(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2010-49629
(P2010-49629A)
(43)公開日　平成22年3月4日(2010.3.4)

(51)Int.Cl.	F I	テーマコード（参考）
G06F 3/048 (2006.01)	G06F 3/048 655A	5E501

審査請求 有　請求項の数 1　ＯＬ　（全 13 頁）

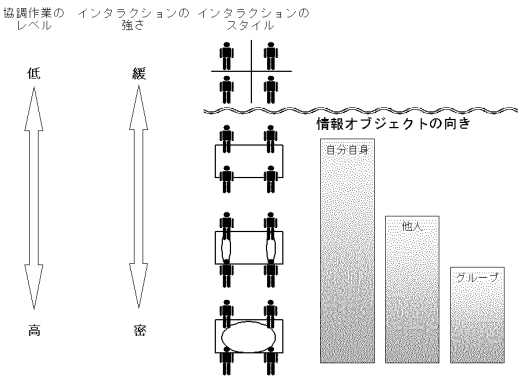
(21)出願番号　特願2008-215360 (P2008-215360)
(22)出願日　平成20年8月25日 (2008.8.25)

(71)出願人　304028726
国立大学法人　大分大学
大分県大分市大字旦野原７００番地
(72)発明者　中島　誠
大分県大分市大字旦野原７００番地国立大学法人大分大学内
(72)発明者　伊藤哲郎
大分県大分市大字旦野原７００番地国立大学法人大分大学内
(72)発明者　佐藤慶三
大分県大分市大字旦野原７００番地国立大学法人大分大学内
(72)発明者　荒木博文
大分県大分市大字旦野原７００番地国立大学法人大分大学内

最終頁に続く

(54)【発明の名称】コンピュータのコラボトレイ

(57)【要約】　（修正有）
【課題】ユーザによる改変なしに，テーブル型ディスプレイ上での協調作業に利用できる機能を付加するアプリケーションプラットフォームの提供。
【解決手段】ウィンドウアプリケーションごとに，ユーザとウィンドウアプリケーションの間のイメージとイベントのやり取りに介入し，ディスプレイ上でのイベント収集を担うイメージ/イベントハンドラと，ウィンドウアプリケーションを制御するアプリケーションハンドラを分離し、分離したイメージ/イベントハンドラとアプリケーションハンドラは，テキストおよびイメージのメッセージストリームで結合し，アプリケーションハンドラとのストリームを共有する、イメージ/イベントハンドラは、ユーザの要求に応じて任意の向き，位置，大きさに画像処理変換し、本来有しないウィンドウ操作の機能（回転や拡大）を，描画したイメージ操作機能として，協調作業のための機能を付加してなるコンピュータのコラボトレイ。
【選択図】図１



【特許請求の範囲】
【請求項１】
ウィンドウアプリケーションごとに，ユーザとウィンドウアプリケーションの間のイメージとイベントのやり取りに介入し，ディスプレイ上でのウィンドウイメージの描画およびイベント収集を担うイメージ/イベントハンドラと，ウィンドウアプリケーションを制御するアプリケーションハンドラを分離し、分離したイメージ/イベントハンドラとアプリケーションハンドラは，テキストおよびイメージのメッセージストリームで結合し，アプリケーションハンドラとのストリームを共有し、イメージ/イベントハンドラは、描画するイメージを，ユーザの要求に応じて任意の向き，位置，大きさに画像処理変換し、このイメージの変換に合わせてイベント情報をイメージ/イベントハンドラからプリケーションハンドラを介して当該ウィンドウアプリケーションへ送り、そのウィンドウアプリケーションでの同じイベント操作に加え，協調作業のための機能を付加してなるコンピュータのコラボトレイ。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、テーブル型ディスプレイ上で個人作業と協調作業の混在する日常作業の支援を可能にするコンピュータのコラボトレイに関するものである。
【背景技術】
【０００２】
日常作業において人々は，個人作業と協調作業の間や様々な協調作業の形態の間での移行を繰り返しながら作業を行う。例えば，個人はあるアイデアをグループに提示する前に，個人で作業を行い，その後，グループで集まってそれぞれのアイデアを持ち寄って協調して作業を行う(Tang et al。 2006)。これらの移行はグループメンバの他へ依存の度合いに応じた協調のレベルの変化に従って生じ，歴史的にテーブルの周りに集まって行われる会議や討論といった協調作業でもよく見られる。各メンバが長く個人で作業が行っている場合，協調レベルは低く，反対にメンバ間のインタラクションがなければ作業がすすまない状況では，協調レベルは高いと言える(Salvador et al。，1996)。
計算機においても然りで、テーブル型ディスプレイを中心とした作業環境の構築に関して，多くの技術がCSCW (Computer Supported Cooperative Work) の分野で提案されてきたが(Scott et al。，2003; Wigdor et al。，2006)，上記の移行の考慮は深くは行われてこなかった(Tang et al。 2006)。 テーブル型ディスプレイの最近の普及に伴って，このような考慮は避けられない基本的問題となっている。
これまで，計算機による協調作業支援では，テーブル型ディスプレイの利用が注目されている。普段の作業は，デスクトップディスプレイ上での個人利用を想定しているウィンドウアプリケーションを使用するのが一般的であり，これらは，例えば，ウィンドウの向きを回転させる，アプリケーションを共有するなど協調作業での利用を想定した機能を有しない。普段の作業は，個人の作業と複数人での協調作業の繰り返しであり，普段使い慣れたアプリケーションの利用が出来ないことは不便さを伴う。
テーブル型ディスプレイ上での協調作業用の専用アプリケーションの開発も従来から行われているが，主に個人で作業したり，特定人と情報交換あるいはグループ全体で作業したりといった協調作業におけるユーザ間での情報共有のスタイルの違いを考慮した機能を有しない。
＜参考公報＞
【特許文献１】特開2006-350997号公報
【特許文献２】特開号2005-266880公報
【特許文献３】特開号2003-323386公報
【発明の開示】
【発明が解決しようとする課題】
【０００３】

本発明は、従来の問題を解決するために，個人利用を想定したアプリケーションをユーザによる改変なしに，その本来の機能を損なわず，テーブル型ディスプレイ上での協調作業に利用できる機能を付加したコンピュータのコラボトレイを提供する。

【課題を解決するための手段】

【０００４】

上記目的を達成するための本発明のコンピュータのコラボトレイの技術的特徴は、「ウィンドウアプリケーションごとに，ユーザとウィンドウアプリケーションの間のイメージとイベントのやり取りに介入し，ディスプレイ上でのウィンドウイメージの描画およびイベント収集を担うイメージ/イベントハンドラと，ウィンドウアプリケーションを制御するアプリケーションハンドラを分離し、分離したイメージ/イベントハンドラとアプリケーションハンドラは，テキストおよびイメージのメッセージストリームで結合し，アプリケーションハンドラとのストリームを共有し、イメージ/イベントハンドラは、描画するイメージを，ユーザの要求に応じて任意の向き，位置，大きさに画像処理変換し、このイメージの変換に合わせてイベント情報をイメージ/イベントハンドラからプリケーションハンドラを介して当該ウィンドウアプリケーションへ送り、そのウィンドウアプリケーションでの同じイベント操作に加え，協調作業のための機能を付加してなるコンピュータのコラボトレイ。」である。

【発明の効果】

【０００５】

本発明の前記構成のコンピュータのコラボトレイは、協調作業で必要とされる機能を既存のウィンドウアプリケーションにそのまま付加することができ，さまざまな情報共有のスタイルにあわせてこれらアプリケーションを利用することができるなどの効果を呈する。例えば，既存のウィンドウアプリケーションをテーブル型ディスプレイ上で，複数人それぞれが見やすいように，回転・拡大・縮小して閲覧・操作できる。

イメージ描画およびイベント収集を担うモジュールに短期記憶を付加することで，通常のコピーアンドペーストのような特定のアプリケーション間での情報のやり取りが可能となる。

テーブル型ディスプレイ上で同一のウィンドウアプリケーションを数箇所から閲覧・操作し，共有することができる。

複数のディスプレイで同一のウィンドウアプリケーションを閲覧・操作し，共有することができ，個人作業と協調作業の繰り返しをスムーズに行なえる。

【発明を実施するための最良の形態】

【０００６】

１．　はじめに

本発明のコンピュータのコラボトレイは、異なる協調レベル間のスムーズな移行を，レガシーアプリケーションを容易に協調作業に利用できるようにし，テキストや図といった情報オブジェクトを用いた対面での議論を，言葉やジェスチャでのコミュニケーションと同様に可能にして，インタラクションを強化することで支援するものである。

インタラクションは，言葉やそれを補助する手振り身振りを解して行われる対面での議論などでよく見られるような意図のコミュニケーションを必要とする (Pinelle et al。，2003)。１つのコラボトレイは，テーブル型ディスプレイ上で丸い部分領域として認識され，(a)の機能を有する。

(a) レガシーアプリケーションを載せる

様々なアプリケーションを載せた多くのコラボトレイでテーブル型ディスプレイ上に作業環境を構築できる。この点は，あらかじめ決められた利用シナリオに合うようにアプリケーションを開発してきた既存の研究と大きく異なる。また，コラボトレイは，次の機能により，コミュニケーションメディアとしての役割も有する。

(b) アプリケーションウィンドウの回転，移動，拡大

(c) 情報オブジェクトの配信

(d) アプリケーションとその内容の共有

これらの機能は，ユーザが情報オブジェクトの操作により様々なインタラクションのスタイルを取ることを可能にする。機能（b)により，任意のユーザが自分自身あるいは他に情報オブジェクトを向けることが可能となる。機能（c）は，ユーザが特定のユーザとの間で情報オブジェクトをやり取りすることを可能にする。機能（d）は，アプリケーションをその情報オブジェクトと共に共有することを可能にする。これは，グループ内のユーザ間で同時にインタラクションを行う際に用いられる。

(a)の能力と(b),(c),(d)の機能は，ユーザとアプリケーションの間でのイメージとイベントの流れの間に介在する２つのハンドラ，すなわち，イメージ/イベントハンドラとアプリケーションハンドラからなる分離アーキテクチャにより，アプリケーションとは独立に実現される。前者は，アプリケーションのウィンドウイメージを描画し，その上でのイベントを取得する。後者は，前者からイベント情報を受け取り，アプリケーションを構成するコンポーネントへ配信する。
次に
(イ),異なる協調のレベルにおける日常作業を支援するための必要事項
(ロ),コラボトレイのアーキテクチャデザインとそのJava（登録商標）言語による実装
について述べる。
発明者等は，インタラクションの強化による，コラボトレイの異なる協調レベルへの適合可能性を調べるために，例として３つの作業環境を構築した。
レガシーアプリケーションをそのまま利用できることは，簡単にフレキシブルな作業環境を構築するのに有効であった。使いやすさについては，種々のユーザスタディで確認した。そこでのアンケート結果から，コラボトレイがテーブル型ディスプレイ上でのインタラクションをとるのに容易に用いることができることがわかった。
【０００７】
２．本発明のコンピュータのコラボトレイの開発の動機付けに関わる必要事項
＜協調作業のレベル＞
協調作業のレベルは，図１に示すようにインタラクションのスタイルに応じて，低から高の間で連続的に変化する。個人作業では，ユーザは個人の作業を個人用作業環境で行う。テーブル型ディスプレイのある協調作業環境に移動することになれば，図で波線で示したようなバリアを超えること，すなわち，レガシーアプリケーションから協調作業のためのアプリケーションへの切り替えが必要となる。ユーザは集まって，インタラクションを強化することにより作業を進めるために情報を交換する。ユーザがテーブル周りに集まった段階では，情報は個人で所有され，それぞれの作業がし易いように利用されるのみであり，インタラクションは緩いと言える。より多くのユーザが集まり，多くの情報が交換されるようになると，インタラクションは密になっていく。特定の他のユーザに向けての話やジェスチャがそれぞれの作業を達成するために必要となってくる。話やジェスチャがグループ全体に向けられるようになると（すなわち，インタラクションがさらに強化されると），情報は，グループメンバによって同時に共有され，インタラクションはより密となる。
会話やジェスチャによるインタラクションはテーブル型ディスプレイの周りでの対面のコミュニケーションにおいて自然に実現される。これらのインタラクションの強化には次の２つが必要となる。
(1)、個人作業環境から協調作業環境へレガシーアプリケーションを容易に移せること。
レガシーアプリケーションは，個人作業で用いられる際と同様に，その変更なく元々の機能を損なわずに利用できる必要がある。個人作業環境から協調作業環境への移行は，協調作業用のアプリケーションとして用いられるとしても，データの再調整を必要とせず，「これをそこで使いたい」という簡単な要求に応えられるべきである。
(2)、会話やジェスチャと同様に情報オブジェクトを用いたコミュニケーションを支援できる。
会話やジェスチャによるインタラクションと同様な情報オブジェクトによるインタラクシ

ョンがテーブル周りで求められる。Kruger 他(2004) は通常のテーブルでの作業におけるコミュニケーションでユーザがどのようにオブジェクトの向きを利用しているかについて示唆している。そこでは，タイプ1：自分自身へ，タイプ2：特定の他のユーザへ，そしてタイプ3：グループへ，といった３つの向きのタイプがある。テーブル型ディスプレイ上でタイプ１を実現するには，ユーザがアプリケーションのウィンドウを正しい位置と向きに調整することができねばならない。タイプ２では，コミュニケーションをとる相手を特定できかつ情報オブジェクトをその相手との間でやり取りする必要がある。タイプ３は，アプリケーションとその情報オブジェクトをディスプレイ周りのユーザと共有する必要がある。

【０００８】

次に、コラボトレイをデザインするに際し，容易にレガシーアプリケーションを載せるために，ユーザとアプリケーションの間に介在する２つのモジュールを用意する。３つのタイプのオブジェクトの向け方は，上のモジュールの分離アーキテクチャの利用とコラボトレイの機能である，アプリケーションウィンドウの回転/移動/拡大の機能，情報オブジェクトの配信機能，そして，アプリケーションとその内容の共有機能の実装により実現される。デザイン（実施の最良の形態）を３で詳細を紹介する。

図2(a) は大学の研究室での作業環境の一例を示している。図において，立っている学生が自身のコンピュータにあるプレゼンテーションスライドを垂直型ディスプレイに示し，他のメンバがその内容を参照している。ここまでは，通常の日常作業で見られる場面である。プレゼンテーションツールを載せたコラボトレイを複製することで，このツールとその内容はメンバ間で共有できている。また，図2(b)に示すように，メンバはコラボトレイを回転，移動，拡大してそこに載せられたアプリケーションに容易にアクセスするようにできる。さらには，テーブル型ディスプレイの周りの各メンバは，各自が有するデータやプレゼンテーションに関係する文書をもってきて，コラボトレイを接触させることで，他のメンバと情報オブジェクトのやり取りによるインタラクションを行うことができる。任意のメンバが，オリジナルや複製のコラボトレイを介して情報オブジェクトを操作でき，それぞれのスライドの変更に関するそれぞれの意図を直接他のメンバに明らかにすることができる。

【０００９】

３.本発明のコンピュータのコラボトレイを実施するための最良の形態（以下単にデザインと称する）

当該コラボトレイのデザインにおいて，イベント駆動プログラミングパラダイムによる既存のウィンドウアプリケーションをレガシーアプリケーションとして扱う。このパラダイムはユーザの入力により引き起こされるイベントを処理するGUI関数を用いるアプリケーションの開発において広く用いられている。

図3は，コラボトレイのアーキテクチャを示している。ここで，起動部は特定のアプリケーションを載せたコラボトレイを起動する。起動部はまた，起動した全てのコラボトレイをそれらの状態をチェックしながら監視する。

【００１０】

コラボトレイのデザインのポイントは，ユーザとアプリケーションの間のイメージとイベントのやり取りの間に介入するメカニズム(Intervening Mechanism)を構築することであり，コラボトレイが１つのウィンドウシステムとしての役割を担うことでアプリケーションを容易に載せられることにある。

コラボトレイ'のデザインにおける２つ目のポイントは，コラボトレイの機能を容易に実現する２つのハンドラの分離メカニズム（Decoupled Mechanism）にある。

この２つのハンドラは，アプリケーションのウィンドウイメージの描画とアプリケーションへのイベントを処理するイメージ/イベントハンドラと，アプリケーション自身とそのコンポーネントを制御するアプリケーションハンドラである。

【００１１】

而して、前記イメージ/イベントハンドラにおいて、そのイメージ変換部は，アプリケー

ションハンドラから渡されるアプリケーションのウィンドウイメージをアフィン変換により変換し，イメージ表示部に送る。イメージ表示部は変換されたアプリケーションのウィンドウイメージとトレイのユーザインタフェースのためのイメージをトレイ上に描画する。変換されたイメージ上で発生する全てのイベントの座標は，逆アフィン変換により，アプリケーションウィンドウの対応する座標に変換される。変換された座標とイベントの種類はアプリケーションハンドラを介してアプリケーションに送られる。短期記憶部は他のアプリケーションに渡される情報オブジェクトを格納する。イメージ送信部とイメージ受信部はネットワーク越しに渡される情報を管理する。

【００１２】

各アプリケーションは通常，様々なイベントを処理するために，複数のコンポーネントから構成される。アプリケーションハンドラは、これらの情報を格納するアプリケーション情報格納部とアプリケーション走査部を用意する。アプリケーション走査部は，アプリケーションウィンドウ上でのコンポーネントのイメージと，その位置およびこれらコンポーネントの従属関係を，ウィンドウシステムのインタフェースプログラム，例えばApplication Program Interface(API)，を介して取得し，これらイメージをまとめてイメージ格納部に格納し，まとめたイメージを前記イメージ/イベントハンドラに渡す。イメージ格納部のイメージは，アプリケーションの更新に応じて自動的に更新される。アプリケーション走査部は，さらにコンポーネントのイベント処理ルーチン（例えば，Javaにおけるイベントリスナや，C++におけるイベントハンドラ）を取得する。 コンポーネントのイメージの位置と従属関係およびこれらルーチンはアプリケーション情報格納部に格納される。イベント分配部は，アプリケーション情報格納部を参照しながら，アプリケーションのウィンドウイメージ上でのイベント情報を，アプリケーションの対応するコンポーネントに配信する。イベント送信部とイベント受信部は，イメージ送信部とイメージ受信部と同様な役割を担う。

【００１３】

次に、アプリケーションのウィンドウイメージの回転，移動，拡大，情報オブジェクトの配信，およびアプリケーションとその内容の共有のための機能の実現（Formalizing Functions）について述べる。

【００１４】

＜回転，移動，拡大機能＞:

ユーザの要求に応じて，イメージ表示部上のイメージに対するアフィン変換が，アプリケーションのウィンドウイメージを回転，移動，拡大するために行われる。元のアプリケーション自身のウィンドウの見易さ操作性は保持される。

アプリケーションの開発者は，様々なプログラミング言語や協調作業のために開発されたツールキット，例えばDiamondSpin (Shen et al。, 2004)，を用いてアプリケーションのウィンドウを回転させられるように作成することができる。これらのツールキットを使用して，回転のできるアプリケーションを作成するさい，オリジナルのアプリケーションは変更を余儀なくされる。しかしながら，コラボトレイでは，これにアプリケーションを載せるだけで，アプリケーションは回転させる機能を実現できる。アプリケーションウィンドウの移動とその一部の拡大は通常のオペレーティングシステムでの普通の操作であるが，コラボトレイ 上では，移動機能は回転機能と合わせて行え，拡大機能は，ウィンドウ全体に対して有効である。

【００１５】

＜配信機能＞

配信機能は，ある特定のユーザと情報オブジェクトの交換を通じてインタラクションするユーザのためのものである。分離と介入のメカニズムからなるアーキテクチャでは，コラボトレイの短期記憶部とシステムのクリックボードの間での情報スイッチバックのアイデアにより，アプリケーションを修正することなくこの機能を実現できる。配信先は，コラボトレイをインタラクションを取るユーザが有している他のコラボトレイと接触させることで特定できる。

通常のオペレーティングシステム上では，情報オブジェクトの配信は，システムのクリップボードを短期記憶として用いる，コピーアンドペースト操作によって行われることが多い。しかしながら，この操作は，一人のユーザの操作を想定したものである。テーブル型ディスプレイ上での協調作業においては，１人以上のユーザが多くのアプリケーションを使うと想定でき，複数のユーザが同時に情報オブジェクトをコピーしようとすると，その情報オブジェクトとシステムのクリップボードの間での衝突が生じる。この衝突を回避し配信機能を実現するため，イメージ/イベントハンドラは短期記憶部の内容をアプリケーション上でイベントが処理される前にシステムのクリップボードに送る。その処理が終わると，イメージ/イベントハンドラは，システムのクリップボードからその内容を取り戻す。図4 は，視覚的にこの情報スイッチバックのメカニズムを示している。
ここで，コラボトレイAの情報オブジェクトが，ユーザがコラボトレイAをコラボトレイBに接触させて配信しようとしている状況を示している。
任意のイベントがイメージ/イベントハンドラＡに来たとき,短期記憶部Ａに現在保持されている内容はシステムクリップボードに送られ，そしてイベント情報はアプリケーションハンドラＡを介して，アプリケーションAに配信される。このイベントの処理の終了時に，イメージ/イベントハンドラＡは，システムのクリップボードの内容を取り出し，それを短期記憶部Ａに戻す。もし，短期記憶部Ａの内容がイベント処理のあとで変化していれば（このことは，アプリケーションＡでのこのイベントがコピー操作によるものだったことを意味する），この内容は，短期記憶部Ｂに送られる。コラボトレイ Bでも，コラボトレイ Aと同様，全てのイベントに対して情報スイッチバックを行う。もし，コラボトレイ B上で，ユーザがペースト操作についてのイベントを起こせば，短期記憶部Ｂの内容は，アプリケーションＢにシステムのクリップボードを介して送られ，結果的にアプリケーションＡでコピーされた情報オブジェクトがアプリケーションＢにペーストされることになる。
情報スイッチバックは，複数のユーザが同時に配信機能を利用することを可能にする。コピーアンドペースト以外の操作でも情報スイッチバックは行われるが，これらはシステムクリップボードを参照しないため，その実行に情報スイッチバックが影響を及ぼすことはない。

【００１６】

＜共有機能＞

グループメンバ内でのインタラクションは，アプリケーションとその内容を共有することで強化できる。分離，介在メカニズムからなるアーキテクチャは，この共有を，図5に示すようにオリジナルのコラボトレイをクローニングすることで容易に行うことに寄与している。複数のユーザは同じアプリケーションをオリジナルとクローンのコラボトレイを介して扱うことができる。オリジナルのコラボトレイのクローンをディスプレイXに繋がったコンピュータ上で作った際，イメージ/イベントハンドラ１のコピー，だけが作成される。 アプリケーションハンドラ1はダミーであり，イメージ/イベントハンドラ１は，オリジナルのコラボトレイのアプリケーションハンドラ を参照する。オリジナルのコラボトレイとクローンコラボトレイとのイベントストリームは，統合され，これらコラボトレイ上の様々なイベントはアプリケーションハンドラに到達した時間順に処理され，アプリケーションは容易に同期処理される。
オリジナルのコラボトレイのクローンがディスプレイYにつながれた別なコンピュータ上でネットワーク越しに作成されるとき，クローンコラボトレイ2は，イメージの受信とイベントの送信機構を除いて，図5に示すようにクローンコラボトレイ1と同様に作成される。 オリジナルのイメージ/イベントハンドラのイメージ送信部とアプリケーションハンドラのイベント受信部は，クローンコラボトレイ 2が生成された後，起動される。イメージ/イベントハンドラは，重量モジュールであるアプリケーションハンドラと緩く結合された軽量モジュールであり，別なコンピュータのユーザは，自由に協調作業に参加することができる。

【００１７】

４．実装（Javaを使った実装についての詳細説明）
ここでは，図3に示すコラボトレイ実施例の構成概略図において、起動部，イメージ/イベントハンドラ，アプリケーションハンドラの実装を，Java言語でのコラボトレイのプロトタイプ作成に関して述べる。コラボトレイに載せるレガシーアプリケーションは，Java言語でSwingを用いたアプリケーションとしている。図6は，ソフトウェアデザインのための記述言語であるUnified Modeling Language (Booch et al。, 2005)を用いたクラス図を示している。各クラス表現は，四角で囲まれた表で現されている。 各表において，トップの行はクラス名を含み，真中はフィールドを定義し，最も下の行は，メソッドを定義している。２つのクラスの表を繋ぐ線は，それぞれのクラスのインスタンスが動作するために，他のクラスの情報を必要とすることを示している。

【００１８】
＜起動部＞
図3の起動部は， TrayLauncher クラスをインスタンス化することで実装される。JavaのJPanelクラスをインスタンス化した，アプリケーションの名前をリストアップしたメニューが，applicationMenu フィールドに格納され，このメニューを起動するためのボタンがJava Buttonクラスをインスタンス化してlaunchButtonフィールドに格納される。コラボトレイを１つのアプリケーションを載せて起動するメソッドmakeTray() はイメージ/イベントハンドラの生成のため，TrayImageEventHandler クラスをインスタンス化するよう用意される。メソッド cloneHandler() は，クローンコラボトレイのイメージ/イベントハンドラを新しく生成するために用意される。全ての生成されたイメージ/イベントハンドラは， コラボトレイ同士の接触を監視するために，handlerListのフィールドに格納される。
applicationMenu フィールドに格納されたメニューは，これを載せたコラボトレイを起動することで使用可能になる。メニュー中でアプリケーションを特定することで，これを載せたコラボトレイが新しく起動される。

【００１９】
＜イメージ/イベントハンドラ＞
イメージ/イベントハンドラは，３つのクラスEventTransmitter，TrayImageSender，と TrayImageReceiver のインスタンス化を行いながら，TrayImageEventHandler クラスのインスタンス化を行うことで，実装される。
　イメージ/イベントハンドラは、変換部、短期記憶部、イメージ送信部とイメージ受信部（Image-Receiver）からなる。
変換部は， 回転，移動，拡大の機能のための３つのメソッドrotateAction()，moveAction()，と magnifyAction()，そしてEventTransmitter クラスで，イベントの座標とその種類をアプリケーションハンドラ（Application-Handler）に送るメソッドtransmitEvent() により実現される。イメージ/イベントハンドラが生成されるたび，EventTransmitter クラスのインスタンスは， eventTransmitter フィールドに格納される。
イメージ表示部は，アプリケーションのウィンドウイメージの描画領域への描画とコラボトレイのクローニングをそれぞれ行う２つのメソッドpaintComponent() とshareAction() を用意することで実現される。描画領域は，３つの部分領域からなる：(1) コラボトレイの回転機能をアフォードする円形領域（Java Ellipse2Dクラスをインスタンス化しvewerAreaフィールドに格納される），(2) アプリケーションのウィンドウイメージを描画する長方形領域（JavaのRectangle2Dクラスをインスタンス化し，applicationAreaフィールドに格納される，(3) 回転，移動，拡大，そして共有機能を使うためのハンドルのイメージを描画する円形領域の残りの領域又は長方形領域に描画するアプリケーションのウィンドウイメージを取得するために，TrayApplicationHandler クラスのインスタンスがapplicationHandler フィールドに格納される。
短期記憶部は，短期記憶とシステムクリップボードを使う配信機能のためのメソッドfeedAction() を用意することで実現される。短期記憶は，Java Clipboard クラスをインスタンス化して，localClipboard フィールドに格納される。

イメージ送信部とイメージ受信部（Image-Receiver）は，それぞれ， TrayImageSender クラスとTrayImageReceiver クラスをインスタンス化することで実現される。 前者のインスタンスは，imageSender フィールドに格納され，後者のそれはimageReceiver フィールドに格納される。Java Socket あるいは ServerSocket クラスのインスタンスを用いるメソッドrun() はイメージの送信あるいは受信のために容易される。メソッドsetImage() はクローンコラボトレイ上のウィンドウイメージを更新するために用意される。

【００２０】

＜アプリケーションハンドラ＞

アプリケーションハンドラは，アプリケーション走査部、コンポーネント情報格納部、イメージ格納部、イベント分配部、イベント送信部とイベント受信部からなり、 TrayEventSender とTrayEventReceiver の２つのクラスのインスタンス化とともに，TrayApplicationHandler クラスをインスタンス化することで実装される。

アプリケーションハンドラのアプリケーション走査部は，コンポーネント情報格納部のための情報を取得するメソッドgetAllComponent()とイメージ格納部のための情報を取得する２つのメソッド setImageBuffer() と getImage() を用意することで実現される。メソッドgetAllComponent() は，コンポーネントのイメージのアプリケーションのウィンドウイメージ上での相対座標と従属関係を，Java API メソッド，例えば getComponents()を利用することで集める。これらの座標と関係はcomponentList フィールドに格納される。

コンポーネントのリスナ，すなわち，イベントを扱うルーチン，も同様に，例えばgetMouseListener()のようなJava API メソッドを用いて集められる。集められたリスナは，その扱うイベントの種類に応じたフィールド，例えばmouseListenerList フィールド，に格納される。集められた，座標，関係およびリスナは，コンポーネント情報格納部の情報となる。

イメージ格納部の情報の取得は，コラボトレイをJavaで実装しようとすると少々複雑となる。なぜなら，アプリケーションのウィンドウイメージは，アプリケーションウィンドウ上のイベントとは独立に更新されるコンポーネントのイメージからなるためである。ここで，２つのメソッドsetImageBuffer() と getImage() は，すべての更新されるコンポーネントイのイメージをインターセプトするステートメントをソースコード中に組み込むことで集めるようにする。これは，Javaで書かれたコードが，オペレーティングシステム上で，Java Virtual Machineによって実行されることによる。 図7(a)と(b)に示したJavaコードは，このインターセプトを説明するためのものである。 メソッドsetImageBuffer() は，コンポーネントのイメージ I を描画するための組み込みメソッドdrawImage() をソースコードのメソッドpaint()の中で見つける。さらに，setImageBuffer()は，以下の４つのステートメントを適切な位置に書き加える：(1) toTrayimage フィールドの定義,(2)イメージIのためのJava BufferedImage クラスのインスタンス B （すなわち イメージIと同じ幅，高さ，タイプのイメージを書くためのバッファ）を取得し，それをtoTrayimageに格納する，(3)Bに対するJava Graphics クラスのインスタンス（すなわち，B上にイメージを描画するための情報）を特定して，それをgTrayに格納し，そして(4)イメージI を B に特定した情報をもとに描画する。 メソッドgetImage() は，このようなコンポーネントのイメージを集め，それらをimageBufferList フィールドに格納し，それらを１つのイメージに統合して，そして最後にそれをapplicaitonImageフィールドに格納する。この最後のイメージが，イメージ格納部のイメージとなる。

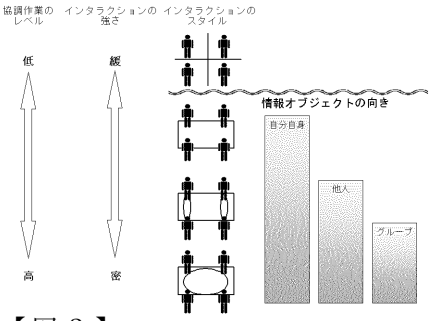
イベント分配部は，TrayImageEventHandlenのeventTransmitter フィールドを参照してイベントを見つけ，componentListフィールドを参照してイベントの対象となるコンポーネントにそれを送るという，イベント分配のためのメソッドdistributeEvent() を用意することで実現される。

イベント送信部とイベント受信部は，それぞれ，TrayEventSender と TrayEventReceiver をインスタンス化することで実現される。前者のインスタンスは，eventSender フィールドに，後者のインスタンスは， eventReceiver フィールドに格納される。イベントの送信と受信は，イメージの送信と受信と同様に実現される。

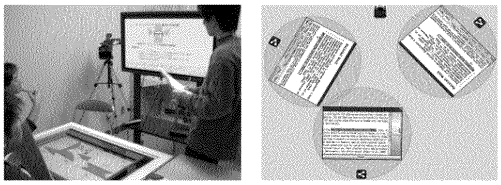
【００２１】
５．コラボトレイの利用例（起動から，移動，回転/拡大，配信（データ転送），共有の簡単な例とその図を紹介する）
図8は，あるユーザが起動ボタンに触れ，メニューを載せたコラボトレイを起動した状況を示している。８つのアプリケーションの名前がリストアップされたメニューが，コラボトレイのイメージ/イベントハンドラの長方形の描画領域に描画されている。移動，回転/拡大，そして共有を行うためのハンドルは，円形の描画領域の残りの部分に描画される。ユーザがテキストエディタをメニューで選んだとき，テキストエディタを載せた新しいコラボトレイが図9の左部分のように起動される。図9の右部分に示すように，このコラボトレイに載せられたテキストエディタは，移動用ハンドルをドラッグすることで，任意の位置に動かすことができ，また，回転／拡大ハンドルを円周方向あるいは半径方向にドラッグすることで，それぞれ回転と拡大ができる。
【００２２】
図10は，テキストエディタを載せた真中のコラボトレイがプレゼンテーションツールを載せた左側のコラボトレイに接触させられている状況を示している。真中のコラボトレイを有するユーザは，カーソルをドラッグすることで強調表示したテキストを，コピーボタンを押して配信しようとしている。このテキストは，左のコラボトレイの短期記憶部に送られる。左のコラボトレイを有するユーザは，プレゼンテーションツールのペーストボタンを押すことで，これらの語を受け取る（受け取ったテキストは四角で囲まれている）。
【００２３】
図10 は，また，左のコラボトレイを有するユーザが，共有ハンドルを押して，新しく生成したクローンコラボトレイを右側のユーザに渡している状況を示している。ユーザがクローンコラボトレイをリモートのコンピュータ上で生成することを望む場合は，オリジナルのコラボトレイを図中でプラグの形をしたアイコンに接触させればよい。
【００２４】
６．評価
協調作業の異なるレベルにおけるコラボトレイの適用可能性を調べるために， 我々は３つの作業領域を用意した。それぞれの場合のインタラクションは漸進的に疎から密へと変化する。また，使い易さを調べるためのユーザスタディも行った。
【産業上の利用可能性】
【００２５】
本発明は、テーブル型ディスプレイを使用して，複数人が協調的に作業を行う際に，普段の作業で使用しているウィンドウアプリケーションを利用できる。
即ち、テーブル型ディスプレイ上でウィンドウアプリケーションをユーザが自由に回転・移動・拡大・縮小できる。従来のToolkit，例えばDiamondSpinを利用して，テーブル型ディスプレイ専用のアプリケーションを開発することでも可能になるが，本発明では一般的に普及しているウィンドウアプリケーションで行うことができる。
同一のウィンドウアプリケーションを共有することができる。VNCなどの共有システムでも可能だが，本発明ではアプリケーション単位で共有が可能で，同一のディスプレイ内でもウィンドウアプリケーションを複数表示して共有できる。
【図面の簡単な説明】
【００２６】
【図１】協調作業のレベルとインタラクションの強さを示す説明図。
【図２】大学の研究室での作業環境の一例を示し
【図３】本発明のコンピュータのコラボトレイの実施例の構成概略図である。
【図４】当該コラボトレイを構成するイメージ/イベントハンドラが，システムのクリップボードからその内容を取り戻す情報スイッチバックのメカニズムを視覚的に示した説明図。
【図５】当該コラボトレイのクローニング説明図。
【図６】ソフトウェアデザインのための記述言語であるUnified Modeling Language (Boo

ch et al。, 2005)を用いた当該コラボトレイのクラス図を示す。
【図７】インターセプトを説明するためのJavaコードである。
【図８】メニューを載せた当該コラボトレイを示す説明図である。
【図９】表示部の回転，移動，および拡大を示す図である。
【図１０】情報オブジェクトの配信とアプリケーションとその内容の共有

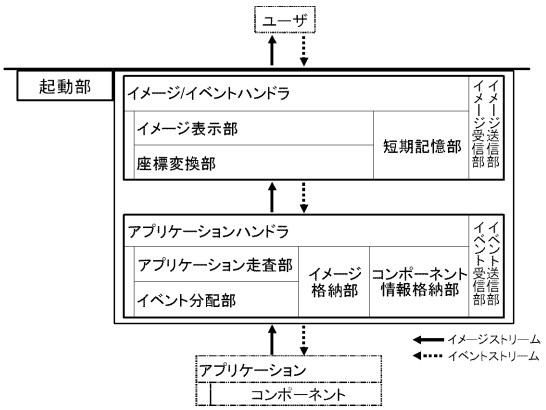
【図１】



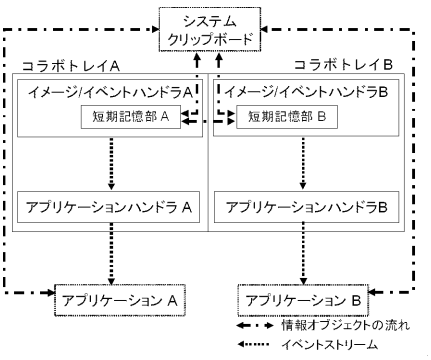
【図２】



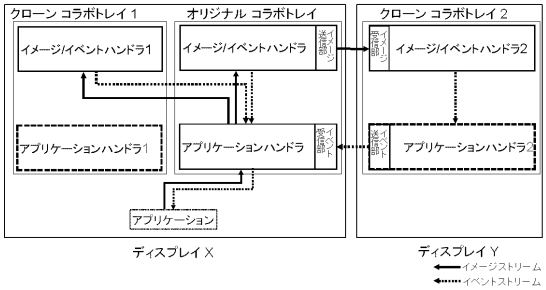
【図３】



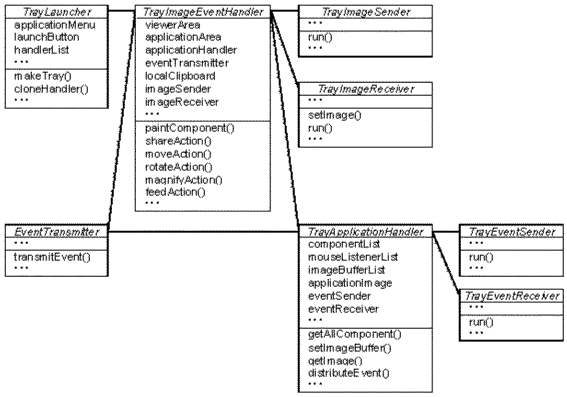
【図４】



【図５】



【図６】



【図７】

```
public void paint(Graphics g) {
   Image img ;
   …
   g.drawImage(img , 0 , 0 , this);
}
```

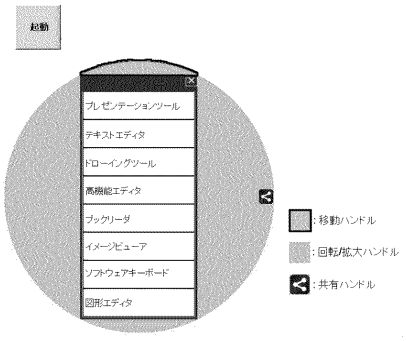
(a)

```
public BufferedImage toTrayImage ;                                    (1)
public void paint(Graphics g) {
 Image img ;
   …
   g.drawImage(img , 0 , 0 , this);
   toTrayImage = new BufferedImage(img.getWidth(),img.getHeight(),img.getType());  (2)
   Graphics gTray = toTrayImage.getGraphics();                        (3)
   gTray.drawImage(img , 0 , 0 , this);                               (4)
}
```

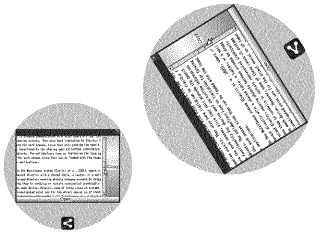
(b)

コンポーネントのイメージを集めるため，ソースコード (a) が (b) のように修正される

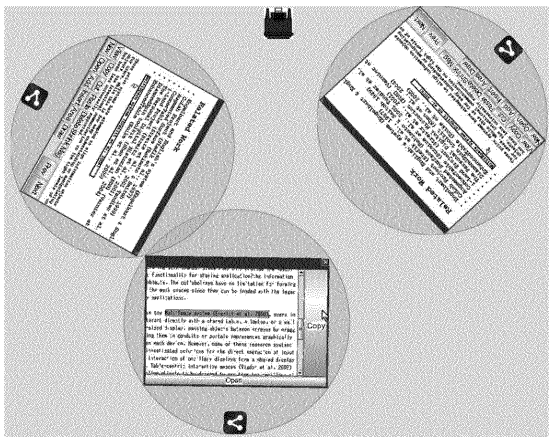
【図８】



【図９】



【図１０】



フロントページの続き

(72)発明者　吉原正樹
大分県大分市大字旦野原７００番地国立大学法人大分大学内
(72)発明者　安部祐樹
大分県大分市大字旦野原７００番地国立大学法人大分大学内
(72)発明者　松迫和樹
大分県大分市大字旦野原７００番地国立大学法人大分大学内

Ｆターム(参考) 5E501 AA01　AC15　BA05　CA01