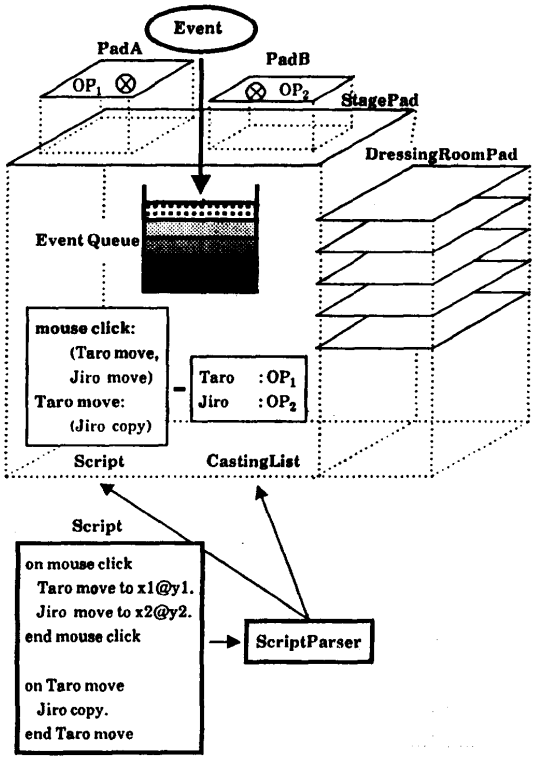

図 7 ステージパッドの動作機構
Fig. 7 The StagePad mechanism.

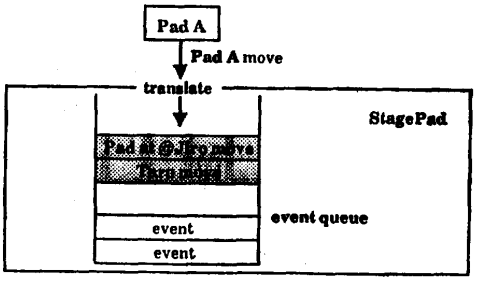

図 8 イベント伝達と解釈の例
Fig. 8 The propagation and interpretation of events.

ステージパッドは，イベントをトリガとして，劇を開始する．そのためには，ステージパッド上のすべてのイベントを検知しなければならない．IntelligentPad システムでは，イベントを発生させたパッド，もしくはユーザイベントを検知したパッドは，そのイベントの種類，および，発生源であるパッド自身を，パッドの貼り合わせを通じて自分の貼られているパッドに次々と伝達する．ステージパッドは，このイベント情報を検知する．

さらに，ステージパッドでは，伝えられたイベントを解釈する必要がある．なぜなら，ステージパッドは，スクリプト中のロールネームによってパッドを参照するため，イベントを生成したパッドを，対応するロールネームを用いて示さなければならないからである．図8は，図5の状態で PadA を動かした場合の例である．PadA から，[PadA move] が伝わると，ステージパッドでは，そのイベントを [Pad at @Jiro move] と [Taro move] と解釈する．これは，PadA がスクリプト中において [Pad at @Jiro] [Taro] で参照されるためである．

4.3 スクリプトの実行

ステージパッド上でイベントが発生した場合，そのイベントは，パッドの貼り合わせのリンクを通じてステージパッドに伝えられる．ステージパッドは，送られてきたイベントをトリガとして実行されるべきスクリプトを探索する．実行すべきスクリプトを見つけた場合には，スクリプトの指示に従い，キャスティングリストを参照してオペレーションポイントを求め，これを通じてアクタパッドに指示を送る．アクタパッド

は指示に従い動作する．その動作もまたイベントとしてステージパッドに伝えられ，ステージパッドはそのイベントに対しても上記の処理を行わなければならない．しかし，前のトリガイベントに対する一連の動作がすべて終了するまで，次のイベントに対する処理を行えないため，イベントは一時的にキューに保持される．このキューは，ステージパッド上で発生したイベントの結果を時系列に並べて保持する．ステージパッドはキューに保持されているイベントに対して順番に（上記の）イベントに対する処理を行う．

図7において，イベントとして **mouse click** がステージパッドに伝えられたとする．ステージパッドは，それに対するスクリプトを実行し，アクタパッドは動作を完了する度，それをイベントとしてステージパッドに伝える．ステージパッドが **mouse click** に対するスクリプトの実行を完了した時，キューには **Taro move** と **Jiro move** がイベントとして保持されている．次に **Taro move** に対するスクリプトが実行され，キューは **Jiro move** と，**Jiro copy** をイベントとして保持した状態となる．これらのイベントに対してのスクリプトは記述されていないため，ステージパッドは何の指示も出さずキューの中からこれらを削除し動作を終了する．

4.4 スクリプティング機能

ステージパッドには，スクリプティング機能と呼ぶものがある．これは，パッドを操作することによりスクリプトを記述していくものである．例えば，**図9**においてPadAを動かした場合にはTaro move to x@y. というスクリプトが生成されエディタ画面に表示される．この機能を用いることで，ユーザは，実際にパッドを動かして，その動作を確かめながらスクリプトを作成していくことができる．また，この記録を見ることで，劇の実行中にどのようなイベントが発生しているかを理解できる．

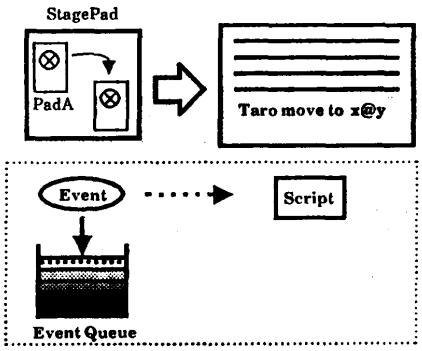

図9 スクリプティング機能
Fig. 9 The script mechanism.

5. 劇中劇

IntelligentPadシステムにおいては，パッドを部品として扱うことができる．プリミティブ・パッドおよびその合成によってつくられたアプリケーションパッドは，新しいパッドを構築するための部品として用いられる．ステージパッドも，このシステムのプリミティブ・パッドの一つである．これを用いて作られたアプリケーションパッドは劇を構成し，その劇は一つの部品として利用できる．ステージパッド上の各パッドは，それぞれの独立性を損なわれずに，ステージパッドに操られる．このため，ステージパッドによるアプリケーションパッドは，その独立性を保ったまま別のステージパッドを構築する部品として用いられる．つまり，ステージパッド上にステージパッドを貼ることで劇中劇を構成できる．

劇中劇とは文字どおり，大きな劇の中に小さな劇が存在するということである．小さな劇は，それのみで独立して劇を構成し，かつ大きな劇の構成要素ともなる．ステージパッドを用いて作られた劇はモジュール化されているため，それらの劇を合成することができる．この場合，劇の合成に伴うスクリプトの書き換えは行われない．それぞれのステージパッドにおいてスクリプトは独立して実行されるが，全体としては，あたかもスクリプトの合成が行われたのかのごとく劇が展開される．つまり，ステージパッドを合成することは，スクリプトの合成を行うことと同じである．

図10はStagePadB上にStagePadAを貼り，劇の合成を行ったものである．キャスティングリストは，それぞれ，図中のCastingListA，CastingListBで示される．この状態で，PadAの動作はStagePadAにおいてTaroおよびPad at @Jiroの動作として

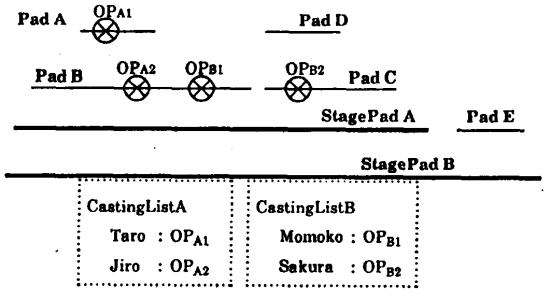

図10 ステージパッドの階層化
Fig. 10 A hierarchy of StagePads.

解釈され StagePadA にトリガをかける．また，PadB の動作は StagePadA においては Jiro の動作として StagePadA にトリガをかける．さらに，PadB の動作は StagePadB において Momoko の動作として解釈され StagePadB にトリガをかけることができる．小さな劇の中の役者は大きな劇の中で別の役割を演じることもできる．

5.1 劇中劇の例

図 11 は，7 枚のステージパッドを用いて作られたアプリケーション（『コン太くんの修行物語』）におけるステージパッドの階層を示している．使われているパッドの総数（ステージパッドを含む）は，112 枚である．個々に独立した劇を組み合わせてひとつの劇に仕上げている．各ステージパッドは，アクタパッドとして，アニメーションパッド，自分に貼られているパッドの数を値に持つパッド，足し算機能を持つパッド等，既存の様々なパッドを利用して劇を実現している（図 12）．これは，CAI のためのアプリケーションで，きつねのコン太くんが和尚さんの出す問題を解いていくというストーリーである．ストーリーを構成する各場面ごとにステージパッドを用いている．Stage2 では，きつねが男の子に化ける場面．Stage3 では，きつねが化けた男の子と和尚さんの問答の場面．Stage4 は，男の子の独り言の場面．Stage5 は，再び和尚さんとの問答の場面．Stage5 は，さらに Stage6，Stage7 の場面を含む構成となっている．Stage1 により，これらの各場面が制御されている．もし，このアプリケーションを，1 枚のステージパッドで作成しようとすると，数十個のトリガイベントに対するスクリプトを記述し，100 個以上のロールネームを把握し各パッドに割り振らなければならない．（図 11 の構成では，1 枚のステージパッド当たり，数個のトリガイベントに対するスクリプトと，高々 20 個あまりのロールネームを把握すればよい．）さらに，実行時には，必要なスクリプトを数十個の中から選び出し，100 個以上のロールネームを元にアクタパッドを捜し出さなければならない．図 11 の構成に比べ，オーサリングの負担は著しく増加し，実行速度もかなり遅くなる．また，図 11 の構成では，劇の展開を（ロールネームの割当

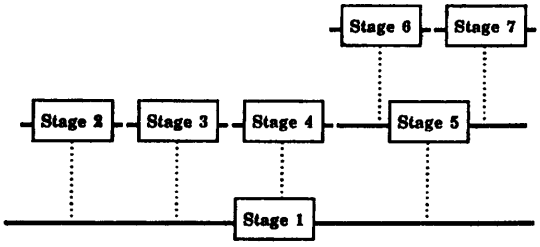

図 11 『コン太くんの修行物語』におけるステージパッドの階層
Fig. 11 A hierachy of StagePads.

図 12 コン太くんの修行物語のハードコピー
Fig. 12 Display hardcopies of a CAI application.

てやスクリプトの一部変更により）自由に変えられるのに対し，1枚のステージパッドで作成されたものを変更するのはかなり大変な作業となる．

6. 楽　　屋

複雑な動きを伴うアプリケーションを開発する場合には，様々なパッドを多数用いることになる．オーサリングを容易に行うためには，限られた画面上でこれらのパッドを効率良く扱うための工夫が必要である．また，劇の中の一場面においては，観客から見えている役者と見えていない役者がいる．見えていない役者の場合，見えていなくてもなんらかの働きをしているものと，なんの役にも立たないものがある．なんの働きもしていない役者は舞台上にいる必要はなく，それらの待避する場所が必要となる．このため，ステージパッドに楽屋を設けた．

楽屋にはドレッシングルームパッドを用いる．ドレッシングルームパッドには，複数のページ（パッド）があり，各ページにパッドを貼ることができる．ステージパッドで扱う予定のパッドを，各ページに貼っておくことで，パッドが重なっていたり，予期せず貼られてしまい見つからなくなるようなことが防げる．

また，アクタパッドには，劇の特定の場面のみに動作し，常にステージパッド上にある必要のないものもある．余計なパッドがあると，スペースの問題や，イベント数の増加の原因となる．このため，一時的に不必要になったパッドの待避場所としても楽屋を利用する．

アクタパッドをステージパッドから楽屋へ移動するには，マウスを用いて楽屋（DressingRoomPad）に貼りつけるか，スクリプトにおいてコマンドmoveの引数として'DressingRoom'または，'@DressingRoom'と記述して，スクリプトを実行する．

図13は，図11，図12で示したアプリケーションにおける楽屋の画面ハードコピーである．また，**表1**は，このアプリケーションを楽屋を用いて作成した場合と利用しなかった場合について，オペレーションポイント数やイベント数を比較したものである．Stage2, Stage3, Stage4においては，楽屋の有無による変化はほとんどない．しかし，Stage1, Stage5, Stage6, Stage7においては，オペレーションポイント数は減少し，伝達イベント数（各ステージパッドに伝わってきたイベントの数）も減少している．前者がアクタパッドの出入りの（ほとんど）ないステージであるのに対して，後者のステージでは，頻繁にアクタパッドが入れ替わっているために，これらの違いが生じる．オペレーションポイント数の減少は，アクタパッドの待避場所として利用していたオペレーションポイントが不要になったためである．また，伝達イベント数の減少は，楽屋内でのイベントがステージに伝わらないためである．基本的に，楽屋内でのイベントは，ステージパッドに伝わる必要はない．もし，伝える必要があるときには，そのパッドをステージ上に移すか，楽屋をステージパッドに貼ればよい．

楽屋の導入によりオーサリングの際の煩雑さを解消し，不必要なオペレーションポイントや伝達イベント

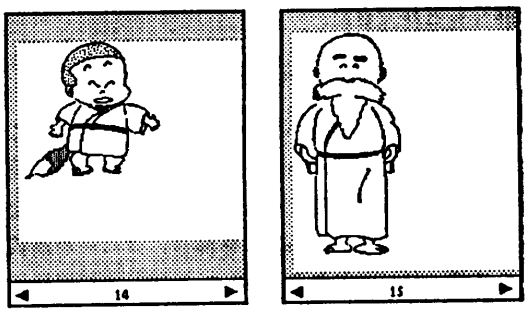

図13 ステージパッドの楽屋
Fig. 13 The DressingRoom for a StagePad.

表1 楽屋の有無によるOperationPoint数，イベント数の比較
Table 1 A reduction of operation points by the use of a DressingRoom.

	楽屋	Operation Point数	伝達イベント数	解釈イベント数	トリガイベント数
Stage1	有	9	271	79	3
	無	18	362	92	3
Stage2	有	4	33	37	1
	無	4	32	37	1
Stage3	有	28	47	55	3
	無	29	48	58	4
Stage4	有	3	13	11	1
	無	3	12	10	1
Stage5	有	34	178	251	6
	無	51	236	229	6
Stage6	有	9	42	55	2
	無	13	47	47	2
Stage7	有	9	42	55	2
	無	13	47	47	2

を除くことができた.

7. パッドの操作性改善への応用

ステージパッドを用いてパッドの操作性を変えこることができる. 例えば, パッドをコピーする場合の操作について考える. 通常, パッドをコピーする場合, ユーザはマウスでメニューを開き, コピーを選択する. ステージパッドを用いることで, この操作を, 『ダブルクリックによりコピーする』という操作に変更することができる. **図 14** におけるステージパッドのスクリプトには, 『'パッド' をダブルクリックされたら, 'パッド' をコピーする』と記述する. ロールネーム 'パッド' に割り振られたパッドは, ダブルクリックにより, そのコピーを作ることになる. プログラムを書き換えることなく, パッドのコピーに関する操作を変更することができる.

また, **図 15** のステージパッドのスクリプトには, 『'ボタン1'をクリックされたら, 'ボタン2'のクリック動作を二回行う.』と記述されている. 'ボタン2' をカウンタパッド上のアップボタン P_2 に, 'ボタン1' をアップボタンの上に貼られているボタンパッド P_1 に割り振る. アップボタンとその上のボタンを同寸にしておく. アップボタン P_2 は P_1 の下にあるため, ユーザからは見えない. このため, カウンタパッドにカウンタアップのコマンドを送るのは P_2 であるが, ユーザがアップボタンとして認識するのは P_1 である. この結果, 一度のマウスクリックにより, カウントを2ずつ増すカウンタを実現することができる.

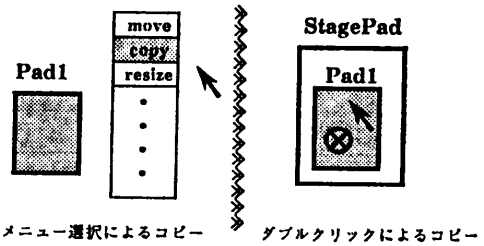

図 14 パッドのコピー操作の変更
Fig. 14 A modification of the copy operation.

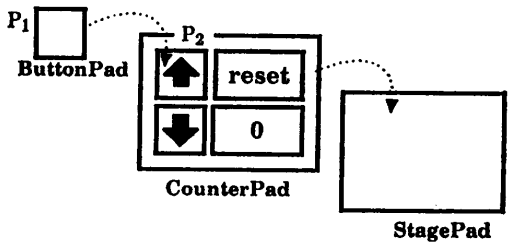

図 15 カウンタの操作変更
Fig. 15 A modification of the count operation.

これまで, パッドの操作を変更するには, そのパッドのプログラムを書き換えなければならなかった. しかし, ステージパッドを利用することにより, パッドの操作を変更することができる. アプリケーション開発の時に, その用途や, 利用者に合わせ操作性を自在に変更し容易に実現することができる.

8. おわりに

IntelligentPad システムに劇のメタファを取り入れるため, 新たにステージパッドを開発した. 本論文ではステージパッドの構成と動作機構について詳しく説明した.

ステージパッドは, スクリプトの指示をオペレーションポイントを通じてアクタパッドに伝達するという, 極めてシンプルでプリミティブな機能を持つにすぎない. しかし, キャスティングリストやオペレーションポイントの導入により, すべてが完全に部品化されている. このため, どのような部品 (パッド) をどのように用いるかによって様々な劇を展開できる. また, ステージパッドをフォームベース (同一フォーマットのパッドを管理するシステム) の一項目としたり, フィールドパッド[13] (操作場共有パッド) 等と組み合わせることで, 様々な機能を実現できる.

参考文献

1) ダニー・グッドマン (著), プロジェクトハウス (訳): *The Complete Hypercard Handbook Vol. 1*, (株) ビー・エヌ・エヌ (1988).
2) ダニー・グッドマン (著), プロジェクトハウス (訳): *The Complete Hypercard Handbook Vol. 2*, (株) ビー・エヌ・エヌ (1988).
3) 今滝隆元, 田中 譲: インテリジェント・パッドの開発, 第37情回報処理学会全国大会論文集, pp. 707-708 (1988).
4) 今滝隆元, 田中 譲: IntelligentPad におけるパッドの実現機構, 第 38 回情報処理学会全国大会論文集, pp. 1277-1278 (1989).
5) Tanaka, Y. and Imataki, T.: IntelligentPad: A Hypermedia System Allowing Functional Compositions of Active Media Objects through Direct Manipulations, *Proc. of the IFIP 11th World Computer Congress*, pp. 541-546, San Francisco (Aug. 1989).
6) Tanaka, Y.: A Toolkit System for the Synthesis and the Management of Active Media Objects, *Proc. 1st International Conference*

on Deductive and Object-Oriented Databases, Kyoto, pp. 259-277 (Dec. 1989).
7) 長崎 祥，今滝隆元，田中 譲：IntelligentPad の機構，第 40 回情報処理学会全国大会論文集，pp. 1146-1147 (1990).
8) Tanaka, Y.: A Synthetic Dynamic-Media System, *Proc. International Conference on Multimedia Information Systems*, Singapore, pp. 299-310 (Jan. 1991).
9) 赤石美奈，田中 譲：IntelligentPad における演劇機能の実現，第 42 回情報処理学会全国大会論文集，第 5 分冊，pp. 37-38 (1991).
10) Goldberg, A. and Robson, D.: *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley (1983).
11) 梅村恭司（著），竹内郁雄（監修）：Smalltalk-80 入門，ソフトウェアライブラリ=4，サイエンス社 (1986).
12) Finzer, W. and Gould, L.: Programming by Rehearsal, *Byte*, Vol. 9, No. 6, pp. 187-210 (1984).
13) 長崎 祥，田中 譲：IntelligentPad における協調作業場の実現，第 41 回情報処理学会全国大会論文集，第 5 分冊，pp. 90-91 (1990).

赤石 美奈（学生会員）

昭和 42 年生．平成 2 年北海道大学工学部電気工学科卒業．平成 4 年同大学院修士課程修了．現在，北海道大学大学院工学研究科博士後期課程電気工学専攻在学中．ヒューマンインタフェース，オブジェクト指向データベース等の研究に従事．ソフトウェア科学会会員．

田中 譲（正会員）

昭和 25 年生．昭和 47 年京都大学電気工学科卒業．昭和 49 年京都大学電子工学専攻修士課程修了．工学博士．現在，北海道大学電気工学科教授．データベースマシン，データベース理論，メディア・ベース，論理型プログラミング等の研究に従事．主たる著書，「コンピュータ・アーキテクチャ」（オーム社，共著）．IEEE，ソフトウェア科学会，人工知能学会各会員．