(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2009-205685
(P2009-205685A)
(43)公開日　平成21年9月10日(2009.9.10)

(51)Int.Cl.	F I	テーマコード（参考）
G06F 3/038 (2006.01)	G06F 3/038 310Z	5B087

審査請求 有　請求項の数 53　ＯＬ　外国語出願　（全 49 頁）

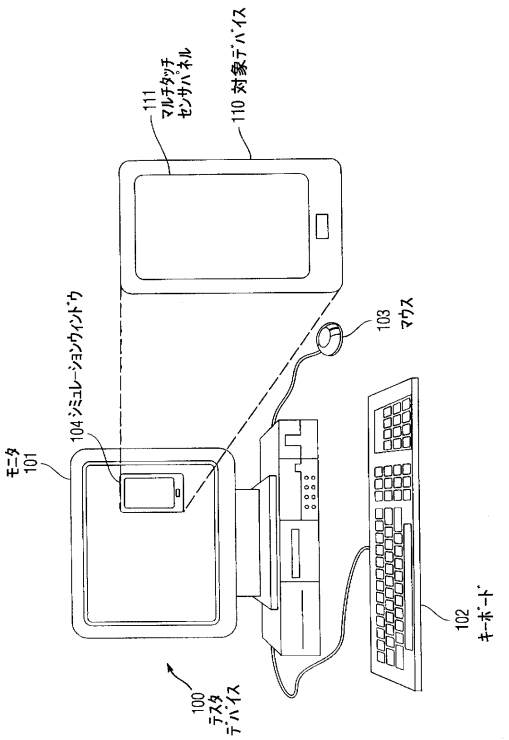
(21)出願番号　特願2009-70904(P2009-70904)
(22)出願日　平成21年2月26日(2009.2.26)
(31)優先権主張番号　12/037,848
(32)優先日　平成20年2月26日(2008.2.26)
(33)優先権主張国　米国(US)

(71)出願人　503260918
アップル　インコーポレイテッド
アメリカ合衆国　９５０１４　カリフォルニア州　クパチーノ　インフィニット　ループ　１
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100067013
弁理士　大塚　文昭
(74)代理人　100086771
弁理士　西島　孝喜
(74)代理人　100109070
弁理士　須田　洋之
(74)代理人　100109335
弁理士　上杉　浩

最終頁に続く

(54)【発明の名称】シングルポインティングデバイスによるマルチポイントジェスチャのシミュレーション

(57)【要約】
【課題】本発明は、シングルポインティングデバイスを使用するコンピュータシステムがマルチポイントジェスチャ入力をシミュレートできるようにする。
【解決手段】シミュレーションソフトウェアが、（例えば、マウスからの入力などの）シングルポインティング入力を受け取り、これを指のピンチ、逆ピンチ、平行移動、回転などのシミュレートしたマルチポイントジェスチャ入力に変換することができる。このシミュレーションソフトウェアにより、ユーザは、マルチポイントジェスチャ入力を作成する際に、キーボードのキーを使用して追加の制御を行えるようにもなる。
【選択図】図１



【特許請求の範囲】
【請求項１】
マルチポイントセンサパネル上のマルチポイントジェスチャをシミュレートするためのシステムであって、
前記マルチポイントセンサパネルの表現を表示するためのディスプレイと、
シングルポインティングユーザ入力デバイスと、
前記シングルポインティングユーザ入力デバイスから入力を受け取り、所定の変換ルールに従って該入力をマルチポイントジェスチャに変換するように構成されたデバイスシミュレータと、
を備えることを特徴とするシステム。
【請求項２】
前記シングルポインティングユーザ入力デバイスはマウスである、
ことを特徴とする請求項１に記載のシステム。
【請求項３】
前記システムは、前記センサパネルを含むマルチポイント対応デバイスにおいて実行するように意図されたソフトウェアを実行するように構成されたプロセッサをさらに含み、前記デバイスシミュレータは、前記変換されたマルチポイントジェスチャ入力を、該マルチポイントジェスチャ入力が前記マルチポイント対応デバイスにおいて実行されたとしたら前記ソフトウェアが受け取ったであろうフォーマットと同一のフォーマットで前記ソフトウェアへ送るようにさらに構成された、
ことを特徴とする請求項１に記載のシステム。
【請求項４】
前記シングルポインティングユーザ入力デバイスからの前記入力は、前記シングルポインティングユーザ入力デバイスが制御するカーソルがたどる経路により定められる、
ことを特徴とする請求項１に記載のシステム。
【請求項５】
前記変換されたマルチポイントジェスチャ入力は、少なくとも２つの別個のポイントジェスチャ入力により定められ、第１のポイントジェスチャ入力は、前記シングルポインティングデバイスが制御する前記カーソルがたどる前記経路により定められ、少なくとも１つの別のポイントジェスチャ入力は、前記シングルポインティングデバイスが制御する前記カーソルがたどる前記経路から導き出された経路により定められる、
ことを特徴とする請求項４に記載のシステム。
【請求項６】
前記少なくとも１つの別のポイントジェスチャ入力は、前記シングルポインティングデバイスが制御するカーソルから所定のベクトルだけ変位したポイントがたどる経路により定められる、
ことを特徴とする請求項５に記載のシステム。
【請求項７】
前記少なくとも１つの別のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルの位置と所定のポイントに関して対称の位置にあるポイントがたどる経路により定められる、
ことを特徴とする請求項５に記載のシステム。
【請求項８】
前記デバイスシミュレータは、前記シングルポインティングデバイスを通じて連続して入力された複数のシングルポイントジェスチャ入力を入力として受け取り、
前記複数の受け取ったシングルポイントジェスチャ入力を前記マルチポイントジェスチャ入力と合成して、前記マルチポイントジェスチャ入力が、前記複数のシングルポイントジェスチャ入力により定められる複数のシングルポイントジェスチャの少なくとも部分的に同時に行われる動作を表すようにすることにより、前記複数のシングルポイントジェスチャ入力を前記マルチポイントジェスチャ入力に変換するように構成された、

ことを特徴とする請求項１に記載のシステム。
【請求項９】
ＣＰＵ及びコンピュータ可読メモリをさらに含み、前記デバイスシミュレータは、前記コンピュータ可読メモリに記憶されるとともに前記ＣＰＵにより実行されるソフトウェアである、
ことを特徴とする請求項１に記載のシステム。
【請求項１０】
前記マルチポイントセンサパネルを含むマルチポイント対応デバイスのためのソフトウェアを開発するように構成されたソフトウェア開発キットをさらに含み、該ソフトウェア開発キットは、前記コンピュータ可読メモリに記憶されるとともに前記ＣＰＵにより実行され、前記デバイスシミュレータは前記ソフトウェア開発キットの一部である、
ことを特徴とする請求項９に記載のシステム。
【請求項１１】
前記マルチポイントジェスチャはマルチタッチジェスチャであり、前記マルチポイントセンサパネルはマルチタッチセンサパネルであり、前記マルチポイントジェスチャ入力はマルチタッチジェスチャ入力である、
ことを特徴とする請求項１に記載のシステム。
【請求項１２】
マルチポイントジェスチャをシミュレートする方法であって、
シングルポインティングデバイスからシングルトラッキング入力を受け取るステップと、
前記受け取ったシングルトラッキング入力に応答して、２又はそれ以上のシミュレートしたタッチポイントを含むとともに前記シングルトラッキング入力に少なくとも一部基づくシミュレートしたマルチポイントジェスチャの視覚的表現を表示するステップと、
を含むことを特徴とする方法。
【請求項１３】
前記シングルポインティングデバイスから初期位置決めコマンドを受け取るステップと、
前記シングルトラッキング入力を受け取る前に２又はそれ以上のシミュレートしたタッチポイントの初期位置を表示するステップと、
をさらに含むことを特徴とする請求項１２に記載の方法。
【請求項１４】
前記シミュレートしたマルチポイントジェスチャは、前記トラッキング入力に所定のルールを適用することにより決定される、
ことを特徴とする請求項１２に記載の方法。
【請求項１５】
前記シングルポインティングデバイスはマウスである、
ことを特徴とする請求項１２に記載の方法。
【請求項１６】
前記シングルポインティングデバイスはシングルタッチトラックパッドである、
ことを特徴とする請求項１２に記載の方法。
【請求項１７】
第１のデバイスにおいて実行するように構成されたソフトウェアを含むコンピュータ可読媒体であって、前記第１のデバイスはシングルポインティングユーザ入力デバイスを含み、前記ソフトウェアは、
前記シングルポインティングユーザ入力デバイスを通じてシングルポインティングジェスチャを受け取るステップと、
前記シングルポインティングジェスチャに基づいて、所定の変換ルールに従ってマルチポイントジェスチャを生成するステップと、
前記マルチポイントジェスチャを表示するステップと、

を行うことにより、マルチポイントセンサパネル上のマルチポイントジェスチャをシミュレートするように構成された、
ことを特徴とするコンピュータ可読媒体。
【請求項１８】
前記マルチポイントジェスチャの生成ステップ及び前記マルチポイントジェスチャの表示ステップは、前記シングルポインティングジェスチャを受け取っている間にリアルタイムで行われる、
ことを特徴とする請求項１７に記載のコンピュータ可読媒体。
【請求項１９】
前記ソフトウェアは、
制御信号を受け取り、
前記制御信号に基づいて、前記マルチポイントジェスチャの生成時に従う変換ルールとして複数の所定の前記変換ルールのうちの１つを選択するようにさらに構成された、
ことを特徴とする請求項１７に記載のコンピュータ可読媒体。
【請求項２０】
前記シングルポインティングユーザ入力デバイスはマウスである、
ことを特徴とする請求項１７に記載のコンピュータ可読媒体。
【請求項２１】
前記第１デバイスにおいて第２のソフトウェアが実行され、該第２のソフトウェアは、前記マルチポイントセンサパネルを含むマルチポイント対応デバイスにおいて実行することを意図されたものであり、前記ソフトウェアは、
前記生成したマルチポイントジェスチャを、該マルチポイントジェスチャが前記マルチポイント対応デバイスにおいて実行されたとしたら前記第２のソフトウェアが受け取ったであろうフォーマットと同一のフォーマットで前記第２のソフトウェアへ送るようにさらに構成された、
ことを特徴とする請求項１７に記載のコンピュータ可読媒体。
【請求項２２】
前記シングルポインティングジェスチャは、前記シングルポインティングデバイスが制御するカーソルがたどる経路により定められる、
ことを特徴とする請求項１７に記載のコンピュータ可読媒体。
【請求項２３】
前記生成したマルチポイントジェスチャは、少なくとも２つの別個のポイントジェスチャにより定められ、前記第１のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルがたどる経路により定められ、少なくとも１つの別のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルがたどる経路から導き出された経路により定められる、
ことを特徴とする請求項２０に記載のコンピュータ可読媒体。
【請求項２４】
前記少なくとも１つの別のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルから所定のベクトルだけ変位した位置がたどる経路により定められる、
ことを特徴とする請求項２３に記載のコンピュータ可読媒体。
【請求項２５】
前記少なくとも１つの別のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルの位置と所定のポイントに関して対称の位置にあるポイントがたどる経路により定められる、
ことを特徴とする請求項２３に記載のコンピュータ可読媒体。
【請求項２６】
前記ソフトウェアはソフトウェア開発キットの一部である、
ことを特徴とする請求項１７に記載のコンピュータ可読媒体。

【請求項２７】
前記マルチポイントジェスチャはマルチタッチジェスチャであり、前記マルチポイントセンサパネルはマルチタッチセンサパネルであり、前記マルチポイントジェスチャ入力はマルチタッチジェスチャ入力である、
ことを特徴とする請求項１７に記載のコンピュータ可読媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般にマルチタッチジェスチャに関し、より具体的には、シングルポインティング入力デバイスを利用してマルチタッチジェスチャをシミュレートすることに関する。
【背景技術】
【０００２】
マルチポイントセンサパネルとは、同時に複数の地点のイベントを感知できるパネルのことである。従って、マルチポイントセンサパネルは、例えば、２つの異なる位置で同時に発生する、パネルに押し付けられた２本の指又はその他のオブジェクトによって引き起こされる２つのタッチイベントを感知することができる。マルチポイントセンサパネルの例については、２００７年１月３日に出願された「近接センサ及びマルチタッチセンサの検出及び復調」という名称の米国特許出願第１１／６４９，９９８号に解説があり、該特許は全体が引用により本明細書に組み入れられる。出願の後半で解説されるように、マルチポイントセンサパネルは、マルチタッチセンサパネルの他に（マルチ近接センサパネルなどの）別のタイプのセンサパネルも含むことができる。マルチポイントセンサパネルを使用して、様々な電子デバイスに向上したユーザインタフェースを提供することができる。
【０００３】
マルチポイントセンサパネルを利用して、向上したユーザ体験を提供する１つの方法として、ユーザがマルチポイントジェスチャを使用してデバイスとコミュニケーションできるようにすることがある。ジェスチャとは、（例えば、通常のマウスクリックの場合のように）単に位置を特定するだけでなく、任意的にある方向及び速度を含む１又は複数のオブジェクトの一定の動きを特定することもできるユーザ入力のことである。例えば、従来のマウスベースのジェスチャでは、通常、ユーザがジェスチャを行うためにマウスボタンを押し、所定の経路に従ってマウスを動かすことが行われる。マルチタッチ機能では、より複雑なジェスチャを使用できるようになる。例えば、ユーザは、パネルの表面上で２又はそれ以上の指を同時に動かすことによりジェスチャを行うことができる。マルチポイントジェスチャ（及びより正確にはマルチタッチジェスチャ）については、２００４年７月３０日に出願された「タッチセンシティブ入力デバイスのためのジェスチャ」という名称の米国特許出願第１０／９０３，９６４号で詳細に解説されており、該特許は全体が引用により本明細書に組み入れられる。
【先行技術文献】
【特許文献】
【０００４】
【特許文献１】米国特許出願第１１／６４９，９９８号
【特許文献２】米国特許出願第１０／９０３，９６４号
【発明の概要】
【発明が解決しようとする課題】
【０００５】
マルチタッチジェスチャの最大の利点を得るために、マルチタッチ対応デバイスで実行されるソフトウェアもまた、マルチタッチ対応であることが必要であると考えられる。しかしながら、このようなソフトウェアの開発は困難な場合がある。普通のパーソナルコンピュータ及び／又はワークステーションコンピュータなどのソフトウェアを開発するため

の既存のコンピュータプラットフォームは、通常マルチタッチ対応ではない。このような対応がなされていなければ、既存のソフトウェア開発コンピュータは、通常、開発中のマルチタッチ対応ソフトウェアをこれらのコンピュータ上でテストすることができない。

【０００６】

開発者は、開発中のソフトウェアをマルチタッチ対応デバイスでロードすることができ、その後このソフトウェアをこのデバイスでテストする。しかしながら、実際には、開発者は様々なバージョンのソフトウェアに対して多くの反復テストを行う必要があり、個々のバージョンのソフトウェアをロードして別個のデバイスでテストする必要があるため、非常に時間がかかるとともに開発プロセスを著しく減速させる可能性がある。

【課題を解決するための手段】

【０００７】

本発明は、シングルポインティングデバイスを使用するコンピュータシステムがマルチポイントジェスチャ入力をシミュレートできるようにすることに関する。シミュレーションソフトウェアが、（例えば、マウスからの入力などの）シングルポインティング入力を受け取り、これを、指のピンチ、逆ピンチ、平行移動、回転などのシミュレートしたマルチポイントジェスチャ入力に変換することができる。このシミュレーションソフトウェアにより、ユーザは、マルチポイントジェスチャ入力を作成する際に、キーボードのキーを使用して追加の制御を行えるようにもなる。

【０００８】

様々な所定の方法により、受け取ったシングルポイントジェスチャ入力をマルチポイントジェスチャ入力に変換することができる。例えば、受け取ったシングルポイントジェスチャ入力を第１のジェスチャ入力として使用し、一方で第１のジェスチャ入力を所定のベクトルだけ変位させることにより第２のジェスチャ入力を作成することができる。これとは別に、或いはこれに加えて、第２のジェスチャ入力を、第１のジェスチャ入力と所定の地点に関して対称なジェスチャとして定めることができる。別の代替例では、複数のシングルポイントジェスチャ入力をシングルポインティングデバイスから連続して受け取り、この連続して受け取った複数のシングルポイント入力の少なくとも部分的に同時に行われた動作を定めるマルチポイントジェスチャ入力に変換することができる。

【図面の簡単な説明】

【０００９】

【図１】マルチタッチジェスチャを特徴とする例示的なデバイスと、本発明の１つの実施形態による、このデバイスのためのソフトウェアの開発に使用する例示的なデバイスとを示す図である。

【図２】本発明の１つの実施形態によるテスタデバイスで実行できる例示的なソフトウェアを示す図である。

【図３Ａ】本発明の１つの実施形態によるタッチの開始位置を定めるための例示的なスキームを示す図である。

【図３Ｂ】本発明の１つの実施形態によるタッチの開始位置を定めるための例示的なスキームを示す図である。

【図４Ａ】本発明の１つの実施形態によるタッチに関するジェスチャの動きを定めるための例示的なスキームを示す図である。

【図４Ｂ】本発明の１つの実施形態によるタッチに関するジェスチャの動きを定めるための例示的なスキームを示す図である。

【図５】本発明の１つの実施形態によるジェスチャを定めるための例示的なスキームを示す図である。

【図６】本発明の１つの実施形態によるジェスチャを定めるための例示的なスキームを示す図である。

【図７】本発明の１つの実施形態によるシングルポインティングデバイスを利用して入力できるいくつかの例示的なシミュレートしたマルチタッチジェスチャを示す図である。

【発明を実施するための形態】

【００１０】
以下の好ましい実施形態の説明では、本明細書の一部を形成する添付図面を参照し、該図面において、本発明を実施できる特定の実施形態を例示目的で示す。本発明の好ましい実施形態の範囲から逸脱することなく、他の実施形態を利用し、構造的な変更を行うことができる。
【００１１】
本発明は、シングルポインティングデバイスを使用するコンピュータシステムがマルチポイントジェスチャ入力をシミュレートできるようにすることに関する。シミュレーションソフトウェアが、（例えば、マウスからの入力などの）シングルポインティング入力を受け取り、これを、指のピンチ、逆ピンチ、平行移動、回転などのシミュレートしたマルチポイントジェスチャ入力に変換することができる。このシミュレーションソフトウェアにより、ユーザは、マルチポイントジェスチャ入力を作成する際に、キーボードのキーを使用して追加の制御を行えるようになる。
【００１２】
ユーザが、シミュレートしたマルチポイントジェスチャ入力を入力した場合、デバイスシミュレータがマーカーを表示し、シミュレートした対象デバイスの画面全体に渡って動かして、マウス及びキーボード（又はその他の入力デバイス）を使用して行われているタッチイベントの種類を示すことができる。これらのマーカーは、例えば、マルチタッチパネル上で、或いはその近くで検出された指先を表す小さな円又はその他の形状であってもよい。マルチポイントソフトウェアをテストする際には、このマーカーを円の重心などの実際の点入力として解釈することができる。
【００１３】
本明細書では、ポータブルデバイス、パーソナルコンピュータ及び／又はワークステーションのマルチポイント機能をシミュレートするという観点で本発明の実施形態について説明するが、本発明の実施形態はこれらのデバイスに制限されるものではなく、一般に、任意のマルチポイント対応デバイスの機能を任意の他のデバイス上でシミュレートすることができる。以下の詳細な説明は、マルチタッチセンサパネルのシミュレーションに焦点を当てたものであるが、その教示をマルチポイントセンサパネルに大まかに適用することができる。
【００１４】
図１は、マルチタッチジェスチャ入力を受け取ることができる例示的なデバイス（１１０）と、本発明の実施形態による、このデバイス用のソフトウェアの開発に使用できるデバイス（１００）とを示す図である。デバイス１１０は、ハンドヘルドデバイス、ノートブックコンピュータなどであってもよい。いくつかの実施形態では、デバイス１１０は、ディスプレイとマルチタッチセンサパネル１１１との組合せを含むことができる。しかしながら、別の実施形態では、デバイス１１０は、トラックパッドなどのディスプレイのないマルチタッチセンサパネルを含むことができる。後半の実施形態のいくつかでは、デバイス１１０は、別個のディスプレイを含むこともできる。例えば、デバイス１１０は、マルチタッチ対応トラックパッドとモニタとを含むノートブックコンピュータであってもよい。
【００１５】
デバイス１００は、ユーザと交信するためのモニタ１０１、キーボード１０２、及びマウス１０３を含むことができる。或いは、デバイスは、ユーザと交信するための他のインタフェースデバイスを含むことができる。本実施例では、デバイス１００は、シングルポインティングデバイス（すなわち、マウス１０３）を含む。マウスは、一度に１つの空間点の選択しか許可しないためシングルポインティングデバイスと考えることができる。これとは逆に、マルチタッチセンサパネルは、（例えば、パネル上又はパネル近くの２又はそれ以上の異なる地点において２又はそれ以上の指を降ろすことにより）一度に複数の空間点の選択を許可するため、マルチポイントティングデバイスと考えることができる。本発明の実施形態は、デバイス１００がシングルポインティングデバイスのみを含むことを求

めるものではなく、マルチポインティングデバイスを含むこともできる。デバイス１００は、ＣＰＵ及び１又はそれ以上のメモリを含むことができる。１又はそれ以上のメモリは命令及びデータを記憶することができ、ＣＰＵはメモリが記憶した命令を実行することができる。従って、デバイス１００は、限定的な意味ではないが、ソフトウェア開発キット（Ｓｏｆｔｗａｒｅ　Ｄｅｖｅｌｏｐｍｅｎｔ　Ｋｉｔ：ＳＤＫ）ソフトウェアを含む様々なソフトウェアを実行することができる。

【００１６】

上述したように、デバイス１００を使用して、デバイス１１０のためのソフトウェアを開発又はテストすることができる。従って、デバイス１００をテスタデバイスと呼ぶことができ、デバイス１１０を対象デバイスと呼ぶことができる。

【００１７】

図２は、本発明の１つの実施形態によるテスタデバイスで実行できる例示的なソフトウェアを示す図である。このソフトウェアは、オペレーティングシステム（ＯＳ２００）を含むことができる。ソフトウェアはまた、ユーザインタフェースアプリケーションプログラミングインタフェース（ＡＰＩ）２０１を含むこともできる。ＡＰＩ２０１は、対象デバイス（すなわち、デバイス１１０）で実行されるプログラムをユーザと交信できるようにするアプリケーションプログラミングインタフェースであってもよい。これらのＡＰＩは、通常、対象デバイス１１０で実行されるが、デバイス１１０のために設計されたソフトウェアをデバイス１００側でテストするためにデバイス１００側で実行することができる。ＡＰＩ２０１は、対象デバイス（１１０）側での実行を意図された対応するＡＰＩと同じものであってもよい。或いは、ＡＰＩ２１０を異なるデバイス（デバイス１００）側で実行できるようにするために、デバイス１１０側で実行されるＡＰＩからＡＰＩ２１０を修正することができる。しかしながら、第２の代替例においても、ＡＰＩ２０１は、同じ又は同様のインタフェースを、これらを使用しているソフトウェア（例えば、本実施例ではソフトウェア２０２）に提供することができる。従って、例えば、ＡＰＩ２０１は、デバイス１１０側で実行される同様のＡＰＩが提供するヘッダと同じヘッダをソフトウェア２０２に提供することができる。

【００１８】

本発明のいくつかの実施形態では、エミュレーションソフトウェア２０５を使用して、ＵＩ　ＡＰＩ２０１をＯＳ２００及びデバイス１００上で実行できるようにすることができる。別の実施形態では、エミュレーションソフトウェアが必要ないように、ＯＳ２００と対象デバイス（１１０）側で実行されるＯＳとは、同一の或いはほぼ同様のものであってもよい。

【００１９】

テスタデバイス１００はまた、テストを受けるソフトウェア２０２を実行することもできる。このソフトウェアは、最終的にはデバイス１１０における実行を目的としたソフトウェアであってもよいが、現在のところ、デバイス１００において開発されテストされている。テストを受けるソフトウェアは、ＵＩ　ＡＰＩ２０１を使用してユーザと交信することができる。ＵＩ　ＡＰＩは、テストを受けるソフトウェアと、それを実行しているデバイスとの間の全ての通信を実現することができる。上述したように、テスタデバイスで実行されるＵＩ　ＡＰＩ２０１は、対象デバイス１１０で実行される同様のＡＰＩと同一のもの又は非常に類似したものであってもよい。従って、ＵＩ　ＡＰＩは、テストを受けるソフトウェアに、実際にデバイス１１０側で実行されているように見せることができる。すなわち換言すれば、ＵＩ　ＡＰＩにより、テストを受けるソフトウェアが対象デバイス１１０側で実行されたとしたら行ったであろう外部と通信するための方法と同じ方法を使用できるようになる。

【００２０】

通常（すなわち、対象デバイス１１０側で実行される場合）、ＵＩ　ＡＰＩ２０１は、デバイス１１０のより低水準のソフトウェア及び／又はハードウェアと通信することができ、これらは様々なユーザインタフェース機能を実行することができる。従って、ＵＩ

ＡＰＩは、情報又はグラフィクスが表示されるようにするために、及び／又はこれらがユーザ入力を示すタッチイベントを受け取るようにするために、デバイス１１０のディスプレイ／マルチタッチパネル１１１（又はディスプレイ／マルチタッチパネルを制御するより低水準のソフトウェア）と通信することができる。しかしながら、ＵＩ　ＡＰＩがデバイス１００側で実行されている場合、デバイス１００はこのような要素を含むことができないため、ＵＩ　ＡＰＩはディスプレイ／マルチタッチパネル１１１と通信することができない。テスタデバイス１００はディスプレイ１０１を含むことができるが、このディスプレイ１０１は、対象デバイス１１０のディスプレイとは異なる種類のものであってもよい。さらに、デバイス１００は、いずれのマルチタッチセンサパネルも含む必要はない。

【００２１】

従って、デバイスシミュレータ２０３を使用して、デバイス１１０のディスプレイ及び／又はマルチタッチセンサパネルをデバイス１００側でシミュレートすることができる。デバイスシミュレータは、ディスプレイ／マルチタッチパネル１１１に接続するために、これらのＵＩ　ＡＰＩが対象デバイス１１０において通信を行う同じ種類の（単複の）インタフェースをＵＩ　ＡＰＩ２０１に提供することができる。デバイスシミュレータ２０３は、デバイス１００のディスプレイ１０１にウィンドウ１０４（図１を参照）が表示されるようにすることができる。デバイスシミュレータは、デバイス１１０が、テストを受けるソフトウェア２０２及びＵＩ　ＡＰＩ２０１を実行したとしたら出力したであろう同じ又は類似のグラフィクスをウィンドウ１０１に出力することができる。従って、ウィンドウ１０４は、デバイス１１０のディスプレイのシミュレーションであると考えることもできる。

【００２２】

同様に、デバイスシミュレータ２０３は、デバイス１００のユーザからのユーザ入力を受け入れ、この入力をデバイス１１０のユーザから受信したであろう種類に変換することができる。従って、デバイスシミュレータは、（キーボード１０２及びマウス１０３などの）デバイス１００のインタフェースデバイスを介して行われた入力を受け入れ、この入力を、マルチタッチセンサパネルが生成したであろう入力に変換することができる。デバイスシミュレータがこの変換をどのように行うかについてのさらなる詳細を以下に示す。

【００２３】

いくつかの実施形態では、デバイスシミュレータは、サウンド、マイク、電源又はその他のボタン、光センサ、加速度センサなどのデバイス１１０の他の入力／出力機能をシミュレートすることもできる。

【００２４】

いくつかの実施形態では、テスタデバイス１００及び対象デバイス１１０は、異なる命令セットを含む異なる種類のプロセッサを使用することができる。このような場合、テストを受けるソフトウェア２０２及びＵＩ　ＡＰＩは、一方がデバイス１００側での実行を意図され、他方がデバイス１１０側での実行を意図された２つの異なるバージョンを各々含むことができる。この２つのバージョンは、同じ又は類似の高水準コードを、デバイス１００及び１１０に関連する２つの異なる命令セットにコンパイルした結果であってもよい（この実施例のために、高水準コードは、アセンブリコード及びマシンコードよりも高水準の任意のコードを含むことができる）。従って、デバイス１００を使用して、テストを受けるソフトウェア２０２の高水準コードをテストすることができる。デバイス１００及び１１０のためのコンパイラがエラー又は不一致を全く持ち込まなければ、これで十分と言える。

【００２５】

ソフトウェア開発キット（ＳＤＫ）２０４をデバイス１００側で実行することもできる。ＳＤＫを使用して、テストを受けるソフトウェア２０２を開発することができる。さらに、ＵＩ　ＡＰＩ（２０１）及びデバイスシミュレータ（２０３）は、ＳＤＫを使用して開発したソフトウェアのテストに使用するＳＤＫの一部であると考えることができる。代替の実施形態では、ＳＤＫをデバイス１００で実行する必要はない。これらの実施形態で

は、デバイス１００をテスト目的で使用することができ、必ずしもソフトウェア開発のために使用する必要はない。

【００２６】

いくつかの実施形態では、デバイス１００をテスト又はソフトウェア開発に使用する必要は全くない。代わりに、デバイス１００を使用して、デバイス１１０向けのソフトウェアを単純に実行し、デバイス１１０のシミュレーションを行うことができる。例えば、本発明の実施形態を使用してマルチタッチ対応デバイスの動作のデモンストレーションを行うことにより。ユーザがそのデバイスを購入するかどうかを決定できるようになる。

【００２７】

前述したように、シミュレーションソフトウェアは、シングルポインティング入力又は（マウスで入力されたジェスチャなどの）ユーザから出されたシングルポインティングジェスチャを受け入れ、これをマルチタッチジェスチャ入力に変換することができる。シミュレーションソフトウェアは、結果として得られたマルチタッチジェスチャ入力に対して、ユーザがキーボードのキーを使用して追加の制御を行えるようにすることもできる。所定のルールに従って、ユーザ入力からマルチタッチジェスチャ入力への変換を行うことができる。

【００２８】

通常、指、手のひら、人体の様々な他の部分、又は（スタイラス又はペンなどの）オブジェクトをマルチタッチセンサパネル上に、或いはその近くに置くことによりマルチタッチジェスチャを行うことができる。本発明のいくつかの実施形態は、ユーザが上記の種類のシミュレートしたジェスチャの全てを入力できるようにすることができる。１つの容易に行われるグループのジェスチャは、タッチセンサパネルの表面上に、或いはその近くに２又はそれ以上の指先を置くこと及び動かすことを伴う。

【００２９】

ユーザがシミュレートしたマルチタッチジェスチャ入力を入力している間、デバイスシミュレータ２０３は、マーカーを表示し、シミュレートした対象デバイスの画面（すなわち、ウィンドウ１０４）全体に渡って動かして、ユーザがマウス及びキーボード（又はデバイス１００の他のインタフェース）を使用して入力しているジェスチャの種類をユーザに示すことができる。これらのマーカーは、例えば、マルチタッチパネルを押す指先を表す小さな円であってもよい。マーカーについて以下にさらに詳細に説明する。

【００３０】

いくつかの実施形態では、ユーザは、開始位置を入力することによりマルチタッチジェスチャシミュレーションを開始することができる。図３Ａ及び図３Ｂは、このような位置を入力する２つの実施例を示す図である。図３Ａ及び図３Ｂは、指先などの２つのタッチポイントを動かすことによって行われるジェスチャに関連するものである。従って、２つの指先の初期位置を定める開始位置の入力を必要とすることができる。

【００３１】

図３Ａ及び図３Ｂは、対象デバイス１１０の画面及び／又はマルチタッチパネルをシミュレートするように意図されたシミュレーションウィンドウ３００及び３０１を示している。いくつかの実施形態では、画面とマルチタッチパネルとを重ね合わせることにより、これらを同じウィンドウ内に表示することができる。従って、ウィンドウ３００及び３０１は、図１のウィンドウ１０４に類似したものであってもよい。

【００３２】

ウィンドウ３００及び３０１は、ジェスチャ入力の初期配置段階を示す。キーボードのキーを押したり、マウスボタン（図示せず）をクリックしたり、或いはシミュレーションウィンドウ（３００又は３０１）上で単純にマウスカーソルを動かしたりなどの様々な方法で初期配置段階を開始することができる。円３０２～３０５はタッチ入力の位置を表す。換言すれば、これらの円は、シミュレートした画面／マルチタッチパネルに触れているバーチャルな指先の位置を表す。

【００３３】

（図３Ａに示す）第１の代替例では、第１のタッチ（３０２）はマウスポインタ（３０８）に従うことができる。第１のタッチから一定の所定の変位した位置に第２のタッチを置くことができる。例えば、第１のタッチ３０２から所定のベクトル３０６だけ離れたところに第２のタッチ３０３を変位させることができる。ベクトル３０６は、例えば、何らかのデフォルト値であってもよいし、或いはユーザが予め定めたものであってもよい。最初に、ユーザは、ウィンドウ３００のあちこちにカーソル３０８を動かすことができ、その後タッチ３０２及び３０３の動きを引き起こすことができる。この結果、ユーザは、これらのタッチのための望ましい位置を探し出し、ユーザの望むタッチの初期位置を示すことができるようになる（これは、例えばマウスボタンをクリックすることにより行われる）。このようにして、ユーザは、（マウスなどの）シングルポインティング入力デバイスのみを使用しながら、２つのタッチを含む所望の開始位置を指定することができる。

【００３４】

第２の代替例では、所定のベクトル３０６の代わりに、所定の中心点３０７を使用することができる。この場合も、ユーザは、マウスポインタ（３０９）を使用して第１のタッチ（３０４）を位置決めすることができる。この代替例では、中心点３０７に関して第１のタッチの位置から鏡の位置、すなわち対称位置に第２のタッチ（３０５）を位置決めすることができる。換言すれば、中心点から第１のタッチまでの変位がベクトル３１０を定める場合、第２のタッチ３０５の位置は、第２のタッチと中心点との間の変位が同じベクトル（３１０）を定めるような位置となる。この場合も、ユーザは、カーソルをあちこちに動かして望ましい位置を決定し、（マウスボタンをクリックすることなどにより）望ましい開始位置を示すことができる。この場合も、ユーザが中心点３０７を入力してもよいし、或いは（ウィンドウの中心などの）デフォルト値を使用してもよい。

【００３５】

様々な実施形態が、開始位置を入力するための上述の代替例のいずれかを利用することができる。いくつかの実施形態は両方の代替例を実施することができ、（ボタンを押したり、或いはクリックしたりすることなどにより）ユーザがこれらの代替例の間で選択を行えるようにすることができる。

【００３６】

いくつかの実施形態では、ユーザは、タッチを操作しながら２つの代替例の間で切り換えを行うことができる。例えば、ユーザは、図３Ａの代替例から開始し、タッチ３０２及び３０３を所望の第１の位置の組に変位させることができる。その後、ユーザは、（キーボードのキーを押すことなどにより）第２の代替例に切り換えることができる。第２の代替例を起動したら、第１の位置の組を使用して中心点を定めることができる。例えば、中心点を、第１の位置の組のタッチ３０２の位置と３０３の位置との間の点として定めることができる。このようにして、ユーザは、所望の中心点を容易に定め、図３Ｂの代替例を使用して開始位置を選択する段階に進むことができる。

【００３７】

また、ユーザは、タッチ３０４及び３０５の第１の位置の組を定めるために図３Ｂの代替例から開始することができる。ユーザは、その後、図３Ａの代替例に切り換えることができる。第１の位置の組を使用して、図３Ａの代替例のベクトル３０６を定めることができる。ユーザは、その後、図３Ａの代替例を使用して実際の初期位置を定めることができる。

【００３８】

両方の代替例では、デバイスシミュレータは、例えばタッチの位置を示す小さな半透明の円を表示することにより、シミュレーションウィンドウ内のタッチ３０２～３０４の位置決めを示すことができる。シミュレーションウィンドウ内に中心点の位置を示すこともできる。図３Ａに示した位置決め方法を並行位置決めと呼び、図３Ｂの方法をミラー位置決めと呼ぶことができる。

【００３９】

当業者であれば、図３Ａ及び図３Ｂに関して上述した教示を、３以上のタッチの位置を

定めるために応用できることを認識するであろう。例えば、異なる所定のベクトルに従って、複数のタッチをタッチ３０２から変位しているものとして定めることができる。これに加えて、或いはこれとは別に、タッチ３０４と中心点（３０７）との間の距離に等しい半径を有する円の周囲に複数のタッチを配置することができる。その後、円を拡大したり、縮小したり、或いは回転させたりすることにより、タッチ３０４を動かすことによってこれらのタッチを動かすことができる。

【００４０】

図３Ａ及び図３Ｂ及び上記の説明は、２又はそれ以上のタッチの初期位置を定める段階について説明したものである。しかしながら、ジェスチャの初期位置によってのみジェスチャを定める必要はない。同様にジェスチャは、初期位置からの何らかの動きを要求することもできる。従って、マルチタッチジェスチャは、タッチの動きを要求することができる。図４Ａ及び図４Ｂは、その初期位置が定められた後のタッチの動きを定めるためのスキームを示す図である。

【００４１】

上述したように、マウスボタンをクリックすることにより、ユーザが所望の初期位置を示すことができる。いくつかの実施形態では、マウスボタンをクリック（又はダウン）したままでマウスを動かすことにより、動きを定めることができる。

【００４２】

初期位置を定める態様と類似した態様で動きを定めることができる。従って図４Ａは、図３Ａに示した初期位置を定めるためのスキームに類似した、動きを定めるためのスキームを示している。このため、図４Ａのスキームを並行移動定義と呼ぶことができる。位置４０２及び４０３は、ユーザが定めた２つのタッチの初期位置を表すことができる。上述したように、図３Ａ及び図３Ｂに関連して上述した方法のいずれか又は両方を使用してこれらの初期位置を入力することができる。或いは、初期位置を入力するための他の方法を使用することもできる。初期位置の設定後、ユーザは、マウスボタンを押したまま経路４１０に沿ってマウスを導くことができる。この結果、デバイスシミュレータは同様に、位置４０２から開始するタッチのグラフィカル表示を経路４１０に沿って位置４０２’に到達するまで導くことができる。デバイスシミュレータはまた、他のタッチ（位置４０３で始まるタッチ）を同様の経路４１１に沿って位置４０３’に到達するまで動かすこともできる。従って、図３Ａの場合と同様に、１つのタッチがマウスカーソルによって動かされている間に、他のタッチがシミュレータによって動かされ、マウスカーソルによって動かされたタッチから所定の距離に留まるようになる。タッチの初期位置決めにより変位ベクトルを定めることができる（すなわち、このベクトルは、位置４０２と４０３との間のベクトルであると考えることができる）。

【００４３】

図３Ａのスキームと図４Ａのスキームとの間の１つの違いは、図４Ａの動きの最中には、デバイスシミュレータが両方のタッチの動きを追跡し、この動きを適切なデータフォーマットに変換し、ジェスチャとしてＵＩ　ＡＰＩ２０１へ送信できるという点である。一方、（マウスボタンが押される前などの）図３Ａの処理中の動きは、初期位置を定めるためにのみ使用することができ、特定の動きの経路を定めるために使用することはできないため、この動きを追跡する必要はない。

【００４４】

図４Ｂは、図３Ｂに示す初期位置を定めるためのスキームに類似した動きを定めるためのスキームを示す図である。換言すれば、図４Ｂはミラー移動の定義を表すことができる。図４Ｂでは、２つのタッチが、位置４０４及び４０５でそれぞれ開始される。カーソル４０９を経路４１４に沿って位置４０４’まで動かすことにより、位置４０４のタッチ（第１のタッチ）を動かすことができる。いくつかの実施形態では、マウスボタンを押しながらカーソルが動かされる。

【００４５】

デバイスシミュレータは、位置４０５から開始するタッチ（第２のタッチ）を、この第

２のタッチの位置が中心点４０７を挟んで第１のタッチの位置から忠実に表現されるような態様で、位置４０５から位置４０５’へ動かすことができる。従って、第２のタッチは経路４１５に沿って移動することができる。２つのタッチの初期位置に基づいて、中心点４０７を定めることができる。従って、この中心点４０７は、（図示のように）初期位置４０４と４０５との間の中心点であると考えることができる。この場合も、デバイスシミュレータは、両方のタッチの動きを追跡し、この動きを適切なデータフォーマットに変換し、ＵＩ　ＡＰＩ２０１へ送信することができる。

【００４６】

いくつかの実施形態は、動きを定めるための図４Ａの方法及び図４Ｂの方法の両方を提供し、ユーザがキーボードのキーを押すことによって切り換えを行えるようにすることができる。いくつかの実施形態では、初期位置をどのようにして定めたかに関わらず、図４Ａ及び図４Ｂの動きの定義スキームを使用することができる。従って、例えば、図３Ａのスキームに従って２つのタッチの初期位置を定めることができる一方で、図４Ｂのスキームに従ってタッチの動きを定めることができる。

【００４７】

いくつかの実施形態では、ユーザは、ジェスチャを定めている途中で図４Ａのスキームと図４Ｂのスキームとの間で切り換えを行うことができる。従って、図４Ａのスキームに従ってジェスチャの一部を定め、図４Ｂのスキームに従ってその他の部分を定めることができる。図４Ａ及び図４Ｂの方法を使用して、図３Ａ及び図３Ｂに関連して上述した態様で、３以上のタッチを含むジェスチャを定めることができる。

【００４８】

上述した方法は、いくつかのマルチタッチ対応デバイスで使用されるいくつかの種類のジェスチャを容易に定めるために有用なものとなり得る。これらのジェスチャは、例えば、２本の指を並行にドラッグしたり、２本の指を狭めたり及び広げたり、（眼に見えないノブを回転させるように）２本の指を回転させたりなどを含むことができる。しかしながら、これらの方法は、２又はそれ以上の指を利用する全ての考えられるジェスチャを定めることはできない。全ての考えられるジェスチャを定めることが必要となる可能性はないため、このことを障害とする必要はない。シミュレートしたデバイス（すなわち、対象デバイス１１０）及び／又はテストを受けるソフトウェアが重要とみなすジェスチャの定義のみをシミュレートする必要がある。

【００４９】

とは言いながらも、図５は、より大きな柔軟性を見込むジェスチャをシミュレートするための別の方法を示す図である。ジェスチャ入力の排他的な方法として、或いは上述した方法の１又はそれ以上の代替例として、様々な実施形態により図５の方法を提供することができる。図５は、マルチタッチジェスチャを定める様々な段階を示すことができる画面５０１、５０２、及び５０３を含む。

【００５０】

図５のスキームによれば、複数のシングルタッチジェスチャの構成要素を別個に定めることにより、マルチタッチジェスチャを定めることができる。最初に、シングルタッチを動かすことにより、第１の構成要素を定めることができる。より具体的には、例えば、５０５の位置にマウスカーソル５０４を置きマウスボタンを押すことにより、シングルタッチの初期位置５０５を選択することができる。次に、マウスボタンを押しながらマウスを動かし、ジェスチャの最後にマウスボタンを放すことによりジェスチャを定めることができる。この結果、ジェスチャが、位置５０５でタッチを開始するステップと、経路５０６に沿ってタッチを動かすステップと、位置５０５’でタッチを終了するステップとを含むことができるようになる。

【００５１】

このようにして、マルチタッチジェスチャの１つの構成要素であるシングルタッチジェスチャを定めることができる。同様の態様で、１又はそれ以上の追加の構成要素を続けて定めることができる。例えば、画面５０２を参照すると、最初に位置５０６においてマウ

スをクリックし、次にこれを経路５０７に沿って位置５０６’まで動かすことにより、第１の構成要素の後に第２のジェスチャの構成要素を定めることができる。いくつかの実施形態では、第２の又はそれに続くジェスチャの構成要素を定めている間に、次の構成要素を定めながら、１又はそれ以上の以前に定めたジェスチャの構成要素を「再生」することができる。定められたジェスチャは、全ての構成要素が少なくとも部分的に同時に実行されると推測するため、上記の「再生」により、関連する構成要素を定める際にユーザを支援することができる。従って、ユーザが、位置５０６から位置５０６’へカーソルを動かすことにより第２の構成要素を定めている間に、デバイスシミュレータにより、位置５０５から位置５０５’へ動かされる別のタッチのアニメーション５０８を同時に表示することができる。

【００５２】

第２のジェスチャの構成要素の入力後、第３のジェスチャの構成要素を入力することができる。第３のジェスチャの構成要素は、位置５０９から経路５１０に沿って位置５０９’へカーソルを動かすステップを含むことができる。同様に、第３のジェスチャの構成要素を入力している間に、２つの以前に入力したジェスチャの構成要素のアニメーション５１１及び５１２を「再生」することができる。

【００５３】

本発明の実施形態は、任意の数のジェスチャの構成要素がこのようにして入力されることを可能にする。いくつかの実施形態では、対象デバイス１１０のユーザがジェスチャの入力に使用すると予想される指の本数に関連して、入力できるジェスチャの構成要素の数が制限される。様々な実施形態により、誤って入力した１又はそれ以上のジェスチャの構成要素を再入力又は削除できるようにもなる。

【００５４】

ユーザが所望の数のジェスチャの構成要素を入力し終えると、ユーザは、（指定されたボタンをクリックすることなどにより）そのように指示を与えることができる。この時点で、デバイスシミュレータは、全てのジェスチャの構成要素を重ね合わせる（すなわち、これらを同時に実行する）ことにより、単一のマルチタッチジェスチャを作成することができる。従って、図５に関連して説明した構成要素に基づいて、デバイスシミュレータは、２本の右の指を下方向にドラッグしながら左端の指を上方向にドラッグするステップを含むマルチタッチジェスチャを作成することができる。

【００５５】

いくつかの実施形態では、デバイスシミュレータは、様々なジェスチャの構成要素を標準化することができる。より詳細には、デバイスシミュレータは、様々な構成要素の速度を調整して、全てのジェスチャの構成要素が同時に開始し、同時に終了できるようにすることができる。代替の実施形態では、いくつかの構成要素が他の構成要素よりも先に終了できるように速度を調整しなくてもよい。さらに別の実施形態では、ユーザは、他のジェスチャの構成要素が開始した後に開始するジェスチャの構成要素を入力できるようになる。

【００５６】

図６は、本発明のいくつかの実施形態によるジェスチャを定めるための別の例示的な方法を示す図である。図５と同様に、要素６０１及び６０２は、ジェスチャを定める際のシミュレーションウィンドウ１０４の様々な段階を示す。最初に、ユーザは、マウスカーソル６０５を位置６０３に置いてボタンをクリックすることにより、静的タッチを定めることができる。ユーザは、例えば、位置６０４においてマウスカーソルをクリックし、マウスカーソルを経路６０６に沿って位置６０４’へ動かすことにより、動くタッチを続けて定めることができる。結果として得られるジェスチャは、位置６０３において１本の指を押したまま、この指を動かさずに位置６０４から経路６０５に沿って位置６０４’へ別の指を動かすステップを表すことができる。或いは、動的タッチ、又は２以上の静的及び／又は動的タッチを定めることができた後に、静的タッチを定めることができる。マルチタッチジェスチャを入力する異なるモードとして図６の方法を提供することができ、それぞ

れの制御キー又はマウスクリック可能なボタンにより、この方法を起動することができる。或いは、図５に関連して上述した方法の特定のケースとして図６の方法を実行することができる。

【００５７】

図７は、本発明のいくつかの実施形態によるシングルポインティングデバイスを使用して入力できるいくつかの例示的なシミュレートしたマルチタッチジェスチャを示す図である。実施例７０１は、ピンチを示す。実施例７０２は、逆ピンチを示す。実施例７０３は、回転を示す。実施例７０４は、シミュレートしたパネルの中心とは異なる位置に回転の中心７０５を選択した場合を示す。当業者であれば、上述した方法を使用して図７の全ての実施例を実施できることを認識するであろう。

【００５８】

当業者であれば、上記に加え、マルチタッチジェスチャを入力するための別の方法を使用できることを認識するであろう。例えば、タッチの輪郭の形状を、例えばマウスでこの輪郭をたどり、或いは所定の選択肢から選択することにより入力することができる。この形状は、単に指先で画面をタッチするよりも複雑なタッチイベントを示すことができる。この形状は、例えば、手のひらで画面をタッチすること、或いは画面上にオブジェクトを置くことを示すことができる。形状を入力し終えると、マルチタッチジェスチャを定めるために、マウスカーソルを動かすことによりこの形状をあちこちに動かすことができる。

【００５９】

上記の説明は、テスタデバイスが（マウスなどの）シングルポインティングデバイスのみを特徴として備える場合に焦点を当てたものであるが、いくつかの実施形態では、テスタデバイスは、同様にマルチタッチパネルを特徴として備えることができる。例えば、テスタデバイスは、マルチタッチ対応トラックパッドを特徴として備えたラップトップであってもよい。対象デバイスは、ディスプレイと一体になったマルチタッチパネルを含むことができる（従って、ユーザは、ディスプレイの表面とやりとりすることにより、マルチタッチ入力を入力することができる）。テスタデバイスは、テスタデバイスのモニタ１０１のシミュレーションウィンドウ１０４内で対象デバイスのディスプレイのシミュレーションを行うことにより対象デバイスをシミュレートすることができる一方で、テスタデバイスのユーザは、テスタデバイスのトラックパッドを使用してマルチタッチ入力を入力できるようになる。テスタデバイスは、ユーザがタッチパッドを通じてタッチを入力している間に、（シミュレーションウィンドウ内に小さな円を表示することなどにより）シミュレーションウィンドウ内にシミュレートしたタッチの位置を示すことができる。

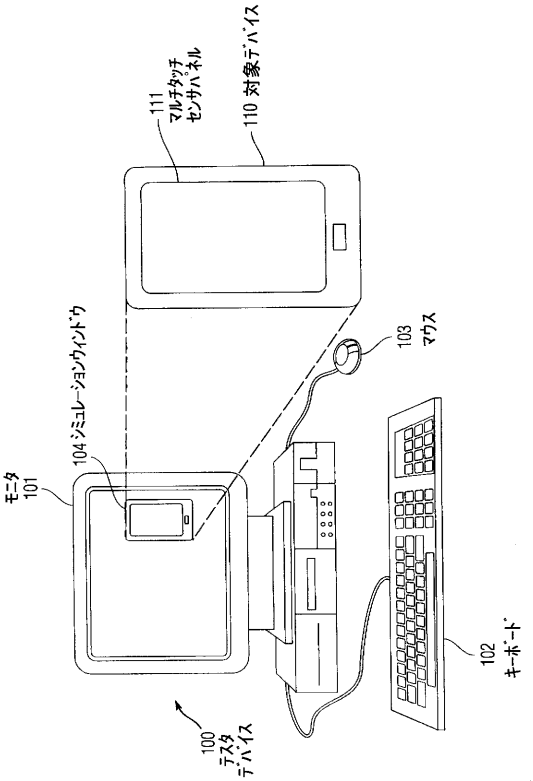
【００６０】

上述した実施形態のいくつかは、シングルポイントジェスチャ入力をマルチタッチジェスチャ入力に変換することに関するものであるが、本発明をこのように限定する必要はない。より一般的には、本発明の実施形態は、シングルポイント入力をマルチポイント入力に変換することに関する。マルチポイント入力は、マルチタッチ入力を含むことができるが、例えば、米国特許出願第１１／６４９，９９８号により解説されるマルチ近接入力などの他の種類の入力を含むこともできる。

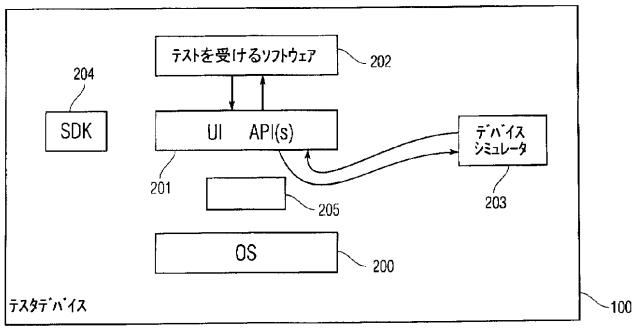
【００６１】

添付の図面を参照しながら本発明の実施形態に関連して本発明について十分に説明してきたが、当業者には様々な変更及び修正が明らかになるであろう。これらの変更及び修正は、添付の特許請求の範囲により定められる本発明の範囲に含まれると理解すべきである。

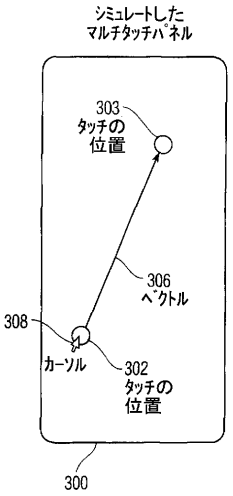
【図１】



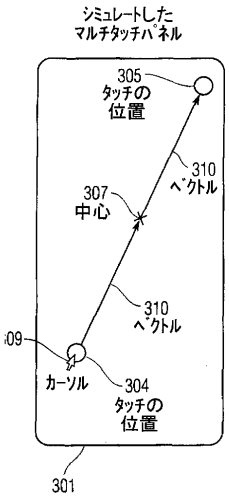
【図２】



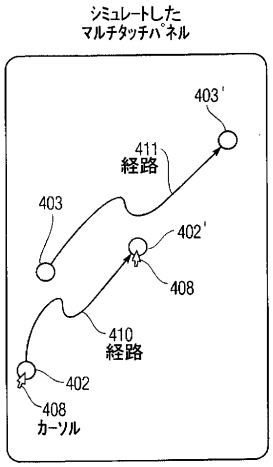
【図３Ａ】



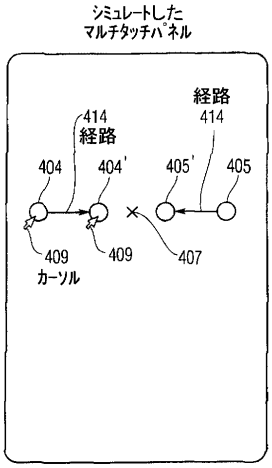
【図３Ｂ】



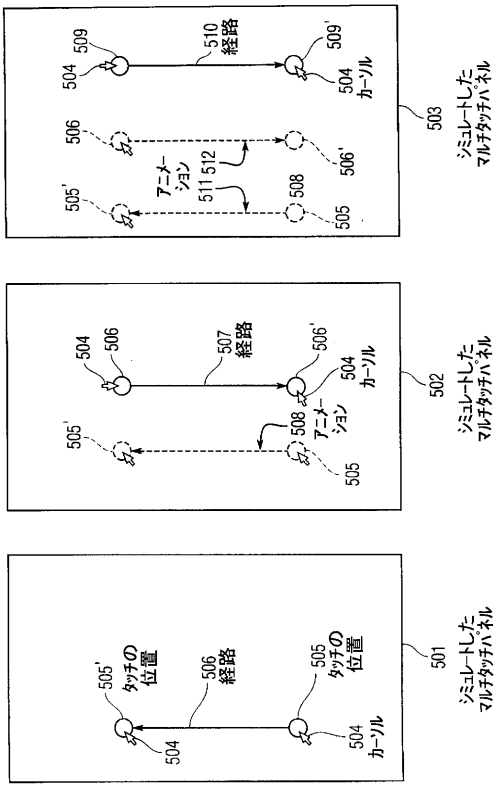
【図４Ａ】



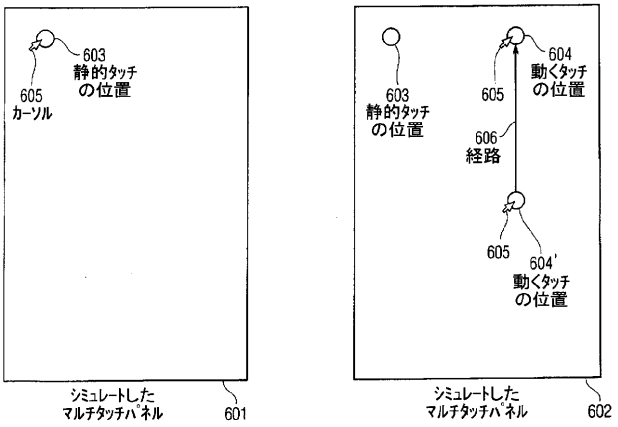
【図４Ｂ】


シミュレートした
マルチタッチパネル
414
経路
404
404'
405'
405
409
カーソル
409
407
経路
414

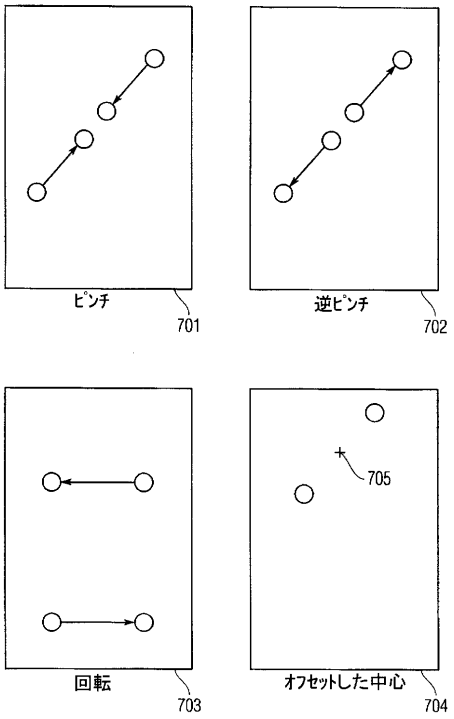
【図５】


509
510
経路
509'
504
カーソル
504
503
シミュレートした
マルチタッチパネル
506
506'
アニメーション
511 512
505'
508
505
504
506
507
経路
506'
504
カーソル
502
シミュレートした
マルチタッチパネル
505'
508
アニメーション
505
505'
タッチの位置
506
経路
505
タッチの位置
501
シミュレートした
マルチタッチパネル
504
504
カーソル

【図６】


603
静的タッチ
の位置
605
カーソル
シミュレートした
マルチタッチパネル
601
603
静的タッチ
の位置
604
動くタッチ
の位置
605
606
経路
605
604'
動くタッチ
の位置
シミュレートした
マルチタッチパネル
602

【図７】


ピンチ
701
逆ピンチ
702
回転
703
705
オフセットした中心
704

【手続補正書】
【提出日】平成21年4月21日(2009.4.21)
【手続補正１】
【補正対象書類名】特許請求の範囲
【補正対象項目名】全文
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】
マルチポイントセンサパネル上のマルチポイント入力をシミュレートするためのシステムであって、
前記マルチポイントセンサパネルの表現を表示するためのディスプレイと、
シングルポインティングユーザ入力デバイスと、
前記シングルポインティングユーザ入力デバイスから入力を受け取り、所定の変換ルールに従って該入力をマルチポイント入力に変換するように構成されたデバイスシミュレータと、
を備えることを特徴とするシステム。
【請求項２】
前記シングルポインティングユーザ入力デバイスはマウスである、
ことを特徴とする請求項１に記載のシステム。
【請求項３】
前記システムは、前記センサパネルを含むマルチポイント対応デバイスにおいて実行するように意図されたソフトウェアを実行するように構成されたプロセッサをさらに含み、前記デバイスシミュレータは、前記変換されたマルチポイント入力を、該マルチポイント入力が前記マルチポイント対応デバイスにおいて実行されたとしたら前記ソフトウェアが前記マルチポイント入力を受け取ったであろう場合のフォーマットと同一のフォーマットで前記ソフトウェアへ送るようにさらに構成された、
ことを特徴とする請求項１に記載のシステム。
【請求項４】
前記シングルポインティングユーザ入力デバイスからの前記入力は、前記シングルポインティングユーザ入力デバイスが制御するカーソルの位置により定められる、
ことを特徴とする請求項１に記載のシステム。
【請求項５】
前記変換されたマルチポイント入力は、少なくとも２つの別個のポイント入力により定められ、第１のポイント入力は、前記シングルポインティングデバイスが制御する前記カーソルの前記位置により定められ、少なくとも１つの別のポイント入力は、前記シングルポインティングデバイスが制御する前記カーソルの前記位置から導き出された位置により定められる、
ことを特徴とする請求項４に記載のシステム。
【請求項６】
前記少なくとも１つの別のポイント入力は、前記シングルポインティングデバイスが制御するカーソルから所定のベクトルだけ変位したポイントがたどる経路により定められる、
ことを特徴とする請求項５に記載のシステム。
【請求項７】
前記少なくとも１つの別のポイント入力は、前記シングルポインティングデバイスが制御するカーソルの位置と所定のポイントに関して対称となる位置により定められる、
ことを特徴とする請求項５に記載のシステム。
【請求項８】
前記デバイスシミュレータは、前記シングルポインティングデバイスを通じて連続して

入力された複数のシングルポイント入力を入力として受け取り、
前記複数の受け取ったシングルポイント入力を前記マルチポイント入力と合成して、前記マルチポイント入力が、前記複数のシングルポイントジェスチャ入力の少なくとも部分的に同時に行われる動作を表すようにすることにより、前記複数のシングルポイント入力を前記マルチポイント入力に変換するように構成された、
ことを特徴とする請求項１に記載のシステム。
【請求項９】
ＣＰＵ及びコンピュータ可読メモリをさらに含み、前記デバイスシミュレータは、前記コンピュータ可読メモリに記憶されるとともに前記ＣＰＵにより実行されるソフトウェアである、
ことを特徴とする請求項１に記載のシステム。
【請求項１０】
前記マルチポイントセンサパネルを含むマルチポイント対応デバイスのためのソフトウェアを開発するように構成されたソフトウェア開発キットをさらに含み、該ソフトウェア開発キットは、前記コンピュータ可読メモリに記憶されるとともに前記ＣＰＵにより実行され、前記デバイスシミュレータは前記ソフトウェア開発キットの一部である、
ことを特徴とする請求項９に記載のシステム。
【請求項１１】
前記マルチポイント入力はマルチタッチ入力であり、前記マルチポイントセンサパネルはマルチタッチセンサパネルである、
ことを特徴とする請求項１に記載のシステム。
【請求項１２】
マルチポイント入力をシミュレートする方法であって、
シングルポインティングデバイスからシングルトラッキング入力を受け取るステップと、
前記受け取ったシングルトラッキング入力に応答して、２又はそれ以上のシミュレートしたタッチポイントを含むとともに前記シングルトラッキング入力に少なくとも一部基づくシミュレートしたマルチポイント入力の視覚的表現を表示するステップと、
を含むことを特徴とする方法。
【請求項１３】
前記シミュレートしたマルチポイント入力は、前記トラッキング入力に所定のルールを適用することにより決定される、
ことを特徴とする請求項１２に記載の方法。
【請求項１４】
前記シングルポインティングデバイスはマウスである、
ことを特徴とする請求項１２に記載の方法。
【請求項１５】
前記シングルポインティングデバイスはシングルタッチトラックパッドである、
ことを特徴とする請求項１２に記載の方法。
【請求項１６】
第１のデバイスにおいて実行するように構成されたソフトウェアを含むコンピュータ可読媒体であって、前記第１のデバイスはシングルポインティングユーザ入力デバイスを含み、前記ソフトウェアは、
前記シングルポインティングユーザ入力デバイスを通じてシングルポインティング入力を受け取るステップと、
前記シングルポインティング入力に基づいて、所定の変換ルールに従ってマルチポイント入力を生成するステップと、
前記マルチポイント入力を表示するステップと、
を行うことにより、マルチポイントセンサパネル上のマルチポイント入力をシミュレートするように構成された、

ことを特徴とするコンピュータ可読媒体。
【請求項１７】
前記マルチポイント入力の生成ステップ及び前記マルチポイント入力の表示ステップは、前記シングルポインティング入力を受け取っている間にリアルタイムで行われる、
ことを特徴とする請求項１６に記載のコンピュータ可読媒体。
【請求項１８】
前記ソフトウェアは、
制御信号を受け取り、
前記制御信号に基づいて、前記マルチポイント入力の生成時に従う変換ルールとして複数の所定の前記変換ルールのうちの１つを選択するようにさらに構成された、
ことを特徴とする請求項１６に記載のコンピュータ可読媒体。
【請求項１９】
前記シングルポインティングユーザ入力デバイスはマウスである、
ことを特徴とする請求項１６に記載のコンピュータ可読媒体。
【請求項２０】
前記第１デバイスにおいて第２のソフトウェアが実行され、該第２のソフトウェアは、前記マルチポイントセンサパネルを含むマルチポイント対応デバイスにおいて実行することを意図されたものであり、前記ソフトウェアは、
前記生成したマルチポイント入力を、該マルチポイント入力が前記マルチポイント対応デバイスにおいて実行されたとしたら前記第２のソフトウェアが前記マルチポイント入力を受け取ったであろう場合のフォーマットと同一のフォーマットで前記第２のソフトウェアへ送るようにさらに構成された、
ことを特徴とする請求項１６に記載のコンピュータ可読媒体。
【請求項２１】
前記シングルポインティング入力は、前記シングルポインティングデバイスが制御するカーソルの位置により定められる、
ことを特徴とする請求項１６に記載のコンピュータ可読媒体。
【請求項２２】
前記生成したマルチポイント入力は、少なくとも２つの別個のポイント入力により定められ、前記第１のポイント入力は、前記シングルポインティングデバイスが制御するカーソルの前記位置により定められ、少なくとも１つの別のポイント入力は、前記シングルポインティングデバイスが制御するカーソルがたどる位置から導き出された位置により定められる、
ことを特徴とする請求項１９に記載のコンピュータ可読媒体。
【請求項２３】
前記少なくとも１つの別のポイント入力は、前記シングルポインティングデバイスが制御するカーソルから所定のベクトルだけ変位した位置により定められる、
ことを特徴とする請求項２２に記載のコンピュータ可読媒体。
【請求項２４】
前記少なくとも１つの別のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルの位置と所定のポイントに関して対称となる位置により定められる、
ことを特徴とする請求項２２に記載のコンピュータ可読媒体。
【請求項２５】
前記ソフトウェアはソフトウェア開発キットの一部である、
ことを特徴とする請求項１６に記載のコンピュータ可読媒体。
【請求項２６】
前記マルチポイント入力はマルチタッチ入力であり、前記マルチポイントセンサパネルはマルチタッチセンサパネルである、
ことを特徴とする請求項１６に記載のコンピュータ可読媒体。

【請求項２７】
マルチポイントセンサパネル上のマルチポイントジェスチャをシミュレートするためのシステムであって、
前記マルチポイントセンサパネルの表現を表示するためのディスプレイと、
シングルポインティングユーザ入力デバイスと、
前記シングルポインティングユーザ入力デバイスから入力を受け取り、所定の変換ルールに従って該入力をマルチポイントジェスチャ入力に変換するように構成されたデバイスシミュレータと、
を備えることを特徴とするシステム。
【請求項２８】
前記シングルポインティングユーザ入力デバイスはマウスである、
ことを特徴とする請求項２７に記載のシステム。
【請求項２９】
前記システムは、前記センサパネルを含むマルチポイント対応デバイスにおいて実行するように意図されたソフトウェアを実行するように構成されたプロセッサをさらに含み、
前記デバイスシミュレータは、前記変換されたマルチポイントジェスチャ入力を、該マルチポイントジェスチャ入力が前記マルチポイント対応デバイスにおいて実行されたとしたら前記ソフトウェアが前記マルチポイントジェスチャ入力を受け取ったであろう場合のフォーマットと同一のフォーマットで前記ソフトウェアへ送るようにさらに構成された、
ことを特徴とする請求項２７に記載のシステム。
【請求項３０】
前記シングルポインティングユーザ入力デバイスからの前記入力は、前記シングルポインティングユーザ入力デバイスが制御するカーソルがたどる経路により定められる、
ことを特徴とする請求項２７に記載のシステム。
【請求項３１】
前記変換されたマルチポイントジェスチャ入力は、少なくとも２つの別個のポイントジェスチャ入力により定められ、第１のポイントジェスチャ入力は、前記シングルポインティングデバイスが制御する前記カーソルがたどる前記経路により定められ、少なくとも１つの別のポイントジェスチャ入力は、前記シングルポインティングデバイスが制御する前記カーソルがたどる前記経路から導き出された経路により定められる、
ことを特徴とする請求項３０に記載のシステム。
【請求項３２】
前記少なくとも１つの別のポイントジェスチャ入力は、前記シングルポインティングデバイスが制御するカーソルから所定のベクトルだけ変位したポイントがたどる経路により定められる、
ことを特徴とする請求項３１に記載のシステム。
【請求項３３】
前記少なくとも１つの別のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルの位置と所定のポイントに関して対称の位置にあるポイントがたどる経路により定められる、
ことを特徴とする請求項３１に記載のシステム。
【請求項３４】
前記デバイスシミュレータは、前記シングルポインティングデバイスを通じて連続して入力された複数のシングルポイントジェスチャ入力を入力として受け取り、
前記複数の受け取ったシングルポイントジェスチャ入力を前記マルチポイントジェスチャ入力と合成して、前記マルチポイントジェスチャ入力が、前記複数のシングルポイントジェスチャ入力により定められる複数のシングルポイントジェスチャの少なくとも部分的に同時に行われる動作を表すようにすることにより、前記複数のシングルポイントジェスチャ入力を前記マルチポイントジェスチャ入力に変換するように構成された、
ことを特徴とする請求項２７に記載のシステム。

【請求項３５】
ＣＰＵ及びコンピュータ可読メモリをさらに含み、前記デバイスシミュレータは、前記コンピュータ可読メモリに記憶されるとともに前記ＣＰＵにより実行されるソフトウェアである、
ことを特徴とする請求項２７に記載のシステム。
【請求項３６】
前記マルチポイントセンサパネルを含むマルチポイント対応デバイスのためのソフトウェアを開発するように構成されたソフトウェア開発キットをさらに含み、該ソフトウェア開発キットは、前記コンピュータ可読メモリに記憶されるとともに前記ＣＰＵにより実行され、前記デバイスシミュレータは前記ソフトウェア開発キットの一部である、
ことを特徴とする請求項３５に記載のシステム。
【請求項３７】
前記マルチポイントジェスチャはマルチタッチジェスチャであり、前記マルチポイントセンサパネルはマルチタッチセンサパネルであり、前記マルチポイントジェスチャ入力はマルチタッチジェスチャ入力である、
ことを特徴とする請求項２７に記載のシステム。
【請求項３８】
マルチポイントジェスチャをシミュレートする方法であって、
シングルポインティングデバイスからシングルトラッキング入力を受け取るステップと、
前記受け取ったシングルトラッキング入力に応答して、２又はそれ以上のシミュレートしたタッチポイントを含むとともに前記シングルトラッキング入力に少なくとも一部基づくシミュレートしたマルチポイントジェスチャの視覚的表現を表示するステップと、
を含むことを特徴とする方法。
【請求項３９】
前記シングルポインティングデバイスから初期位置決めコマンドを受け取るステップと、
前記シングルトラッキング入力を受け取る前に２又はそれ以上のシミュレートしたタッチポイントの初期位置を表示するステップと、
をさらに含むことを特徴とする請求項３８に記載の方法。
【請求項４０】
前記シミュレートしたマルチポイントジェスチャは、前記トラッキング入力に所定のルールを適用することにより決定される、
ことを特徴とする請求項３８に記載の方法。
【請求項４１】
前記シングルポインティングデバイスはマウスである、
ことを特徴とする請求項３８に記載の方法。
【請求項４２】
前記シングルポインティングデバイスはシングルタッチトラックパッドである、
ことを特徴とする請求項３８に記載の方法。
【請求項４３】
第１のデバイスにおいて実行するように構成されたソフトウェアを含むコンピュータ可読媒体であって、前記第１のデバイスはシングルポインティングユーザ入力デバイスを含み、前記ソフトウェアは、
前記シングルポインティングユーザ入力デバイスを通じてシングルポインティングジェスチャを受け取るステップと、
前記シングルポインティングジェスチャに基づいて、所定の変換ルールに従ってマルチポイントジェスチャを生成するステップと、
前記マルチポイントジェスチャを表示するステップと、
を行うことにより、マルチポイントセンサパネル上のマルチポイントジェスチャをシミュ

レートするように構成された、
ことを特徴とするコンピュータ可読媒体。
【請求項４４】
前記マルチポイントジェスチャの生成ステップ及び前記マルチポイントジェスチャの表示ステップは、前記シングルポインティングジェスチャを受け取っている間にリアルタイムで行われる、
ことを特徴とする請求項４３に記載のコンピュータ可読媒体。
【請求項４５】
前記ソフトウェアは、
制御信号を受け取り、
前記制御信号に基づいて、前記マルチポイントジェスチャの生成時に従う変換ルールとして複数の所定の前記変換ルールのうちの１つを選択するようにさらに構成された、
ことを特徴とする請求項４３に記載のコンピュータ可読媒体。
【請求項４６】
前記シングルポインティングユーザ入力デバイスはマウスである、
ことを特徴とする請求項４３に記載のコンピュータ可読媒体。
【請求項４７】
前記第１デバイスにおいて第２のソフトウェアが実行され、該第２のソフトウェアは、前記マルチポイントセンサパネルを含むマルチポイント対応デバイスにおいて実行することを意図されたものであり、前記ソフトウェアは、
前記生成したマルチポイントジェスチャを、該マルチポイントジェスチャが前記マルチポイント対応デバイスにおいて実行されたとしたら前記第２のソフトウェアが前記マルチポイントジェスチャを受け取ったであろう場合のフォーマットと同一のフォーマットで前記第２のソフトウェアへ送るようにさらに構成された、
ことを特徴とする請求項４３に記載のコンピュータ可読媒体。
【請求項４８】
前記シングルポインティングジェスチャは、前記シングルポインティングデバイスが制御するカーソルがたどる経路により定められる、
ことを特徴とする請求項４３に記載のコンピュータ可読媒体。
【請求項４９】
前記生成したマルチポイントジェスチャは、少なくとも２つの別個のポイントジェスチャにより定められ、前記第１のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルがたどる経路により定められ、少なくとも１つの別のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルがたどる経路から導き出された経路により定められる、
ことを特徴とする請求項４４に記載のコンピュータ可読媒体。
【請求項５０】
前記少なくとも１つの別のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルから所定のベクトルだけ変位した位置がたどる経路により定められる、
ことを特徴とする請求項４９に記載のコンピュータ可読媒体。
【請求項５１】
前記少なくとも１つの別のポイントジェスチャは、前記シングルポインティングデバイスが制御するカーソルの位置と所定のポイントに関して対称の位置にあるポイントがたどる経路により定められる、
ことを特徴とする請求項４９に記載のコンピュータ可読媒体。
【請求項５２】
前記ソフトウェアはソフトウェア開発キットの一部である、
ことを特徴とする請求項４３に記載のコンピュータ可読媒体。
【請求項５３】

前記マルチポイントジェスチャはマルチタッチジェスチャであり、前記マルチポイントセンサパネルはマルチタッチセンサパネルであり、前記マルチポイントジェスチャ入力はマルチタッチジェスチャ入力である、
ことを特徴とする請求項４３に記載のコンピュータ可読媒体。

フロントページの続き

(72)発明者 ジョージ アール ディッカー
アメリカ合衆国 カリフォルニア州 ９５０７０ サラトガパセオ ラドー １８５６４
(72)発明者 マルセル ファン オス
アメリカ合衆国 カリフォルニア州 ９４１１０ サンフランシスコ ランディーズ レーン １１６ ユニット エイ
(72)発明者 リチャード ウィリアムソン
アメリカ合衆国 カリフォルニア州 ９５０３３ ロス ガトス サミット ロード ２３５８３
(72)発明者 クリス ブルーメンバーグ
アメリカ合衆国 カリフォルニア州 ９４１１４ サンフランシスコ トゥウェンティファースト ストリート ３６００ ＃３０５

Ｆターム(参考) 5B087 AA09 BB00 DD03 DE05

【外国語明細書】

SIMULATION OF MULTI-POINT GESTURES WITH A SINGLE POINTING DEVICE

Field of the Invention

This relates to multi-touch gestures in general, and more specifically to simulating multi-touch gestures utilizing a single pointing input device.

Background of the Invention

A multi-point sensor panel is a panel that can sense multiple point events at the same time. Thus, a multi-point sensor panel can, for example, sense two touch events that take place simultaneously at two different positions and caused by two fingers or other objects being pressed to the panel. Examples of multi-point sensor panels are discussed in U.S. Pat. Application No. 11/649,998, entitled "PROXIMITY AND MULTI-TOUCH SENSOR DETECTION AND DEMODULATION," filed on January 3, 2007 and hereby incorporated by reference in its entirety. As discussed in the latter application, multi-point sensor panels can include multi-touch sensor panels as well as other types of sensor panels (such as multi-proximity sensor panels). Multi-point sensor panels can be used to provide an improved user interface for various electronic devices.

One way to leverage multi-point sensor panels to provide an improved user experience is to allow users to communicate with the device using multi- point gestures. A gesture is a user input that does not merely specify a location (as is the case with an ordinary mouse click, for example), but can also specify a certain movement of an object or objects, optionally with a certain direction and velocity. For example, traditional mouse based gestures usually provide that a user press a mouse button and move the mouse according to a predefined path in order to perform a gesture. Multi-touch functionality can allow for more complex gestures to be used. For example, a user can perform a gesture by moving two or more fingers on the surface of the panel simultaneously. Multi- point gestures (and more specifically multi-touch gestures) are discussed in more detail in U.S. Pat. Application No. 10/903,964, entitled "GESTURES

FOR TOUCH SENSITIVE INPUT DEVICES," filed on July 30, 2004 and hereby incorporated by reference in its entirety.

In order to obtain the full benefit of multi-touch gestures, software that runs on a multi-touch capable device may also need to be multi-touch capable. However, developing such software can be difficult. Existing computing platforms for developing software, such as ordinary personal computers and/or workstation computers, are usually not multi-touch capable. Without such capabilities, existing software development computers are usually unable to test the multi-touch capable software being developed on them.

A developer can load the software being developed on a multi-touch capable device and then test it there. However, in practice a developer may need to perform many repeated tests on different versions of the software, and having to load each version of the software to be tested on a separate device can prove to be very time consuming and can significantly slow down the development process.

Summary of the Invention

This relates to allowing a computer system using a single pointing device to simulate multi-point gesture inputs. Simulating software can receive single pointing inputs (such as, for example, input from a mouse) and convert them to simulated multi-point gesture inputs such as finger pinches, reverse pinches, translations, rotation, and the like. The simulating software can also allow the user to use keyboard keys to give the user additional control when generating the multi- point gesture inputs.

A received single-point gesture input can be converted to a multi-point gesture input by various predefined methods. For example, a received single point gesture input can be used as a first gesture input while a second gesture input can be generated by displacing the first gesture input by a predefined vector. Alternatively, or in addition, the second gesture input can be defined as a being a gesture symmetrical to the first gesture input with respect to a predefined point. In another alternative, multiple single point gesture inputs can be consecutively received from the single pointing device and

converted into a multi-point gesture input that defines an at least partially simultaneous performance of the consecutively received multiple single point inputs.

Brief Description of the Drawings

Fig. 1 is a diagram of an exemplary device that features multi-touch gestures and an exemplary device used for developing software for that device according to one embodiment of this invention.

Fig. 2 is a diagram showing exemplary software that may run on a tester device according to one embodiment of this invention.

Figs 3A and 3B are diagrams showing exemplary schemes for defining starting locations of touches according to one embodiment of this invention.

Figs 4A and 4B are diagrams showing exemplary schemes for defining gesture movement for touches according to one embodiment of this invention.

Fig 5 is a diagram showing an exemplary scheme for defining gestures according to one embodiment of this invention.

Fig 6 is a diagram showing an exemplary scheme for defining gestures according to one embodiment of this invention.

Fig. 7 is a diagram showing several exemplary simulated multi-touch gestures that may be entered utilizing according to one embodiment of this invention.

Detailed Description of the Preferred Embodiment

In the following description of preferred embodiments, reference is made to the accompanying drawings which form a part hereof, and in which it is shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the preferred embodiments of the present invention.

This relates to allowing a computer system using a single pointing device to simulate multi-point gesture inputs. Simulating software can receive single pointing inputs (such as, for example, input from a mouse) and convert them to simulated multi-point gesture inputs such as finger pinches, reverse pinches, translations, rotation, and the like. The simulating software can also allow the user to use keyboard keys to give the user additional control when generating the multi- point gesture inputs.

When a user enters simulated multi- point gesture inputs, the device simulator can cause markers to appear and move across the simulated subject device screen to indicate the type of touch event being performed using the mouse and keyboard (or other input devices). These markers can be, for example, small circles or other shapes representing fingertips detected on or in proximity to a multi-touch panel. The markers can then be interpreted as actual point inputs, such as the centroid of the circle, when testing multi- point software.

Although embodiments of the present invention may be described herein in terms of simulating the multi- point capabilities of portable devices, personal computers and/or workstations, it should be understood that embodiments of the invention are not limited to such devices, but are generally applicable to simulating the capabilities of any multi- point capable device on any other device. While the detailed description below centers on simulating multi-touch sensor panels, its teachings can apply to multi-point sensor panels in general.

FIG. 1 is a diagram of an exemplary device (110) that may receive multi-touch gesture inputs and a device (100) that can be used for developing software for the device according to embodiments of the invention. Device 110 can be a handheld device, a notebook computer or the like. In some embodiments, device 110 can include a combination of a display and a multi touch sensor panel 111. However, in other embodiments, device 110 can include a multi-touch sensor panel without a display, such as a trackpad. In some of the latter embodiments, device 110 can also include a separate display. For example, device 110 can be a notebook computer which includes a multi-touch capable trackpad and a monitor.

Device 100 can include a monitor 101, a keyboard 102 and a mouse 103 for communicating with a user. Alternatively, the device can include other interface devices for communicating with the user. It should be noted that in the present example, device 100 includes a single pointing device (i.e., mouse 103). The mouse can be considered a single pointing device because it only allows the selection of one spatial point at a time. In contrast, a multi-touch sensor panel can be considered a multi-pointing device because it allows for multiple spatial points to be selected at a single time (e.g., by placement of two or more fingers down at two or more different points on or near the panel). Embodiments of the invention do not require that device 100 include only a single pointing device and can include multi-pointing devices. Device 100 can include a CPU and one or more memories. The one or more memories can store instructions and data, and the CPU can execute instructions stored by the memory. Thus, device 100 may execute various software, including but not limited to Software Development Kit (SDK) software.

As noted above, device 100 can be used for developing or testing software for device 110. Thus, device 100 can be referred to as a tester device and device 110 as a subject device.

FIG. 2 is a diagram showing exemplary software that can run on a tester device according to one embodiment of the invention. The software can include an Operating System (OS 200). The software can also include User Interface Application Programming Interfaces (APIs) 201. APIs 201 can be application programming interfaces that allow programs running on the subject device (i.e., device 110) to communicate with a user. These APIs ordinarily run on subject device 110, but can be executed at device 100 for the purposes of testing software designed for device 110 at device 100. APIs 201 can be the same as corresponding APIs intended to be executed at the subject device (110). Alternatively, APIs 210 can be modified from those that execute at device 110 in order to allow for execution at a different device (device 100). However, even in the second alternative, APIs 201 can provide the same or similar interfaces to software that is using them (e.g., software 202, in the present example). Thus, for example, APIs 201 can provide the same headers to software 202 as would be provided by similar APIs running at device 110.

In some embodiments of the invention, emulation software 205 can be used to allow UI APIs 201 to run on OS 200 and device 100. In other embodiments, OS 200 and the OS running at subject device (110) may be identical or substantially similar, so that no emulation software is necessary.

Tester device 100 can also run software to be tested 202. This software can be software that is eventually intended to be run on device 110, but is presently being developed and tested on device 100. Software to be tested can use UI APIs 201 to communicate with the user. UI APIs, can provide all communications between the software to be tested and the device it is running on. As noted above, the UI APIs 201 running on the tester device can be identical or very similar to similar APIs that run on the subject device 110. Thus, UI APIs can make it appear to the software to be tested that it is actually executing at device 110. Or, in other words, the UI APIs can allow the software to be tested to use the same methods for communicating with the outside world as it would have done if it had been running at the subject device 110.

Ordinarily (i.e., when being executed at subject device 110), UI APIs 201 can communicate with lower level software and/or hardware of device 110, that may perform various user interface functions. Thus, the UI APIs can communicate with display/multi touch panel 111 of device 110 (or lower level software that controls the display/multi touch panel) in order to cause information or graphics to be displayed, and/or receive touch events indicating user input. However, if the UI APIs are being executed at device 100, they may not be able to communicate with a display/multi touch panel 111, as device 100 may not include such an element. While tester device 100 can include a display 101, it can be of a different type than the display of the subject device 110. Furthermore, device 100 need not include any multi touch sensor panel.

Thus, device simulator 203 can be used to simulate the display and/or multi touch sensor panel of device 110 at device 100. The device simulator can provide for UI APIs 201 the same type of interface(s) that these APIs would communicate with in subject device 110 in order to connect to display/multi-touch panel 111. Device simulator 203 can cause a window 104 (see Fig. 1) to be displayed at the display 101 of device 100. Device simulator can output in window 101 the same or similar graphics that would have

been output by device 110, had it been running the software to be tested 202 and UI APIs 201. Thus, window 104 can be a simulation of the display of device 110.

Similarly, device simulator 203 can take in user input from a user of device 100 and convert it to a type that would have been received from a user of device 110. Thus, the device simulator can take in input provided through the interface devices of device 100 (e.g., keyboard 102 and mouse 103) and convert it to input that would have been produced by a multi-touch sensor panel. More details as to how the device simulator achieves this conversion are provided below.

In some embodiments, the device simulator can also simulate other input/output functionalities of device 110, such as sounds, a microphone, power or other buttons, a light sensor, an acceleration sensor, etc.

In some embodiments, tester device 100 and subject device 110 can use different types of processors with different instruction sets. In such cases, the software to be tested 202 and UI APIs can each include two different versions, one intended for execution at device 100 and the other at device 110. The two versions can be the results of compiling the same or similar high level code into the two different instruction sets associated with devices 100 and 110 (for the purposes of this example, high level code can include any code at a higher level than assembly and machine code). Thus, device 100 can be used to test the high level code of the software to be tested 202. This can be sufficient if the compilers for devices 100 and 110 do not introduce any errors or inconsistencies.

Software development kit (SDK) 204 can also be executed at device 100. The SDK can be used to develop the software to be tested 202. Furthermore, UI APIs (201) and device simulator (203) can be considered a part of the SDK used for the testing of software developed using the SDK. In alternative embodiments, no SDK needs to run on device 100. In these embodiments, device 100 can be used for testing purposes and not necessarily for software development.

In some embodiments, device 100 need not be used for testing or software development at all. Instead, it can be used to simply execute software intended for device 110 and provide a simulation of device 110. For example, an embodiment of the

invention can be used to provide a demonstration of the operation of a multi-touch enabled device so that a user can decide whether to purchase that device.

As noted above, the simulating software can take in single pointing input, or single pointing gestures issued from the user (such as, for example, gestures input by a mouse) and convert it to multi-touch gesture inputs. The simulating software can also allow the user to use keyboard keys to give the user additional control over the resulting multi-touch gesture inputs. The conversion from user input to multi-touch gesture inputs can be performed according to predefined rules.

Ordinarily, multi-touch gestures can be performed by placement of fingers, palms, various other parts of the human body, or objects (e.g., stylus or pens) on or near a multi-touch sensor panel. Some embodiments of the present invention can allow a user to enter all of the above types of simulated gestures. One easily performed group of gestures involves placement and movement of two or more finger tips on or near the surface of a touch sensor panel.

While a user is entering simulated multi-touch gesture inputs, the device simulator 203 can cause markers to appear and move across the simulated subject device screen (i.e., window 104) to indicate to the user the type of gesture he/she is entering using the mouse and keyboard (or other interfaces of device 100). These markers can be, for example, small circles representing fingertips pressing against a multi-touch panel. The markers are discussed in more detail below.

In some embodiments, a user can begin a multi-touch gesture simulation by entering a starting position. Figs 3A and 3B show two examples of entering such a position. Figs. 3A and 3B are related to gestures performed by moving two touch points, such as finger tips. Thus, a starting position defining the initial positions of two finger tips may need to be entered.

Figs 3A and 3B show simulation windows 300 and 301 which are intended to simulate the screen and/or multi touch panel of subject device 110. In some embodiments, the screen and the multi-touch panel are superimposed, so they can be shown in the same window. Thus, windows 300 and 301 can be similar to window 104 of Fig. 1.

Windows 300 and 301 show an initial placement stage of entering a gesture. The initial placement stage can be initialized in various ways, such as by pressing a keyboard key, clicking on a mouse button (not shown) or simply moving a mouse cursor over the simulation window (300 or 301). Circles 302-305 represent the positions of touch inputs. In other words, they represent the positions of virtual fingertips that are touching the simulated screen/multi-touch panel.

In a first alternative (illustrated in Fig. 3A), a first touch (302) can follow the mouse pointer (308). A second touch can be placed at a fixed predefined displacement from the first touch. For example, second touch 303 can be displaced from first touch 302 by predefined vector 306. Vector 306 can, for example, be some default value or it can be previously defined by the user. Initially, the user can move cursor 308 around window 300 and subsequently cause movements of touches 302 and 303. The user can thus find desirable positions for these touches, and indicate his/her desired initial position of the touches (this can be done by, for example, clicking a mouse button). Thus, the user can specify a desired starting position that includes two touches while only using a single pointing input device (e.g., a mouse).

In a second alternative, instead of a predefined vector 306, a predefined middle point 307 can be used. The user can again position a first touch (304) using the mouse pointer (309). In this alternative, the second touch (305) can be positioned in a mirror or symmetrical position from that of the first touch with respect to middle point 307. In other words, if the displacement from the middle point to the first touch defines vector 310, then the position of second touch 305 is such that the displacement between the second touch and the middle point defines the same vector (310). Again, the user can move the cursor around to determine a desirable position and indicate the desirable starting position (e.g., by clicking on a mouse button). Again, the middle point 307 can be entered by the user, or a default value (e.g., the middle of the window) can be used.

Various embodiments can utilize either of the above discussed alternatives for entering a starting position. Some embodiments can implement both alternatives and allow the user to choose between them (e.g., by pressing or clicking on a button).

In some embodiments, a user may switch between the two alternatives while manipulating the touches. For example, the user may start out with the Fig. 3A alternative, and displace touches 302 and 303 to a desired first set of locations. The user can then switch to the second alternative (e.g., by pressing a keyboard key). Once the second alternative is activated, the first set of locations can be used to define the middle point. For example, the middle point can be defined as the point between the locations of touches 302 and 303 of the first set of locations. Thus, the user can easily define a desired middle point and proceed to choose the starting locations using the Fig. 3B alternative.

In addition, the user can start with the Fig. 3B alternative in order to define a first set of locations for touches 304 and 305. The user can then switch to the Fig. 3A alternative. The first set of locations can be used to define the vector 306 for the Fig. 3A alternative. The user can then use the Fig. 3A alternative to define the actual initial locations.

In both alternatives, the device simulator can indicate the positioning of touches 302-304 in the simulation window by, for example, showing small semi-transparent circles indicating the positions of touches. The position of the middle point can also be indicated in the simulation window. The method of positioning shown in Fig. 3A can be referred to as parallel positioning, and the method of Fig. 3B, as mirrored positioning.

A person of skill in the art would recognize that the teachings discussed above in connection with Figs 3A and 3B can be applied for defining positions of more than two touches. For example, multiple touches can be defined as being displaced from touch 302 according to different predefined vectors. In addition, or alternatively, multiple touches can be disposed around a circle having a radius equal to the distance between touch 304 and the middle point (307). Movement of touch 304 can then move these touches by expanding, contracting or turning the circle.

Figs 3A and 3B and the discussion above describe defining an initial position of two or more touches. However, a gesture need not be defined by only its initial position. A gesture may also require some movement from the initial position as well. Thus, a multi-touch gesture may require movement of the touches. Figs. 4A and 4B

show a scheme for defining movement of touches after their initial positions have been defined.

As noted above, the desired initial position can be indicated by the user by clicking a mouse button. In some embodiments, movement can be defined by keeping the mouse button clicked (or down) while moving the mouse.

Movement can be defined in a manner similar to that of defining the initial position. Thus, Fig. 4A illustrates a scheme for defining movement that is similar to the scheme for defining an initial position shown in Fig. 3A. Accordingly, the scheme of Fig. 4A can be referred to as parallel movement definition. Positions 402 and 403 can represent the initial positions of two touches as defined by the user. As noted above, these initial positions can be entered using either or both of the methods discussed above in connection with Figs 3A and 3B. Alternatively, other methods for entering initial positions can be used. After setting the initial positions, the user can, while keeping the mouse button pressed, lead the mouse along path 410. As a result, the device simulator can lead the graphical representation of the touch that starts at position 402 along path 410 as well, until it reaches position 402'. The device simulator can also move the other touch (the one starting at position 403) along a similar path 411 until it reaches position 403'. Thus, as was the case with Fig. 3A, while one touch is being moved by the mouse cursor, the other touch is moved by the simulator so that it stays at a predefined displacement from the touch being moved by the mouse cursor. The displacement vector can be defined by the initial positioning of the touches (i.e., it can be the vector between positions 402 and 403).

One difference between the schemes of Figs 3A and 4A is that during the movement of Fig. 4A, the device simulator can track the movement of both touches, convert it into a proper data format and send it to UI APIs 201 as a gesture. On the other hand, movement during the process of Fig. 3A (e.g., before the mouse button has been pressed down) need not be tracked as that process can be used to define an initial position only and not a particular movement path.

Fig. 4B illustrates a scheme for defining movement that is similar to the scheme for defining an initial position shown in Fig. 3B. In other words, Fig. 4B may

represent mirrored movement definition. In Fig. 4B, two touches start in positions 404 and 405 respectively. The touch at position 404 (the first touch) can be moved by movement of cursor 409 to position 404' along path 414. In some embodiments, the cursor is moved while the mouse button is pressed.

The device simulator can move the touch that starts at position 405 (the second touch) from position 405 to position 405' in such a manner that the position of the second touch is mirrored from that of the first touch across from middle point 407. Thus, the second touch may travel along path 415. Middle point 407 can be defined in accordance with the initial position of the two touches. Thus, it can be the middle point between initial positions 404 and 405 (as shown). Again, the device simulator can track the movement of both touches, convert it into proper data format and send it to UI APIs 201.

Some embodiments may offer both the methods of Figs 4A and 4B for defining movement and allow a user to switch between them by pressing keyboard keys. In some embodiments, the movement definition schemes of Figs 4A and 4B can be used regardless of how the initial positions were defined. Thus, for example, the initial positions of two touches can be defined according to the scheme of Fig. 3A, while the movements of the touches can be defined according to the scheme of Fig. 4B.

In some embodiments, a user can switch between the schemes of Figs 4A and 4B while in the middle of defining a gesture. Thus, part of a gesture can be defined according to the scheme of Fig. 4A and another part according to the scheme of Fig. 4B. The methods of Figs. 4A and 4B can be used to define gestures featuring more than two touches in the manner discussed above with reference to Figs 3A and 3B.

The above discussed methods can be useful for easily defining certain types of gestures that are used in certain multi-touch enabled devices. These gestures can include, for example, dragging two fingers in parallel, pinching and expanding two fingers, turning two fingers (as if turning an invisible knob), etc. However, these methods may not be able to define all possible gestures that utilize two or more fingers. This need not be an impediment, because definition of all possible gestures may not be needed.

Only definition of gestures considered meaningful by the simulated device (i.e., subject device 110) and/or the software to be tested may need to be simulated.

Nevertheless, Fig 5 shows another method for simulating gestures which allows for greater flexibility. The method of Fig 5 can be provided by various embodiments as an exclusive method of gesture entry or as an alternative to one or more of the methods discussed above. Fig. 5 includes screens 501, 502 and 503 which can show different stages of defining a multi touch gesture.

According to the scheme of Fig. 5, a multi touch gesture can be defined by separately defining multiple single touch gesture components. Initially a first component may be defined by moving a single touch. More specifically, an initial position 505 of a single touch can be selected by, for example, placing mouse cursor 504 at that position and pressing a mouse button. Then a gesture can be defined by, for example, moving the mouse while the mouse button is pressed and releasing the mouse button at the end of the gesture. Thus, the gesture may involve starting a touch at position 505, moving the touch along path 506 and ending it at position 505'.

Thus, one component single touch gesture of a multi-touch gesture can be defined. One or more additional components can be subsequently defined in a similar manner. For example, with reference to screen 502, a second gesture component can be defined after the first one by initially clicking the mouse at position 506 and then moving it along a path 507 to position 506'. In some embodiments, while a second or subsequent gesture component is being defined, one or more previously defined gesture components can be "played back" while the subsequent component is being defined. This can assist the user in defining the relevant component, as the gesture being defined assumes that all components are performed at least partially simultaneously. Thus, while the user is defining the second component by moving the cursor from position 506 to position 506', animation 508 of another touch being moved from position 505 to position 505' can be simultaneously displayed by the device simulator.

After the second gesture component is entered, a third gesture component can be entered. The third gesture component can involve moving a cursor from position 509 to position 509' along path 510. Similarly, animations 511 and 512 of the two

previously entered gesture components can be "played back" while the third gesture component is being entered.

Embodiments of the present invention can allow any number of gesture components to be thus entered. In some embodiments, the number of gesture components that can be entered can be limited in relation to the number of fingers a user of the subject device 110 can be expected to use to enter a gesture. Various embodiments can also allow one or more erroneously entered gesture components to be re-entered or deleted.

Once the user has entered a desired number of gesture components, the user can indicate so (e.g., by clicking on a designated button). At this point the device simulator can compose a single multi touch gesture by superimposing all gesture components (i.e., performing them simultaneously). Thus, based on the components discussed in connection with Fig. 5, the device simulator can create a multi-touch gesture that involves dragging a leftmost finger up while dragging two right fingers down.

In some embodiments, the device simulator can normalize the various gesture components. More specifically, the device simulator can adjust the speed of the various components so all gesture components can begin and end simultaneously. In alternative embodiments, the speed may not be adjusted, so that some components can end before others. In still other embodiments, users can be allowed to enter gesture components that begin after other gesture components begin.

Fig 6 is a diagram of another exemplary method for defining gestures according to some embodiments of the invention. Similar to Fig. 5, elements 601 and 602 show different stages of the simulation window 104 when defining a gesture. Initially, the user can define a static touch by placing the mouse cursor 605 at position 603 and clicking a button. The user can subsequently define a moving touch by, for example, clicking on the mouse cursor at position 604 and moving the mouse cursor to position 604' along path 606. The resulting gesture may represent keeping one finger pressed at position 603 without moving it while moving another finger from position 604 to position 604' along path 605. Alternatively, the static touch can be defined after the dynamic touch or more than one static and/or dynamic touches can be defined. The method of Fig. 6 can be offered as a different mode of entering a multi-touch gesture and may be

activated by a respective control key or mouse clickable button. Alternatively, the method of Fig. 6 can be executed as a specific case of the method discussed above in connection with Fig. 5.

Fig. 7 is a diagram showing several exemplary simulated multi-touch gestures that may be input using a single pointing device according to some embodiments of this invention. Example 701 shows a pinch. Example 702 shows a reverse pinch. Example 703 shows a rotation. Example 704 shows a case where the center of rotation 705 is chosen at a position different than the center of the simulated panel. A person of skill in the art would recognize that all the examples of Fig. 7 can be implemented using the methods discussed above.

A person of skill in the art would recognize that, in the addition to the above, other methods for entering multi-touch gestures may be used. For example, a shape of a touch outline can be entered, by for example tracing it with a mouse or selecting from predefined choices. The shape can signify a more complex touch event than simply touching the screen with a finger tip. It can, for example, signify touching the screen with a palm, or placing an object on the screen. Once the shape has been entered, it can be moved around by moving a mouse cursor in order to define a multi-touch gesture.

While the above discussion centers on the case in which the tester device features only a single pointing device (such as a mouse), in some embodiments the tester device can feature a multi touch panel as well. For example, the tester device can be a laptop featuring a multi-touch enabled trackpad. The subject device can include a multi-touch panel that is combined with a display (thus allowing a user to enter multi-touch inputs by interacting with the surface of the display). The tester device can simulate the subject device by providing a simulation of the subjects device's display in the simulation window 104 of the tester device's monitor 101, while allowing a user of the tester device to enter multi-touch inputs using the tester device's track pad. The tester device can indicate simulated locations of touches in the simulation window (e.g., by showing small circles in the simulation window) while the user is entering touches through the touchpad.

While some of the above discussed embodiments relate to converting single point gesture inputs into multi-touch gesture inputs, the invention need not be thus limited. More generally, embodiments of the invention can relate to converting single point inputs into multi-point inputs. Multi-point inputs can include multi-touch inputs, but can also include other types of inputs such as, for example, the multi-proximity inputs discussed by U.S. Pat. Application No. 11/649,998.

Although the present invention has been fully described in connection with embodiments thereof with reference to the accompanying drawings, it is to be noted that various changes and modifications will become apparent to those skilled in the art. Such changes and modifications are to be understood as being included within the scope of the present invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A system for simulating multi-point gestures on a multi- point sensor panel, the system comprising:

a display for displaying a representation of the multi- point sensor panel;

a single pointing user input device; and

a device simulator, the device simulator configured to receive an input from the single pointing user input device and convert it into a multi-point gesture input according to predefined conversion rules.

2. The system of claim 1, wherein the single pointing user input device is a mouse.

3. The system of claim 1, wherein the system further includes a processor configured to execute software intended to be executed at a multi- point enabled device including the sensor panel, the device simulator being further configured to send the converted multi- point gesture input to the software in a format identical to the format in which the software would have received the multi- point gesture input had it been executing at the multi- point enabled device.

4. The system of claim 1, wherein the input from the single pointing user input device is defined by a path followed by a cursor controlled by the single pointing user input device.

5. The system of claim 4, wherein the converted multi- point gesture input is defined by at least two distinct point gesture inputs, the first point gesture input being defined by the path followed by the cursor controlled by the single pointing device and at least one other point gesture input being defined by a path derived from the path followed by the cursor controlled by the single pointing device.

6. The system of claim 5, wherein the at least one other point gesture input is defined by a path followed by a point that is displaced from the cursor controlled by the single pointing device by a predefined vector.

7. The system of claim 5, wherein the at least one other point gesture is defined by a path followed by a point that is in a position symmetrical to the position of the cursor controlled by the single pointing device with respect to a predefined point.

8. The system of claim 1, wherein the device simulator is configured to receive as input a plurality of single point gesture inputs entered consecutively through the single pointing device, and to convert the plurality of single point gesture inputs to the multi-point gesture input by:

combining the plurality of received single point gesture inputs into the multi-point gesture input, so that the multi- point gesture input represents an at least partially simultaneous performance of a plurality of single point gestures defined by the plurality of single point gesture inputs.

9. The system of claim 1, further comprising a CPU and a computer readable memory, wherein the device simulator is software stored at the computer readable memory and executed by the CPU.

10. The system of claim 9, further comprising a software development kit configured for development of software for a multi- point enabled device including the multi- point sensor panel, the software development kit being stored at the computer readable memory and executed by the CPU, the device simulator being part of the software development kit.

11. The system of claim 1, wherein the multi-point gestures are multi-touch gestures, the multi- point sensor panel is a multi-touch sensor panel and the multi-point gesture input is a multi-touch gesture input.

12. A method for simulating multi-point gestures comprising:

receiving a single tracking input from a single pointing device;

in response to the received single tracking input, displaying a visual representation of a simulated multi-point gesture, wherein the simulated multi-point gesture includes two or more simulated touch points and is at least partially based on the single tracking input.

13. The method of claim 12, further comprising:

receiving an initial positioning command from the single pointing device; and

displaying an initial position for two or more simulated touch points before the receipt of the single tracking input.

14. The method of claim 12, wherein the simulated multi-point gesture is determined by applying predefined rules to the tracking input.

15. The method of claim 12, wherein the single pointing device is a mouse.

16. The method of claim 12, wherein the single pointing device is a single-touch trackpad.

17. A computer readable medium comprising software configured for execution at a first device, the first device comprising a single pointing user input device, the software being configured to simulate multi-point gestures on a multi- point sensor panel by performing the following:

receiving a single pointing gesture through the single pointing user input device;

generating a multi- point gesture based on the single pointing gesture according to predefined conversion rule; and

displaying the multi-point gesture.

18. The computer readable medium of claim 17, wherein the generating of the multi-point gesture and the displaying of the multi-point gesture are performed in real time while the single pointing gesture is being received.

19. The computer readable medium of claim 17, wherein the software is further configured to:

receive a control signal;

based on the control signal, select one of a plurality of predefined conversion rules as the conversion rule according to which the multi-point gesture is generated.

20. The computer readable medium of claim 17, wherein the single pointing user input device is a mouse.

21. The computer readable medium of claim 17, wherein a second software is being executed at the first device, the second software being intended for execution at a multi- point enabled device including the multi- point sensor panel, the software being further configured to:

send the generated multi- point gesture to the second software in a format identical to the format in which the second software would have received a multi- point gesture had it been executing at the multi- point enabled device.

22. The computer readable medium of claim 17, wherein the single pointing gesture is defined by a path followed by a cursor controlled by the single pointing device.

23. The computer readable medium of claim 20, wherein the generated multi-point gesture is defined by at least two distinct point gestures, the first point gesture being defined by the path followed by the cursor controlled by the single pointing device and at least one other point gesture being defined by a path derived from the path followed by the cursor controlled by the single pointing device.

24. The computer readable medium of claim 23, wherein the at least one other point gesture is defined by a path followed by a point that is displaced from the cursor controlled by the single pointing device by a predefined vector.

25. The computer readable medium of claim 23, wherein the at least one other point gesture is defined by a path followed by a point that is in a position symmetrical to the position of the cursor controlled by the single pointing device with respect to a predefined point.

26. The computer readable medium of claim 17, wherein the software is part of a software development kit.

27. The computer readable medium of claim 17, wherein the multi-point gestures are multi-touch gestures, the multi- point sensor panel is a multi-touch sensor panel and the multi-point gesture input is a multi-touch gesture input.

ABSTRACT

This relates to allowing a computer system using a single pointing device to simulate multi-point gesture inputs. Simulating software can receive single pointing inputs (such as, for example, input from a mouse) and convert them to simulated multi-point gesture inputs such as finger pinches, reverse pinches, translations, rotation, and the like. The simulating software can also allow the user to use keyboard keys to give the user additional control when generating the multi- point gesture inputs.

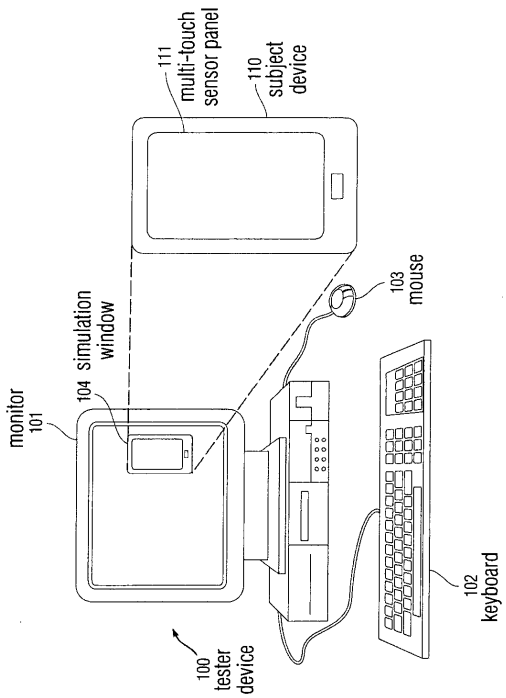


Fig. 1

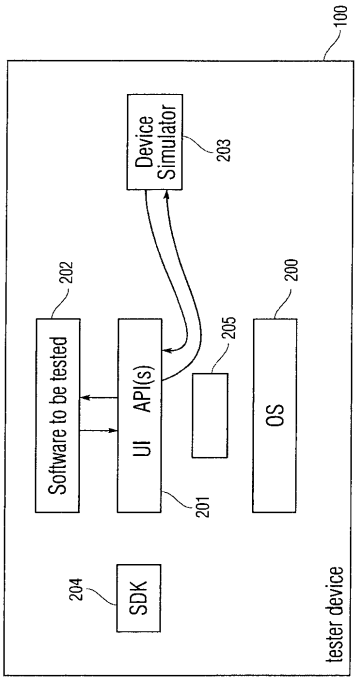


Fig. 2

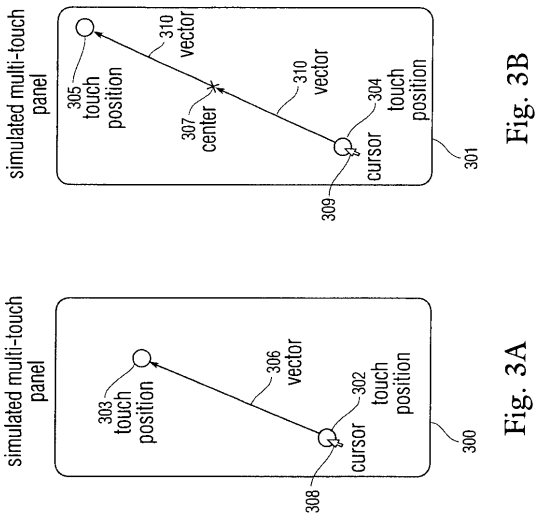


Fig. 3A

Fig. 3B

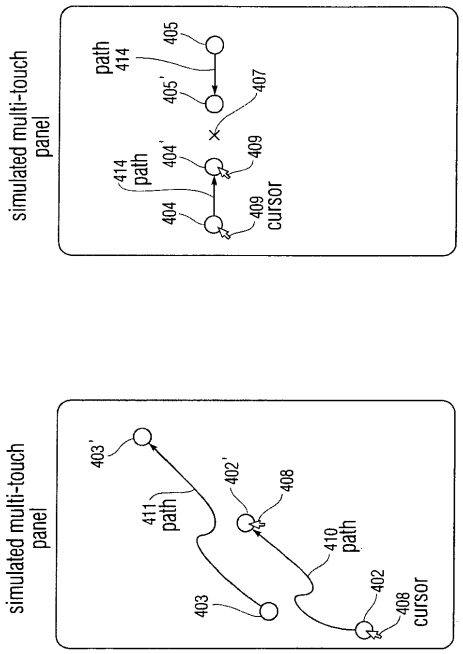


Fig. 4A

Fig. 4B

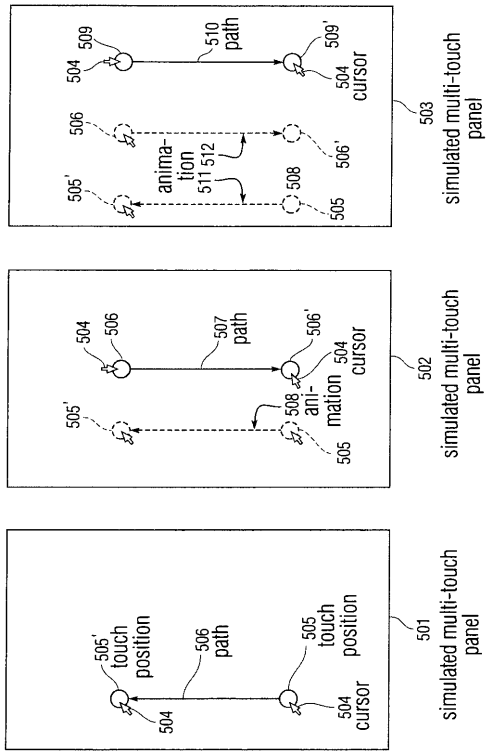


Fig. 5

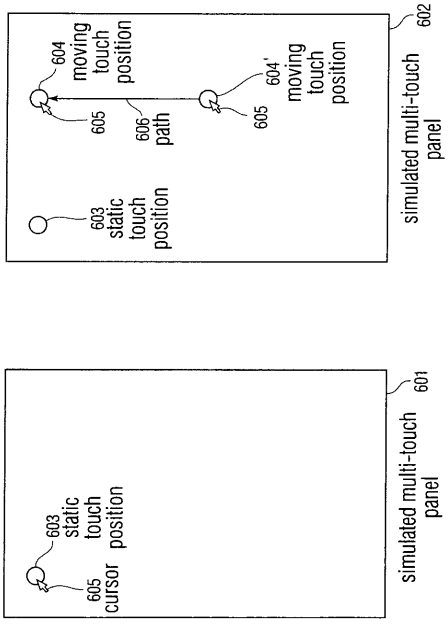


Fig. 6

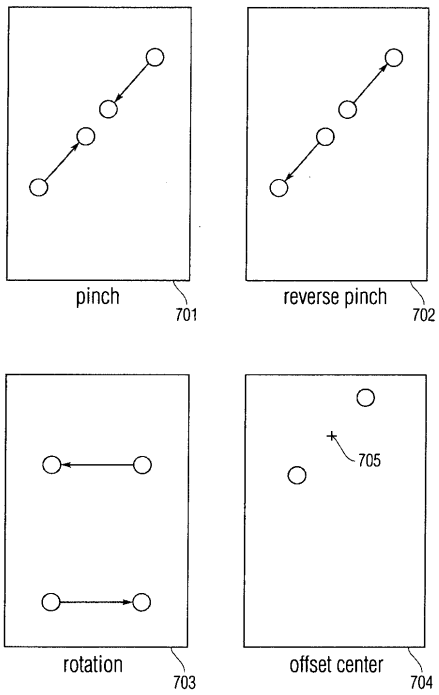

pinch
701
reverse pinch
702
rotation
703
705
offset center
704

Fig. 7