J-008

# テーブル型ディスプレイ上での協調作業を支援するための ユーザの離合集散に対応できるトレイの実現
# A Tray Capable of Responding to Meeting/Parting Activities of Users for Assisting Collaborative Work on a Tabletop Display

吉原 正樹† 荒木 博文† 中島 誠‡ 伊藤 哲郎‡
Masaki Yoshihara, Hirofumi Araki, Makoto Nakashima, and Tetsuro Ito

## 1. はじめに

単一ディスプレイを複数ユーザで共有しながら協調的に作業をするための環境整備に関し Single Display Groupware(SDG)の研究が進められている[4]．その中で，テーブル型ディスプレイ上においても実世界の机上での打ち合わせに似た形態が取れる作業支援方法が開発されている[2][5]．開発に際しては，複数ユーザが一箇所に集まって作業することが想定されており，支援としてはディスプレイ周りのユーザに対し共通の情報を見やすい形で提示することに重点が置かれている．

ところで，協調作業は，基本的には複数人で行うが，各自は時には離れ，必要に応じて声をかけながら進めるという，ユーザが離合集散しながら行う一連の作業と捉える方が自然である．すなわち，ユーザが集まり共通の情報を見て作業する場合も多いが，離れた位置にいるユーザ同士が情報を共有しながら作業をする場合もある．そこでは，ユーザが自身の作業状況を他に示したり，情報を一箇所に持ち寄って全体的にまとめたりする[3].

我々は，これまでに，テーブル型ディスプレイ上のアプリケーションをタッチ操作だけで扱う状況を想定し，上のような協調作業を支援する環境を整備してきた[1]．その過程で，Java のようなメッセージ通信を前提としたプログラミング言語によるアプリケーションに対し，これを上に載せるだけで

(1)アプリケーションウィンドウ（アプリケーションから起動されるウィンドウ）をユーザの望みに合わせて容易に移動・回転できる

仕組みを，トレイと名づけて定式化した．トレイは写像・イベント処理機構を有する．トレイ上に載せるとアプリケーションは，協調作業中の各ユーザが自身の見やすい方向に向けて参照できるようになる．

上記(1)は，協調作業支援のための必要な事柄であるが，これだけでは十分でない．ユーザ同士のよりきめの細かい情報共有のためには，加えて以下のような事柄も必要となる．

(2)アプリケーションウィンドウを容易に拡大・縮小できる
(3)アプリケーションウィンドウを容易に複製できる
(4)アプリケーション間で容易にデータ転送できる

本論文では，(1)に加えて，アプリケーション自体を大きく変更せずとも (2)，(3)および(4)も可能となるよう，[1]でのトレイを機能拡張する方法について述べる．このうち，(2)については，これまでの写像・イベント処理機構を改良し，トレイ自体の拡大・縮小に合わせてその上のウィンドウも拡大・縮小するようにする．(3)については，トレイに新しく複製処理機構を備える．(4)については，アプリケーション間のデータ転送を，アプリケーションとそれが載せられたトレイ間のデータ転送ならびにトレイ相互間のデータ転送とからなるとみなす．そして，公開通信メソッドとクリップボードを備え OS のような役割を果たす通信処理機構を備えて対処する．これらの機能はアプリケーションから独立しているため，トレイの上に載せられた任意のアプリケーションを決まった操作で扱えるようになる．

以下，ここでの方法の優位点を関連研究と比較して明らかにした後，機能拡張されたトレイの詳細ならびにトレイを使った特徴的な協調作業の例について述べる．

†大分大学大学院工学研究科知能情報システム工学専攻
‡大分大学工学部知能情報システム工学科

## 2. 関連研究との比較

テーブル型ディスプレイ上での協調作業を支援するシステム開発では，レガシーアプリケーションのような既存のアプリケーションに対して，(1)～(4)の実現が容易にできるかどうかがポイントになる[6]．また，実現された(1)～(4)がユーザにとってどれほど容易に操作できるかもポイントとなる．

代表的な SDG として，DiamondSpin[5]のように，テーブル型ディスプレイ上での利用を想定したアプリケーションを作るためのツールキットや，指向性のある透明スクリーンを用いてディスプレイの四方向からそれぞれ見やすい形で情報を提示するシステム Lumisight Table[2]がある．いずれも複数ユーザが一箇所に集まった状況での協調作業の支援を目指している．(1)～(4)を実現するには，前者では，個々のアプリケーションをそれらに合わせて改良する必要性があり，操作性もアプリケーションごとに異なる可能性がある．後者では，指向性スクリーンの利用を考慮に入れたアプリケーションの作成が必要である．操作性に関しては，テーブル周りでのユーザの位置が固定されているため，アプリケーションをうまく作成したとしても，おのずと制限が生まれる．

トレイを使うと，アプリケーションをその上に載せるだけで(1)～(4)が実現できる．これらの操作についても，アプリケーションと独立した形で，ユーザの位置にとらわれず行える．

## 3. トレイ

### 3.1 基本構造

トレイは，各アプリケーションウィンドウの描画イメージをディスプレイデバイス上の部分領域に表示するための仮想の場である．図 1 に示すように，アプリケーション空

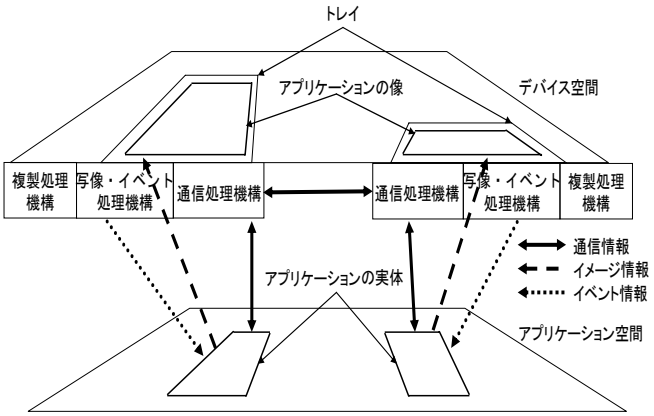



図 1:トレイの基本構造

間にある既存のアプリケーションをトレイの上に載せた形でデバイス空間に表示させ，(1)〜(4)が働くようにする．トレイは次の 3 つの機構を備えている．

(a)写像・イベント処理機構：アプリケーション空間にあるアプリケーションウィンドウの像の生成とデバイス空間のトレイ上でのイベント情報の取得を担う
(b)複製処理機構：トレイの複製と複製されたトレイ上でのイベント情報の取り扱いを担う
(c)通信処理機構：複数のアプリケーション間のデータ転送を担う

以下，各機構の詳細と実現法について述べる．

## 3.2 各機構の詳細と実現法

(a) 写像・イベント処理機構

写像・イベント処理機構は，アプリケーションをトレイ上に表示させる役割とトレイを介してアプリケーションを動作させる役割を持つ．前者は，アプリケーション空間のウィンドウの描画イメージを，ディスプレイデバイス空間にあるトレイ上の像にアフィン変換する．後者は，デバイス空間のトレイ上での（アプリケーションを動作させるための）クリックやドラッグ等のイベント情報に関し，イベントの発生座標を描画イメージの変換時のパラメータをもとに逆アフィン変換し，イベントの種類とともにアプリケーションに送る．この機構により，既存のアプリケーションを変更せずとも，ユーザはデバイス空間のアプリケーションを容易に移動・回転させられるようになる．

本機構の実現は，アプリケーションとディスプレイデバイス間のメッセージ通信に介入する形で行う．メッセージ通信を前提としたプログラミング言語によるアプリケーションではこれは容易である．Java の場合，描画イメージ用のバッファとイベント処理を担うリスナを取得することで，必要な情報を集められる．

拡大・縮小には，アフィン変換時に，アプリケーションの像をトレイのサイズに合わせて描画するようにすれば済む．トレイの拡大により，複数ユーザが同じ情報を見やすくなり，また，単一ユーザがアプリケーションの細かな部分を操作できるようになる．使用しないアプリケーションを載せたトレイを縮小しておけば，協調作業スペースの有効利用となる．

(b) 複製処理機構

複製処理機構は，トレイに既に像を表示しているアプリケーションの実体を，デバイス空間の別な（複数の）トレイに表示させる役割を持つ．複製元だけでなく複製先のトレイ上のアプリケーションも容易に再複製できる．

本機構は，ユーザの複製要求に応じて新規トレイを作成した後，複製元のトレイに表示しているアプリケーションの描画イメージ用バッファを取得し，新規トレイに渡す形で実現する．このとき，複製先のトレイでのイベント処理法を違えることで，2 種のトレイを複製できる．まず，イベント情報を取得しないようにすると，移動や回転は複製先のトレイでも行えるが，アプリケーションの操作は複製元だけとなる．複数ユーザが 1 つのアプリケーションを操作してしまうのを防ぐ場合に有効である．一方，イベント情報も取得するようにすると，単一のアプリケーションを複数人で協調して操作できるようなトレイができあがる．

(c) 通信処理機構

通信処理機構には，トレイとその上に載っているアプリケーションの間でのデータ転送とトレイ相互間でのデータ転送の 2 つの役割を持たせる．この機構によって，アプリケーション間の通信をトレイが媒介する形となり，既存のアプリケーションには大きな変更を施さずとも任意のアプリケーション間のデータ転送が可能になる．ユーザから見れば，データ転送についての特別な知識は必要なく，トレイ同士を接触させた後，通常のコピーやペースト操作をするだけで，望む転送が行われる．トレイを移動させればデータの転送先や転送元を容易に替えられ，2 つ以上のトレイと接触させればブロードキャスト的なデータ転送が可能となる．

本機構は，図 2 に示すように，トレイのクリップボード，クリップボードをアプリケーションや他のトレイから参照するためのトレイ通信インタフェース，および他のトレイへデータを送るためのデータ転送部からなる．

トレイのクリップボードは，文字列や画像等のデータを一時保存するためのトレイ自身が保有する領域である．システムクリップボードによるオブジェクト情報の転送を前提にして作られたアプリケーションに対し，システムクリップボードの代わりにこのクリップボードを扱うよう変更を加えると，トレイを介したデータ転送が可能になる．

トレイ通信インタフェースは，アプリケーションからトレイのクリップボードへのアクセス手段を定める公開通信メソッドに関する記述である．図 3 上部に Java で記された

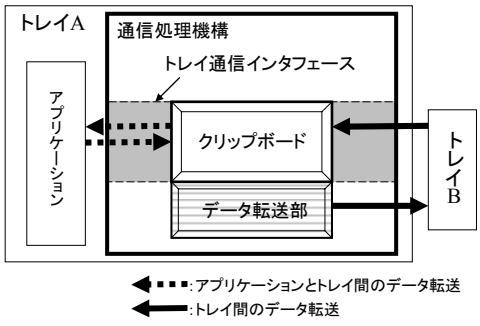



図 2: 通信処理機構

```
// トレイ通信インタフェース
public interface TrayCommunicationInterface {
   public Clipboard getTrayClipboard() ;
}

// トレイを実現するプログラムでの実装例
public Clipboard trayClipboad ; // メンバ宣言
public Clipboard getTrayClipboard() { // メソッドの実装
        return this.trayClipboard ;
}
```

図 3: トレイ通信インタフェースとメソッド実装

トレイ通信インタフェースの例 TrayCommunicationInterface を示す．アプリケーションをトレイ上に載せる際に，この通信メソッド getTrayClipboard()を参照して，データ転送に利用するシステムクリップボードをトレイのクリップボードに変更すればよい．

データ転送部は，デバイス空間で当該トレイと他トレイとが接触しているとき，トレイのクリップボードの内容に変化が生じたかどうかを監視していて，変化が生じた時点で他トレイのクリップボードへデータを転送するプログラムである．

Java で書かれたアプリケーション A と B がそれぞれトレイ A と B に載せられた場合を考える．アプリケーション A と B のシステムクリップボードについての記述

```
public Clipboard clp = getToolkit().getSystemClipboard();
```

を，トレイ通信インタフェースを参照して，次のように書き直しておく（tray は，アプリケーションが載っているトレイの実体を指すためにセットする）．

```
public Clipboard clp = tray.getTrayClpboard();
```

トレイ A が B と接触させられた状況で，アプリケーション A で転送するデータがコピーされると，それはトレイ A のクリップボードに渡され，続いてトレイ A のデータ転送部からトレイ B のクリップボードに送られる．アプリケーション B でペースト操作が実行されると，トレイ B のクリップボードからその内容がアプリケーション B に渡される．

アプリケーション間のデータ転送は，通常，システムクリップボードのような OS が管理するデータ保存領域を用いて行われる．この仕組みを協調作業支援のような複数のアプリケーションが同時に動作する環境でのデータ転送に採用すると，保存領域が 1 つしか確保されていない場合，一時保存されているデータが別のデータ転送要求によって壊されてしまう．保存領域が複数確保されていると，今度は，どのデータをどのアプリケーションに送るかを統一的に管理する仕組みを用意しなければならない．ここでの仕方によると全体として複数の保存領域が確保された形になるが，データの転送先はトレイの接触状況で明示されるため，システムクリップボードを使った場合のような問題は生じない．

## 4. 利用例

トレイの 3 つの利用例を図 4 に示す．トレイは，外見上，一般のウィンドウのような形態をしている．周辺に(1)〜(4)の操作のためのアイコンが，また，中央にアプリケーションの像が載せられている．各アプリケーションはタッチ操作で動作させる．

(a)エディタ内容の提示

あるユーザが用いている（Java で書かれた）エディタの内容を，別な 2 人のユーザに提示することを考える．図 4(a)は，上側のユーザのエディタを載せたトレイ C をトレイ C’として複製・拡大した後，それを下側左右のユーザに向けて回転させ提示している．下側のユーザおのおのは，各自のトレイ D，E を縮小して，スペースを開けている．複製元 C 上のエディタで，ユーザが文字列を選択（白黒反転）すると，C’上のエディタで対応する文字列も同様な状態になっている．

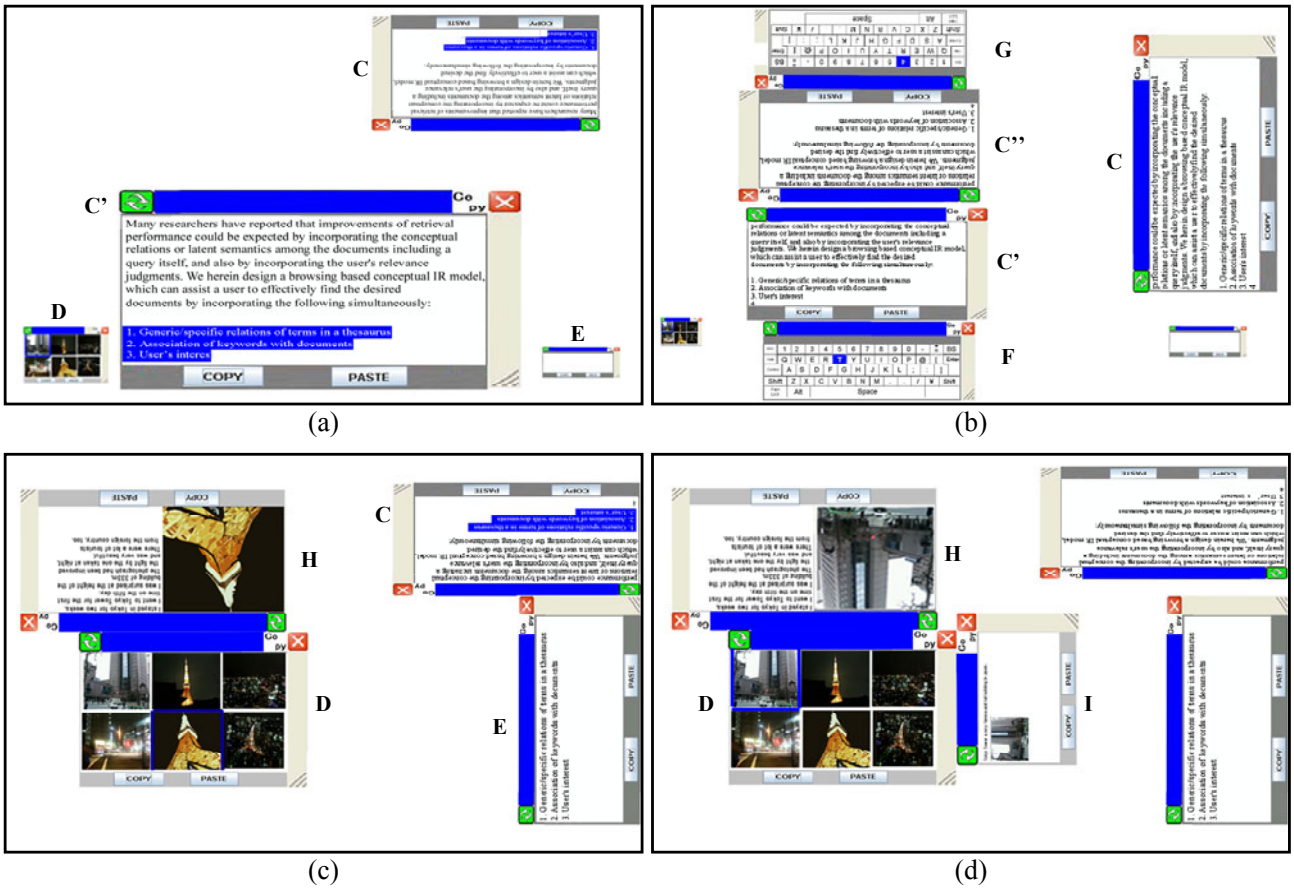
(b)文書の協調編集

エディタの載せられたトレイをイベント情報も取得するように複製して，複数人で協調して文書を編集することを考える．ここでは，アプリケーションへの文字入力を，トレイに載せたソフトウェアキーボードからのデータ転送として扱っている．複数の入力装置の利用が可能となる．図 4(b)に 2 人のユーザが協調して文書を編集している状況を示す．エディタが載せられたトレイ C の 2 つの複製 C’と C’’に，左下側のユーザと上側のユーザが，それぞれのソフトウェアキーボードを載せたトレイ F と G を接触させて，相手の入力状況を見ながら同一文書を編集している．一方で，右側のユーザは，C’と C’’の複製元である C を自分に向け，編集状況を確認している．

(c)データ転送

4 人のユーザそれぞれが文書，画像等のオブジェクトを持ち寄って，それらの一部をやり取りすることを考える．エディタやイメージビューアをそれぞれトレイの上に載せるだけで，これらの間のデータ転送が可能になる．図 4(c)は，左側の 2 人が使用しているエディタあるいはイメージビューアを載せたそれぞれ 2 つのトレイ H と D とを接触させ，下側のユーザが D に載せられたイメージビューアの画像を選択しコピーボタンを押下している状況である．上側のユーザが H に載せられたエディタでペーストボタンを押下すると，選択された画像がカーソル位置に転送される．同時に右側では， 2 人がそれぞれ使用しているエディタを載せたトレイ C と E とを接触させて，C のエディタから E のエディタへ文字列の転送がなされている．

図 4(d)では，右側のユーザが新たにトレイ I を D に接触させている．トレイを使うと，複数のアプリケーションへのデータ転送が可能で，D で新たに選ばれた画像が I と H に転送されている．




(a) (b) (c) (d)

図 4: 利用例

## 5. おわりに

テーブル型ディスプレイを使った環境での協調作業を支援するためのトレイについて述べた．協調作業では，レガシーアプリケーションを含む種々のアプリケーションを動作させ，全員で参照したり互いが情報を交換したりしながら目的を達成する．既存の協調作業支援に関わる手法では，テーブル型ディスプレイ上での利用を想定してアプリケーションを改良したりその見せ方を工夫したりしなければならなかった．ここでのトレイを使えば，その上にアプリケーションを載せた後はトレイを操作するだけで，アプリケーションの所在場所にとらわれない自由な情報提示や，複数のアプリケーション間でのデータ転送が簡単に実現できるようになる．このことは，ユーザにとって使い慣れたアプリケーションで協調作業を行えるということにも繋がる．

協調作業では，単一ディスプレイだけでなく，複数のディスプレイを利用する場合もある．また，同一部屋内の作業だけでなく，ネットワークを介した遠隔環境での作業もある．今後は，このような状況でも利用可能なトレイに機能拡張したい．

## 参考文献

[1] 荒木 博文, 柏本 正昭, 中島 誠, 伊藤 哲郎，“協調作業でのユーザの離合集散に対応できるトレイを備えたテーブル型ディスプレイ”，情報処理学会インタラクション 2006 論文集, pp.63-64．2006.

[2] Kakehi, Y., Iida, M., Naemura, T. Shirai, Y., Matsushita, M. and Ohguro, T., “Lumisight Table: Interactive view-dependent tabletop display surrounded by mutiple users,” IEEE CG&A, vol. 25, no.1, pp 48 -53, 2005.

[3] Scott, S., Grant, K. and Mandryk, R., “System guidelines for co-located collaborative work on a table top display,” Proc. ECSCW, pp.129-178, 2003.

[4] Scott, S. and Sheelagh, C. “Interacting with digital tabletops,” IEEE CG&A, vol. 26, no. 5, pp. 24-27, 2006.

[5] Shen, C., Vernier, F. D., Forlines, C. and Ringel, M., “DiamondSpin: An extensible toolkit for around-the-table interaction,” Proc. CHI 2004, pp. 167-174, 2004.

[6] Shen, C., Vernier, F. D., Everitt, K., Wu, M. Wigdor, D., Morris, M. F., Hancock, M. and Tse, E, “Informing the design of direct-touch tabletops,” IEEE CG&A, vol. 26, no. 5, pp. 36-46, 2006.