(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2010-26542
(P2010-26542A)
(43)公開日　平成22年2月4日(2010.2.4)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 13/00	(2006.01)	G06F	13/00	550C	5B050
G06F 3/14	(2006.01)	G06F	3/14	310A	5B069
G06T 11/80	(2006.01)	G06T	11/80	A	

審査請求 有　請求項の数 17　ＯＬ　（全 34 頁）

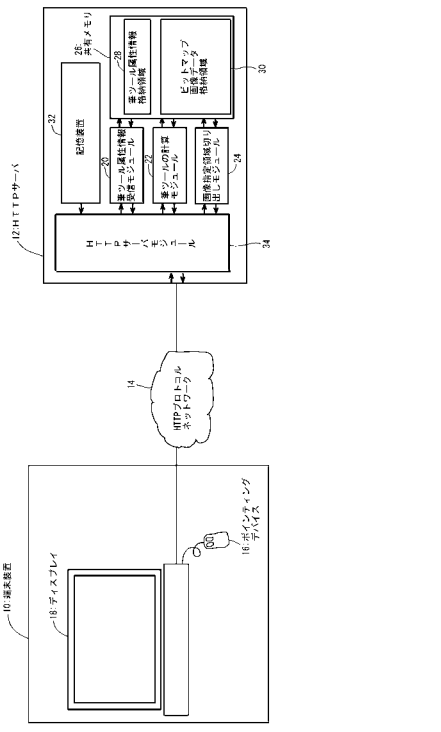
(21)出願番号　特願2007-86855(P2007-86855)
(22)出願日　平成19年3月29日(2007.3.29)
(11)特許番号　特許第4085123号(P4085123)
(45)特許公報発行日　平成20年5月14日(2008.5.14)

(71)出願人　594164379
株式会社サピエンス
東京都豊島区南大塚３－２０－６
(74)代理人　100090228
弁理士　加藤　邦彦
(72)発明者　蓮池　曜
東京都豊島区南大塚３－２０－６　株式会社サピエンス内
(72)発明者　香川　隆宣
東京都豊島区南大塚３－２０－６　株式会社サピエンス内

Ｆターム(参考)　5B050 BA06 BA18 CA07 CA08 DA09 EA09 FA02 FA05 FA09 FA13
5B069 AA01 BA03 BC02 GA08 JA02 LA09

(54)【発明の名称】画像表示更新方法およびサーバ・クライアントシステム並びに描画操作エコーバックスクリプト

(57)【要約】
【課題】ＷＥＢブラウザを利用して濃度表現を有する筆ツールにより描画を行う場合にエコーバックを高速にする。
【解決手段】ＨＴＴＰサーバ１２に描画操作対象画像と画像処理プログラムを用意する。端末装置１０からＨＴＴＰサーバ１２アクセスして描画操作対象画像をダウンロードする。操作者が端末装置１０のＷＥＢブラウザ画面で描画操作をすると、ＸＭＬＨＴＴＰリクエストによりカーソルの座標位置情報と描画半径情報がＨＴＴＰサーバ１２に送られて画像加工処理がされる。ＷＥＢブラウザは描画操作対象画像を仮想的に複数の矩形（セル）に分割し、画像加工処理で画像内容に変更があったセルを計算し、該セルの画像を要求するＨＴＴＰリクエストを発行する。ＨＴＴＰサーバ１２は要求された領域の加工済み画像を切り取って送信する。ＷＥＢブラウザはセル画像を受け取って、該当する領域に配置して表示を更新する。
【選択図】図１



【特許請求の範囲】
【請求項１】
端末装置にインストールされたＷＥＢブラウザの画面に画像を表示し、該画面上で該端末装置の操作者がポインティングデバイスで描画操作をすることにより、該ポインティングデバイスによるカーソル位置の画像の表示に濃度表現を有する筆ツールによる変更を加える画像表示更新方法であって、
ここに「濃度表現を有する筆ツール」とは、ポインティングデバイスの描画操作態様に応じて描画濃度が変化する描画を実現する描画ツールであって、
サーバに、前記操作者による描画操作対象となる画像のデータと、該描画操作に応じて該画像データに対して画像加工処理を実行する画像処理プログラムを保持管理し、
前記端末装置と前記サーバをネットワークを介して相互に通信可能な状態に設定し、
前記端末装置のＷＥＢブラウザが前記サーバに対しＨＴＭＬドキュメントをＨＴＴＰリクエストによって要求すると、該サーバは該ＷＥＢブラウザによって解釈可能な描画操作エコーバックスクリプトが含まれたＨＴＭＬドキュメントを該端末装置に該ＨＴＴＰリクエストに対する応答として返信し、該端末装置のＷＥＢブラウザは該ＨＴＭＬドキュメントを受信して、該ＷＥＢブラウザに前記描画操作エコーバックスクリプトを付加し、
前記ＨＴＭＬドキュメントは、前処理として、前記サーバが保持管理する前記描画操作対象画像をＨＴＴＰリクエストによって該サーバに要求し、その返信として受け取った画像データによる画像を前記ＷＥＢブラウザの画面に表示し、
前記端末装置のＷＥＢブラウザに付加された前記描画操作エコーバックスクリプトは、前記画像が表示された画面上で前記描画操作がされたときに、該描画操作に応じたカーソル位置の座標情報をＸＭＬＨＴＴＰリクエストによって前記サーバに送信し、
前記サーバの画像処理プログラムは、受信した該カーソル位置の座標情報と、別途取得しまたは初めから保持する筆の描画サイズの情報に基づき、自らが保持管理する前記描画操作対象画像に対して、前記濃度表現を有する筆ツールによる画像加工処理を実行し、
前記描画操作エコーバックスクリプトは前記カーソル位置の座標情報を送信後に、前記ＷＥＢブラウザの画面に表示された画像を複数の領域に分割する矩形領域のうち前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を取得し、かつ該単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信し、
前記サーバは該単一または複数のＨＴＴＰリクエストを受信して、前記ＷＥＢブラウザの画面に表示された画像の該当する単一または複数の矩形領域の画像データを選択的に返信し、
前記ＷＥＢブラウザは受信した単一または複数の矩形領域の画像データによる画像を、該ＷＥＢブラウザの画面の該当する矩形領域に、それまで該矩形領域に表示されていた画像に代えて表示し、
前記ＸＭＬＨＴＴＰリクエストによるカーソル位置の座標情報の送信を、それ以前のＸＭＬＨＴＴＰリクエストにより送信されたカーソル位置の座標情報に基づき加工処理された画像が前記ブラウザの画面に表示されるの待たずに順次実行することにより、該ＷＥＢブラウザに表示されている画像があたかも操作者の筆ツールによる描画操作に反応したかのように見せる画像表示更新方法。
【請求項２】
前記描画操作エコーバックスクリプトは前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を自ら計算して取得する請求項１記載の画像表示更新方法。
【請求項３】
前記描画操作エコーバックスクリプトは前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を前記サーバの画像処理プログラムが計算した結果を受信して取得する請求項１記載の画像表示更新方法。
【請求項４】

前記描画操作エコーバックスクリプトは前記筆の描画サイズの情報を前記カーソル位置の座標情報と同時に前記サーバに送信する請求項１から３のいずれか１つに記載の画像表示更新方法。

【請求項５】

前記描画操作エコーバックスクリプトは前記筆の描画サイズの情報を、操作者により筆の描画サイズの設定操作がされたときに前記サーバに送信する請求項１から３のいずれか１つに記載の画像表示更新方法。

【請求項６】

前記サーバの画像処理プログラムは前記端末装置から受信した前記カーソル位置の座標情報と、該端末装置から該カーソル位置の座標情報と同時に受信しまたは該サーバが該カーソル位置の座標情報を受信する以前から保持している筆の描画サイズの情報に基づき前記画像加工処理を実行し、かつ該画像描画処理により描画するまたは描画した筆の描画中心位置の座標情報をＸＭＬＨＴＴＰリクエストに対する応答として前記端末装置のＷＥＢブラウザに返信し、前記描画操作エコーバックスクリプトは該サーバから受信する筆の描画中心位置の座標情報と、該サーバから該筆の描画中心位置の座標情報と同時に受信しまたは該描画操作エコーバックスクリプトが該筆の描画中心位置の座標情報を受信する以前から保持している筆の描画サイズの情報を利用して、前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を自ら計算して取得する請求項１記載の画像表示更新方法。

【請求項７】

前記筆の描画中心位置の座標情報は前記カーソル位置の座標情報そのものまたは該カーソル位置の座標情報を補間処理した座標情報である請求項６記載の画像表示更新方法。

【請求項８】

前記描画操作エコーバックスクリプトは自ら取得しかつ前記サーバを経由しない前記カーソル位置の座標情報と、該描画操作エコーバックスクリプトが該カーソル位置の座標情報を取得する以前から保持している筆の描画サイズの情報を利用して、前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を自ら計算して取得する請求項１記載の画像表示更新方法。

【請求項９】

前記画像処理プログラムは前記描画操作対象画像に対して他の端末装置から同時に描画操作がされたときに該描画操作に基づく画像加工処理を併せて実行するものであり、

前記描画操作エコーバックスクリプトはタイマー駆動により定期的にＸＭＬＨＴＴＰリクエストを発行し、

前記画像処理プログラムは前記他の端末装置からの描画操作に基づく画像加工処理に対応して、該画像処理プログラムで計算した画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報または前記描画操作エコーバックスクリプトで該矩形領域を計算するのに必要な情報を該ＸＭＬＨＴＴＰリクエストに対する応答として返信し、

前記描画操作エコーバックスクリプトは該サーバから受信して取得した情報による単一または複数の矩形領域または該サーバから受信した情報に基づき自ら計算して取得した情報による単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信し、

前記サーバは該ＨＴＴＰリクエストを受信して、前記ＷＥＢブラウザの画面に表示された画像の該当する単一または複数の矩形領域の画像データを選択的に返信し、

前記ＷＥＢブラウザは受信した単一または複数の矩形領域の画像データによる画像を、該ＷＥＢブラウザの画面の該当する矩形領域に、それまで該矩形領域に表示されていた画像に代えて表示する請求項１から８のいずれか１つに記載の画像表示更新方法。

【請求項１０】

前記サーバが保持管理する描画操作対象画像はその画像全体に相当する単一の画像ファイルまたは該画像全体を実際に矩形に分割した複数のスライス画像に相当する複数の画像ファイルで構成され、前記矩形領域は１つの画像ファイルによる画像を仮想的に複数の矩

形に分割した該仮想分割画像単位の領域である請求項１から９のいずれか１つに記載の画像表示更新方法。
【請求項１１】
前記サーバが保持管理する描画操作対象画像はその画像全体を実際に複数の矩形に分割した複数のスライス画像の画像ファイルで構成され、前記矩形領域は該スライス画像単位の領域である請求項１から９のいずれか１つに記載の画像表示更新方法。
【請求項１２】
前記端末装置の描画操作エコーバックスクリプトは、受信した単一または複数の矩形領域の画像を、前記ＷＥＢブラウザの該当する領域に、それまで該領域に表示されていた画像の手前にｚインデックスを変更することにより配置して表示する請求項１から１１のいずれか１つに記載の画像表示更新方法。
【請求項１３】
前記端末装置の描画操作エコーバックスクリプトは、前記画像が表示された画面上で前記描画操作がされたときに、１回のＸＭＬＨＴＴＰリクエストにつき、該描画操作によるカーソルの移動軌跡に応じた該カーソル位置の複数の座標位置の情報を前記サーバに送信し、かつ該複数の座標位置に基づく画像加工処理全体で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を取得し、かつ該単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信する請求項１から１２のいずれか１つに記載の画像表示更新方法。
【請求項１４】
前記ポインティングデバイスの描画操作態様が該ポインティングデバイスの移動速度である請求項１から１３のいずれか１つに記載の画像表示更新方法。
【請求項１５】
請求項１から１４のいずれか１つに記載の画像表示更新方法を実行するサーバ・クライアントシステム。
【請求項１６】
請求項１から１４のいずれか１つに記載の画像表示更新方法を前記端末装置と前記サーバ間で実行させるサーバ・クライアントシステム用プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
この発明は端末装置のＷＥＢブラウザ画面上での描画操作に基づきサーバ側に保管されている画像に対し濃度表現を有する筆ツールによる加工を施す方法において、描画操作に応じて時々刻々加工されていく画像が該描画操作から大きく遅れることなくＷＥＢブラウザの画面に反映されるようにして、端末装置の操作者が思い通りの描画操作をできるようにしたものである。
【背景技術】
【０００２】
インターネットを利用した画像共有サービスが出現している。画像共有サービスは参加者の端末装置（クライアント端末）からサーバに写真、絵、図等の画像をアップロードして保管し、該保管された画像を参加者どうしが共有できるようにしたものである。画像共有サービスではサーバに保管された画像を参加者が自己の端末装置からの操作により任意に加工できるようにしている場合がある。この画像加工機能は従来は端末装置に、ヘルパーアプリケーションプログラム、ＡｃｔｉｖｅＸ（登録商標）プログラム、ＪａｖａＡｐｐｌｅｔプログラム等の専用の画像処理プログラムをインストールすることにより実現していた。すなわち参加者が、加工しようとする画像をサーバから端末装置にダウンロードし、該端末装置での描画操作に基づき該端末装置の画像処理プログラムで該操作に応じて加工処理を行い、加工処理が済んだ画像をサーバにアップロードして再保管するようにしていた。しかしこの従来方法によれば端末装置に専用の画像処理プログラムをインストールする必要があるため、普及の妨げになっていた。

【０００３】
端末装置に専用の画像処理プログラムをインストールすることなくＷＥＢブラウザの機能を利用して画像共有サービスで画像加工処理を実現する方法として、サーバ側に画像処理プログラムを用意し、端末装置から描画操作をして、サーバ側の画像処理プログラムで該当する画像加工処理を実行する方法が考えられる。すなわち参加者は、加工処理しようとする画像をサーバから端末装置にダウンロードして該端末装置の該画像が表示されたブラウザ画面上で描画操作をし、該端末装置はサーバで該描画操作に応じた画像加工処理を実行するのに必要な情報をパラメータとして付加して該画像加工処理された画像を要求するＨＴＴＰリクエストを発行し、サーバはこのＨＴＴＰリクエストを受信して該リクエストにパラメータとして含まれる前記情報に基づき画像処理プログラムを制御して該当する画像加工処理を実行しかつ該加工処理された画像をＨＴＴＰリクエストに対する応答として返信し、端末装置は該加工処理された画像を受信してブラウザ画面に表示されている元の画像を該加工処理された画像で更新するというものである。
【０００４】
ところで画像加工処理を行っている際には、描画操作を行っている端末装置の画面上で描画操作（カーソルの移動等）に追従して加工処理結果が反映される（すなわち画面上でカーソルが通過した位置に沿って線が描かれていく等描画操作の進行に従って表示画像が順次変更されていく）必要がある。このような表示の更新は一般にエコーバックと呼ばれる。描画操作ではエコーバックの応答が遅いと描画操作している画面上で加工処理結果が即座に表示画像に反映されないため、思い通りの描画操作を行えず使い勝手の悪いものとなる。
【０００５】
エコーバックの応答の遅さは特に画像に濃度表現を有する筆ツールによる変更を加える描画操作において問題となる。ここで「濃度表現を有する筆ツール」とはポインティングデバイスの描画操作態様（マウスボタンを押したままカーソルを移動させるときの移動速度、カーソルを１箇所に留まらせたままマウスボタンを押し続ける時間等）に応じて描画濃度が変化する描画を実現する描画ツールであって、例えば筆ツール、ブラシツール、筆ブラシツール、エアブラシツール等と呼ばれるものである。例えばエアブラシツールはスプレーでインクを吹き付けたような効果を出すものであり、マウスボタンを押すことによりマウスカーソルの位置に所定の広がりで円状にインク吹き付け画像を描画する。円の中心に近いほど濃くなるように濃度分布を設定して描画することもできる。そして１箇所でマウスボタンを押し続ける時間により描画濃度をしだいに濃くすることができる。またマウスボタンを押したままカーソルを移動させた場合は、その移動速度を速くすれば薄く描画され、移動速度を遅くすれば濃く描画される。したがってエアブラシの描画操作による加工処理結果（エアブラシにより加工された画像）が即座に該描画操作を行っている画面上に反映されなければ操作者はどの位の時間マウスボタンを押し続ければよいのかあるいはどの位の速度でマウスカーソルを移動させればよいのかがわからず、思い通りの描画を行うことができない。したがってエコーバックの応答を速くすることは画像に濃度表現を有する筆ツールによる変更を加える描画操作において特に重要である。
【発明の開示】
【発明が解決しようとする課題】
【０００６】
エコーバックの応答の速さが描画操作の操作性に影響を与えることは上述のとおりであるが、端末装置に専用の画像処理プログラムをインストールして画像加工処理を行う従来方法によればエコーバックの応答を速くすることは容易であるから、問題なく画面上での描画操作に追従して加工処理結果を該画面上に即座に反映させることができる。これに対しサーバ側の画像処理プログラムを利用して加工処理を行う場合には、端末装置のブラウザ画面上で描画操作をしてサーバ側で対応する画像加工処理を行い、その加工処理結果を端末装置のブラウザ画面上に反映させるのに端末装置とサーバ間で通信が介在するので、必然的にエコーバックの応答が遅くなる。その結果後者の方法では特に、画像に濃度表現

を有する筆ツールによる変更を加える描画操作において使い勝手が悪くなることが予想される。

【０００７】

前記サーバ側の画像処理プログラムを利用して画像加工処理を行う方法においてエコーバックの応答を速くするためには、加工が行われた画像全体をサーバから送信してブラウザ画面の画像表示全体を更新するのでなく、画像全体のうち加工が行われた領域に限ってサーバから送信してブラウザ画面の該当する領域を更新するという処理が必要である。ところが端末装置で描画操作が行われたときに、該端末装置が、該描画操作により加工が行われる領域を計算したうえで、該描画操作に応じた画像加工処理を実行するのに必要な情報をパラメータとして付加して画像全体のうち該画像加工処理が行われる領域の画像を要求するＨＴＴＰリクエストを発行するのでは、加工が行われる領域の計算が済んでから画像加工処理を実行するのに必要な情報がサーバに送られることになるため、サーバでの加工開始時間が該計算に要する時間分遅れることになり、その分エコーバックの応答が遅くなる。

【０００８】

この発明は上述の点に鑑みてなされたもので、端末装置のＷＥＢブラウザ画面上での描画操作に基づきサーバ側に保管されている画像に対し濃度表現を有する筆ツールによる加工を施す方法において、描画操作に応じて時々刻々加工されていく画像が該描画操作から大きく遅れることなくＷＥＢブラウザの画面に反映されるようにして、端末装置の操作者が思い通りの描画操作をできるようにした手法を提供しようとするものである。

【課題を解決するための手段】

【０００９】

この発明の画像表示更新方法は、端末装置にインストールされたＷＥＢブラウザの画面に画像を表示し、該画面上で該端末装置の操作者がポインティングデバイスで描画操作をすることにより、該ポインティングデバイスによるカーソル位置の画像の表示に濃度表現を有する筆ツールによる変更を加える画像表示更新方法であって、サーバに、前記操作者による描画操作対象となる画像のデータと、該描画操作に応じて該画像データに対して画像加工処理を実行する画像処理プログラムを保持管理し、前記端末装置と前記サーバをネットワークを介して相互に通信可能な状態に設定し、前記端末装置のＷＥＢブラウザが前記サーバに対しＨＴＭＬドキュメントをＨＴＴＰリクエストによって要求すると、該サーバは該ＷＥＢブラウザによって解釈可能な描画操作エコーバックスクリプトが含まれたＨＴＭＬドキュメントを該端末装置に該ＨＴＴＰリクエストに対する応答として返信し、該端末装置のＷＥＢブラウザは該ＨＴＭＬドキュメントを受信して、該ＷＥＢブラウザに前記描画操作エコーバックスクリプトを付加し、前記ＨＴＭＬドキュメントは、前処理として、前記サーバが保持管理する前記描画操作対象画像をＨＴＴＰリクエストによって該サーバに要求し、その返信として受け取った画像データによる画像を前記ＷＥＢブラウザの画面に表示し、前記端末装置のＷＥＢブラウザに付加された前記描画操作エコーバックスクリプトは、前記画像が表示された画面上で前記描画操作がされたときに、該描画操作に応じたカーソル位置の座標情報をＸＭＬＨＴＴＰリクエストによって前記サーバに送信し、前記サーバの画像処理プログラムは、受信した該カーソル位置の座標情報と、別途取得しまたは初めから保持する筆の描画サイズの情報に基づき、自らが保持管理する前記描画操作対象画像に対して、前記濃度表現を有する筆ツールによる画像加工処理を実行し、前記描画操作エコーバックスクリプトは前記カーソル位置の座標情報を送信後に、前記ＷＥＢブラウザの画面に表示された画像を複数の領域に分割する矩形領域のうち前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を取得し、かつ該単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信し、前記サーバは該単一または複数のＨＴＴＰリクエストを受信して、前記ＷＥＢブラウザの画面に表示された画像の該当する単一または複数の矩形領域の画像データを選択的に返信し、前記ＷＥＢブラウザは受信した単一または複数の矩形領域の画像データによる画像を、該ＷＥＢブラウザの画面の該当する矩形領域に、それまで

該矩形領域に表示されていた画像に代えて表示し、前記ＸＭＬＨＴＴＰリクエストによるカーソル位置の座標情報の送信を、それ以前のＸＭＬＨＴＴＰリクエストにより送信されたカーソル位置の座標情報に基づき加工処理された画像が前記ブラウザの画面に表示されるの待たずに順次実行することにより、該ＷＥＢブラウザに表示されている画像があたかも操作者の筆ツールによる描画操作に反応したかのように見せるものである。

【００１０】

なおこの発明において「ＷＥＢブラウザ」とは、マイクロソフト社のＩｎｔｅｒｎｅｔ Ｅｘｐｌｏｒｅｒ（登録商標）等のＷＥＢページを閲覧するソフトウェアであり、明示的に「ＷＥＢブラウザ」と称しているもの以外に、ＷＥＢページを閲覧する機能を有しているが「ＷＥＢブラウザ」と称していないものを含むものとする。

【００１１】

この発明によれば、ＷＥＢブラウザの画面に表示された画像を複数の領域に分割する矩形領域のうち、描画操作に基づく画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の画像を選択的にサーバからＷＥＢブラウザに送信して該ＷＥＢブラウザの画面表示を更新するので、ＷＥＢブラウザの画面に表示された画像全体分の画像をサーバからＷＥＢブラウザに送信して該ＷＥＢブラウザの画面に表示された画像全体の表示を更新する場合に比べて、更新画像のダウンロードに要する時間を短縮することができる。その反面、この手法によればＷＥＢブラウザの画面に表示された画像全体の表示を更新する場合に比べて、画像内容に変更が生じる領域を計算する分の時間が必要となる。そこでこの発明では描画操作エコーバックスクリプトは画像が表示された画面上で描画操作がされたときに、該描画操作に応じたカーソル位置の座標情報をＸＭＬＨＴＴＰリクエストによってサーバにいち早く送信するようにしている（加工処理された画像を要求するＨＴＴＰリクエストはこの時点では送信せず、後に別途送信する。画像加工処理で画像内容に変更が生じる領域の計算は該カーソル位置の座標情報をサーバに送信した後に端末装置側またはサーバ側で行うことができる）。したがってサーバの画像処理プログラムは描画操作から短時間のうちに該描画操作に対応した画像加工処理を開始することができる。したがって描画操作がされたときに、加工が行われる領域の計算が済んでから画像加工処理を実行するのに必要な情報をサーバに送る場合に比べて、描画操作がされてから画像加工処理が終了するまでの時間を短縮することができる。これにより描画操作エコーバックスクリプトは前記ＸＭＬＨＴＴＰリクエストを送信した後に、加工処理された画像を要求するＨＴＴＰリクエストを送信すると、該当する加工処理された画像を少ない待ち時間で取得することができる。その結果描画操作に応じて時々刻々加工されていく画像を該描画操作から大きく遅れることなくＷＥＢブラウザの画面に反映させることが可能になり、端末装置の操作者は濃度表現を有する筆ツールを用いた描画を自分の思い通りに実行することが可能になる。

【００１２】

なお描画操作エコーバックスクリプトはＸＭＬＨＴＴＰリクエストによるカーソル位置の座標情報の送信を、それ以前のＸＭＬＨＴＴＰリクエストにより送信されたカーソル位置の座標情報に基づき加工処理された画像がブラウザの画面に表示されるの待たずに順次実行するので、サーバの画像処理プログラムは描画操作に追従して画像加工処理を実行することができる。このことも、描画操作に応じて時々刻々加工されていく画像が該描画操作から大きく遅れることなくＷＥＢブラウザの画面に反映されることに寄与している。

【００１３】

この発明の画像表示更新方法において、描画操作エコーバックスクリプトは画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を自ら計算して取得することができる。また描画操作エコーバックスクリプトは画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報をサーバの画像処理プログラムが計算した結果を受信して取得することもできる。

【００１４】

この発明の画像表示更新方法において、描画操作エコーバックスクリプトは筆の描画サ

イズの情報をカーソル位置の座標情報と同時にサーバに送信することができる。また描画操作エコーバックスクリプトは筆の描画サイズの情報を、操作者により筆の描画サイズの設定操作がされたときにサーバに送信することもできる。

【００１５】

この発明の画像表示更新方法において、サーバの画像処理プログラムは端末装置から受信したカーソル位置の座標情報と、該端末装置から該カーソル位置の座標情報と同時に受信しまたは該サーバが該カーソル位置の座標情報を受信する以前から保持している筆の描画サイズの情報に基づき画像加工処理を実行し、かつ該画像描画処理により描画するまたは描画した筆の描画中心位置の座標情報をＸＭＬＨＴＴＰリクエストに対する応答として端末装置のＷＥＢブラウザに返信し、描画操作エコーバックスクリプトは該サーバから受信する筆の描画中心位置の座標情報と、該サーバから該筆の描画中心位置の座標情報と同時に受信しまたは該描画操作エコーバックスクリプトが該筆の描画中心位置の座標情報を受信する以前から保持している筆の描画サイズの情報を利用して、画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を自ら計算して取得することができる。この場合筆の描画中心位置の座標情報はカーソル位置の座標情報そのものまたは該カーソル位置の座標情報を補間処理した座標情報とすることができる。

【００１６】

この発明の画像表示更新方法において、描画操作エコーバックスクリプトは自ら取得しかつサーバを経由しないカーソル位置の座標情報と、該描画操作エコーバックスクリプトが該カーソル位置の座標情報を取得する以前から保持している筆の描画サイズの情報を利用して、画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を自ら計算して取得することができる。

【００１７】

この発明の画像表示更新方法はさらに、他の端末装置からの描画操作により加工される画像を自端末装置の画面上に反映させることができる。すなわち画像処理プログラムは描画操作対象画像に対して他の端末装置から同時に描画操作がされたときに該描画操作に基づく画像加工処理を併せて実行し、描画操作エコーバックスクリプトはタイマー駆動により定期的にＸＭＬＨＴＴＰリクエストを発行し、画像処理プログラムは他の端末装置からの描画操作に基づく画像加工処理に対応して、該画像処理プログラムで計算した画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報または描画操作エコーバックスクリプトで該矩形領域を計算するのに必要な情報を該ＸＭＬＨＴＴＰリクエストに対する応答として返信し、描画操作エコーバックスクリプトは該サーバから受信して取得した情報による単一または複数の矩形領域または該サーバから受信した情報に基づき自ら計算して取得した情報による単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信し、サーバは該ＨＴＴＰリクエストを受信して、ＷＥＢブラウザの画面に表示された画像の該当する単一または複数の矩形領域の画像データを選択的に返信し、ＷＥＢブラウザは受信した単一または複数の矩形領域の画像データによる画像を、該ＷＥＢブラウザの画面の該当する矩形領域に、それまで該矩形領域に表示されていた画像に代えて表示する。

【００１８】

この発明の画像表示更新方法において、サーバが保持管理する描画操作対象画像はその画像全体に相当する単一の画像ファイルまたは該画像全体を実際に矩形に分割した複数のスライス画像に相当する複数の画像ファイルで構成され、前記矩形領域は１つの画像ファイルによる画像を仮想的に複数の矩形に分割した該仮想分割画像単位の領域とすることができる。またサーバが保持管理する描画操作対象画像はその画像全体を実際に複数の矩形に分割した複数のスライス画像の画像ファイルで構成され、前記矩形領域は該スライス画像単位の領域とすることもできる。

【００１９】

この発明の画像表示更新方法において、端末装置の描画操作エコーバックスクリプトは、受信した単一または複数の矩形領域の画像を、前記ＷＥＢブラウザの該当する領域に、

それまで該領域に表示されていた画像の手前にｚインデックスを変更することにより配置して表示することができる。

【００２０】

この発明の画像表示更新方法において、端末装置の描画操作エコーバックスクリプトは、画像が表示された画面上で描画操作がされたときに、１回のＸＭＬＨＴＴＰリクエストにつき、該描画操作によるカーソルの移動軌跡に応じた該カーソル位置の複数の座標位置の情報をサーバに送信し、かつ該複数の座標位置に基づく画像加工処理全体で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を取得し、かつ該単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信することができる。

【００２１】

この発明の画像表示更新方法において、ポインティングデバイスの描画操作態様は該ポインティングデバイスの移動速度や該ポインティングデバイスのボタン押下時間等とすることができる。

【００２２】

この発明のサーバ・クライアントシステムはこの発明の画像表示更新方法を実行するものである。この発明のサーバ・クライアントシステム用プログラムはこの発明の画像表示更新方法を端末装置とサーバ間で実行させるものである。

【発明を実施するための最良の形態】

【００２３】

《実施の形態１》

この発明の実施の形態を以下説明する。図１はシステム構成を示す。このシステムは端末装置（ＨＴＴＰクライアント）１０とＨＴＴＰサーバ１２をインターネット等のＨＴＴＰプロトコルネットワーク１４を介して相互に接続してサーバ・クライアントシステムとして構成されている。端末装置１０にはマウス、タブレット等のポインティングデバイス１６とディスプレイ１８が付属されている。ここではポインティングデバイス１６としてマウスを使用する場合について説明する。また端末装置１０にはＪＡＶＡＳＣＲＩＰＴ（登録商標）をはじめとするＥＣＭＡＳｃｒｉｐｔ規格に準拠したスクリプト言語を解釈できるＩｎｔｅｒｎｅｔ　Ｅｘｐｌｏｒｅｒ等の一般的なＷＥＢブラウザがインストールされている。この実施の形態では該スクリプト言語としてＪＡＶＡＳＣＲＩＰＴを使用した場合を示している。

【００２４】

ＨＴＴＰサーバ１２には画像処理プログラムがインストールされている。この画像処理プログラムは一般的なペイントソフトや画像編集（レタッチ）ソフトと同様に、様々な描画ツール（濃度表現を有する筆ツール、鉛筆、直線、曲線、塗りつぶし、消しゴム等）を使用して新規画像の作成および作成済み画像の加工を行えるものであるが、ここでは作成済み画像を描画操作対象画像として、濃度表現を有する筆ツール（以下単に「筆ツール」という）を使用して該描画操作対象画像の加工を行う場合について説明する。この実施の形態の画像処理プログラムによれば端末装置１０の操作者は筆ツールについて次の描画属性の指定を行うことができる。

（ａ）筆の色（赤、緑、青ごと）
（ｂ）筆の全体的な濃度
（ｃ）円形、楕円形などの筆の断面形状（描画形状）
（ｄ）筆の濃度分布パターン（筆の種類）
（ｅ）筆の描画サイズ
（ｆ）補間の有無

なお（ｄ）の筆の濃度分布パターンの指定で濃度表現を有する筆ツールの濃度分布パターンを指定することにより濃度表現を有する筆ツールが選択されることになる。また描画さ

れる筆の各部の濃度は「（ｂ）で指定された全体的な濃度×（ｄ）で指定された濃度分布パターン」によって決まる。この実施の形態では筆の描画形状が円形で、円の中心に近いほど濃くなるような濃度分布パターンで描画するものとする。円形の筆を指定した場合は描画サイズの指定は筆の描画半径または描画直径（この実施の形態では描画半径）の指定となる。

【００２５】

ＨＴＴＰサーバ１２には前記画像処理プログラムにより筆ツール属性情報受信モジュール２０、筆ツール計算モジュール２２、画像指定領域切り出しモジュール２４等が予め用意されている。また、ＨＴＴＰサーバ１２の共有メモリ２６には筆ツール属性情報格納領域２８と加工対象のビットマップ画像データ（描画操作対象画像）の格納領域３０が予め用意されている。ＷＥＢブラウザからのＨＴＴＰリクエストに対してサービスするＨＴＴＰサーバ１２は、各リクエストに対して基本的に独立であり、通常はメモリ共有をしない。しかしここでは描画操作のエコーバックをＷＥＢブラウザ上で実現するために、ＨＴＴＰサーバ１２は各サービスで共有メモリ２６を共有する。なおＨＴＴＰサーバ１２は全体を１台のサーバで構成するほか、その機能を分散して複数台のサーバを組織して構成することもできる。

【００２６】

筆ツール属性情報受信モジュール２０は端末装置１０から予め送られてくる筆ツールの属性情報（この実施の形態では、色、全体的な濃度、断面形状、濃度分布パターン、補間の有無）を受信し、共有メモリ２６の筆ツール属性情報格納領域２８に格納する。なおこの実施の形態では操作者によって設定される筆の属性のうち描画サイズ（描画半径）の情報は実際に描画操作がされた時にカーソル位置の座標情報とともに端末装置１０からＨＴＴＰサーバ１２に送信するようにしている。筆ツール計算モジュール２２は端末装置１０で描画操作が実行されているときに、筆ツール属性情報格納領域２８に格納されている筆ツールの属性情報（色、全体的な濃度、断面形状、濃度分布パターン、補間の有無）を取得し、かつ端末装置１０から筆ツールの座標情報および描画サイズ（描画半径）の情報を受け取り、共有メモリ２６のビットマップ画像データ格納領域３０の画像データに対して、濃度計算をしながら画像に変更を加える処理（画像加工処理）を実行する。

【００２７】

この濃度計算は例えば筆ツールによる描画操作により画像データに変更を加える領域（描画直径）内の画素ごとに、該画像データによる画素値に１-α（αは適宜の係数で、０＜α＜１）を掛けた値と筆ツールによる画素値にαを掛けた値どうしを加算することにより行われる。その結果該領域内の画素ごとに背景画像（元の画像）と筆による画像を混ぜた画素に加工され、筆による画像の奥に背景画像が透けて見える半透明の状態の画像が得られる。この濃度計算は描画操作が実行されている間中、ｏｎＭｏｕｓｅＭｏｖｅイベントによって筆ツールの検出座標が変化するごとに前回の加工結果に対して順次繰り返される。その結果同一画素に対して加工が繰り返される場合は、該画素は次第に（つまり加工回数が増えるにつれて）筆の影響が強くなりその分背景画像の影響が弱くなっていく。ここでｏｎＭｏｕｓｅＭｏｖｅイベントによる筆ツールの検出座標はマウスカーソルの移動速度が遅いときは前回の検出座標との距離が短く、該移動速度が速くなるほど前回の検出座標との距離が長くなるので、１つの画素に対して筆の描画直径が通過する間に繰り返される加工回数はマウスカーソルの移動速度が遅いときは多くなり、該移動速度が速いときは少なくなる。したがってマウスカーソルの移動速度が遅いときは筆が濃く描画され、該移動速度が速いときは筆が薄く描画される。このようにして濃度表現を有する筆による描画表現が実現される。

【００２８】

上記濃度計算はｏｎＭｏｕｓｅＭｏｖｅイベントによって筆ツールの検出座標が変化するごとに行うのに代えてタイマーイベント（ｓｅｔＩｎｔｅｒｖａｌあるいはｓｅｔＴｉｍｅｏｕｔ）により一定時間ごとに行うこともできる。タイマーイベントとしてｓｅｔＩｎｔｅｒｖａｌを使用する場合は１回設定すると一定時間ごとに繰り返し設定した濃度計

算の関数が呼び出される。これに対しｓｅｔＴｉｍｅｏｕｔを使用する場合は一定時間後に１回だけ濃度計算の関数が呼び出されるので、この呼び出された関数の中で再びｓｅｔＴｉｍｅｏｕｔを設定することによって繰り返し関数が呼ばれるようにする。タイマーイベントを使用した場合も、ｏｎＭｏｕｓｅＭｏｖｅイベントを使用した場合と同様に、マウスカーソルの移動速度が遅いときは、１つの画素に対して筆の描画直径が通過する時間が長くなるので加工回数が多くなり、その結果筆が濃く描画される。これに対しマウスカーソルの移動速度が速いときは、１つの画素に対して筆の描画直径が通過する時間が短くなるので加工回数が少なくなり、その結果筆が薄く描画される。

【００２９】

濃度表現を有する筆による描画表現は上記濃度計算以外の方法でも実現可能である。例えば筆の描画直径内の背景画像（元の画像）の飛び飛びの位置の画素を筆による画素で置き換えることにより、人の目には（つまり巨視的には）筆による画像の奥に背景画像が透けて見える半透明の状態の画像が得られる。そして描画操作が実行されている間中、筆の描画直径内について、ｏｎＭｏｕｓｅＭｏｖｅイベントによって筆ツールの検出座標が変化するごとにこの画素の置き換え箇所を順次増やしていくことにより、次第に筆の影響が強くなりその分背景画像の影響が弱くなっていく。ここでｏｎＭｏｕｓｅＭｏｖｅイベントによる筆ツールの検出座標はマウスカーソルの移動速度が遅いときは前回の検出座標との距離が短く、該移動速度が速くなるほど前回の検出座標との距離が長くなるので、筆の描画直径が通過する領域の単位面積あたりの画素の置き換え数はマウスカーソルの移動速度に応じて変化する。すなわち該置き換え数はマウスカーソルの移動速度が遅いときは多くなり、該移動速度が速くなるほど少なくなる。したがってマウスカーソルの移動速度が遅いときは筆が濃く描画され、該移動速度が速いときは筆が薄く描画される。このようにして濃度表現を有する筆による描画表現が実現される。

【００３０】

上記画素の置き換えはｏｎＭｏｕｓｅＭｏｖｅイベントによって筆ツールの検出座標が変化するごとに行うのに代えてタイマーイベントで計測した一定時間ごとに行うこともできる。これによれば筆の描画直径が通過する領域の単位面積あたりの画素の置き換え数はマウスカーソルの移動速度が遅いときは多くなり、該移動速度が速くなるほど少なくなる。したがってマウスカーソルの移動速度が遅いときは筆が濃く描画され、該移動速度が速いときは筆が薄く描画される。

【００３１】

ＨＴＴＰサーバ１２の記憶装置３２には端末装置１０からアクセスがあったときに最初に読み出されるＨＴＭＬドキュメントが記憶されている。このＨＴＭＬドキュメントには端末装置１０のＷＥＢブラウザによって実行可能な、ＪＡＶＡＳＣＲＩＰＴによって記述された描画操作エコーバックスクリプトが含まれている。ＨＴＭＬドキュメントに描画操作エコーバックスクリプトを含ませて解釈する方法としては、ＨＴＭＬドキュメントの内部にそのまま描画操作エコーバックスクリプトを記述して一括して読み込み解釈する方法（内部記述による方法）、ＨＴＭＬドキュメントの解釈中に外部ファイルとしてＨＴＴＰサーバ１２から描画操作エコーバックスクリプト部分を読み込んで全体を組み立ててから解釈する方法（外部ファイルによる方法）等がある。ＨＴＴＰサーバモジュール３４はＨＴＴＰプロトコルにより端末装置１０との通信を行う。

【００３２】

図１のシステム構成において筆ツールにより描画される画像について説明する。図２は描画操作対象画像の全体を構成するカンバス３６に対して筆ツールにより描画を行う状態を示す。なお端末装置１０のディスプレイ１８にはこのカンバス３６の全体（カンバス３６すなわち描画操作対象画像全体がディスプレイ１８の表示領域に収まる大きさの場合）または一部（カンバス３６すなわち描画操作対象画像全体がディスプレイ１８の表示領域に収まらない大きさの場合）が表示される。カンバス３６の一部が表示される場合は操作者のスクロール操作により表示領域を移動して任意の画像領域を表示することができる。いま操作者が描画操作対象画像が表示されたディスプレイ１８の画面上で、マウスカーソ

ルを図２（ａ）に示すようにカンバス３６上の例えば左上角を原点Ｏとする任意の位置Ｐ１（x_1，y_1）に位置決めしてマウスボタンを押下すると、該マウスカーソル位置Ｐ１（x_1，y_1）を中心とする半径ｒの円形に筆ツールによる画像３８が描かれる。ここでは円の中心に近いほど濃くなるように濃度分布が設定されている。マウスボタンを押下したままマウスカーソルを移動すると、図２（ｂ）に示すようにマウスカーソルの移動軌跡Ｐ１，Ｐ２，・・・，Ｐｍに追従して筆ツールによる連続した画像３８（太さ２ｒの曲線画像）が描かれていく。このとき画像３８の描画濃度はマウスカーソルの移動速度によって変化する。すなわちマウスカーソルの移動速度を速くすれば薄く描画され、遅くすれば濃く描画される。

【００３３】

図２の描画を実現するための画像加工処理はＨＴＴＰサーバ１２の筆ツール計算モジュール２２によって実現される。そしてこの画像加工処理により時々刻々加工されていく画像を端末装置１０のディスプレイ１８（ＷＥＢブラウザの画面）にエコーバックとして表示する。このとき更新速度を速くするためにカンバス３６を仮想的に矩形に分割して管理する。図３はこの矩形分割の一例を示す。図３ではカンバス３６を、その原点Ｏから横方向（ｘ軸方向）および縦方向（ｙ軸方向）にそれぞれ所定画素数ずつ区切って同一の大きさの矩形４０に分割している。ここでは個々の矩形４０を「セル」と呼ぶことにする。個々のセルの大きさはディスプレイ１８の表示領域全体の大きさに比べて十分に小さく、例えば縦・横各方向とも４８画素程度に設定することができる。カンバス３６の原点Ｏから数えたセル４０の番地を｛Ｘ，Ｙ｝（Ｘ＝０，１，２，・・・Ｍ、Ｙ＝０，１，２，・・・Ｎ）で表す。エコーバックは描画操作に基づく画像加工処理により画像内容に変更を生じたセルの画像（仮想分割画像、セル画像）をＨＴＴＰサーバ１２から端末装置１０のＷＥＢブラウザに供給することにより行う。図２（ａ）に対応する図３の例で言えば、マウスカーソルをカンバス３６上の任意の位置Ｐ１（x_1，y_1）に位置決めしてマウスボタンを押下すると、マウスカーソル位置Ｐ１（x_1，y_1）を中心とする半径ｒの円形に筆による画像３８が描かれる。その結果Ｘ座標が４，５，６，７でＹ座標が３，４，５，６である図３に太線で囲んだ領域の１４個のセルの画像内容に変更が生じる。そこでＷＥＢブラウザはＨＴＴＰリクエストによりこの変更後の１４個のセル画像を要求し、ＨＴＴＰサーバ１２は保持している画像加工処理が行われた画像からこの１４個のセル画像を切り出してＷＥＢブラウザに送信する。ＷＥＢブラウザは受信した各セル画像をディスプレイ１８に表示されている画像の該当する矩形領域に、それまで該矩形領域に表示されていた画像に代えて表示する。このようにしてエコーバックが行われる。そして以上の画像加工処理およびエコーバック処理をカーソルが移動する（画像上でカーソル座標が変化する）ごとに実行することにより描画操作に追従して画像加工処理およびエコーバックがリアルタイムで行われるようにする。このときＨＴＴＰサーバ１２からは画像内容に変更が生じたセル画像のみを送信するので、描画操作対象画像全体を送信する場合に比べて画像表示を高速に更新することができる。なお後述する数４あるいは図７の手法のように、画像内容に変更が生じる領域全体を含む１つの矩形領域が掛かる全てのセル画像（画像内容に変更が生じるセルに隣接する画像内容に変更が生じないセル画像が含まれる場合がある）をＨＴＴＰサーバに要求して画像表示を更新することもできる。

【００３４】

図１のシステムを使用して端末装置１０の操作者によるブラウザ画面からの描画操作によりＨＴＴＰサーバ１２側で描画操作対象画像に対して順次画像加工処理を行い、かつそのエコーバックとして端末装置１０のディスプレイ１８に表示されている描画操作対象画像を順次更新していく処理手順を図４を参照して説明する。はじめに実際に描画操作を開始する前の前処理について説明する。端末装置１０で操作者がＷＥＢブラウザを起動してＨＴＴＰサーバ１２（ＷＥＢサイト）へのアクセスを指示操作をすると（Ｓ１）、ＷＥＢブラウザはＨＴＴＰサーバ１２に対しＨＴＭＬドキュメントを要求するＨＴＴＰリクエストを送信する（Ｓ２）。ＨＴＴＰサーバ１２はこれを受信して、記憶装置３２から指定されたＨＴＭＬドキュメントの入ったファイルを読み出し返信する（Ｓ３）。ＨＴＴＰサー

バ１２は記憶装置３２からＨＴＭＬドキュメントの入ったファイルを読み出すのに代えて、ｃｇｉプログラムでＨＴＭＬドキュメントを生成し返信することもできる。このＨＴＭＬドキュメントには端末装置１０のＷＥＢブラウザによって実行可能な、ＪＡＶＡＳＣＲＩＰＴによる描画操作エコーバックスクリプトが格納（内部記述または外部ファイルによる）されている。ＷＥＢブラウザは送られてきた描画操作エコーバックスクリプトを含むＨＴＭＬドキュメントを解釈する（Ｓ４）。以下ＷＥＢブラウザ側の処理はこのＪＡＶＡＳＣＲＩＰＴによって実行される。

【００３５】

ＷＥＢブラウザはいわゆるペイントシステムを構成するためのメニューシステム（描画ツールの選択や属性の設定を行うためのツールボックス等）をＷＥＢブラウザの画面上に構成（表示）する（Ｓ５）。このＷＥＢブラウザ画面上へのメニューシステムの構成はＪＡＶＡＳＣＲＩＰＴによりメニューシステムをＷＥＢブラウザの画面上に構成する周知の手法を用いて実現することができる。そして複数回のＨＴＴＰリクエストでメニュー画像を読み込んでＷＥＢブラウザの画面上にメニューを組み立てることができる。

【００３６】

またＷＥＢブラウザにはＨＴＴＰサーバ１２側で設定したカンバス３６上の各セル位置を識別するために必要な情報（セルのサイズ、番地の付与方法等）が定義される。ＷＥＢブラウザはさらにＨＴＴＰサーバ１２が保持管理する描画操作対象画像をＨＴＴＰリクエストによってＨＴＴＰサーバ１２に要求する（Ｓ６）。このセルを定義し描画操作対象画像を要求する一連のＪＡＶＡＳＣＲＩＰＴのプログラムリスト例を数１に示す。

【数１】

```
//説明を簡略化するため４８０画素ｘ４８０画素の正方形のカンバス画像を生成する。

                blk= new bsBlock(ge('cCB'), 0, 0, 480, 480);

//ここの「cCB」はこのコードには上無いが、すでに用意されたカンバスを並べる土台の
//名前である。
//この関数で土台「bs」の上にカンバス画像をレイアウトして、初期のカンバス画像を
//ロードしている。ブラウザでは、カンバス画像ロード時にzインデックスによって奥行
//き情報を付加できるので、ここではその値を２としている。エアブラシなどのペイン
//ト操作の度にカンバス画像をロードして置き換えていると時間が掛かってしまうので、
//カンバス画像をさらに分割管理する。ここでの実装では､縦横10分割している。分割
//した画像をセルと呼ぶことにする。

function bsBlock(bs, lft, top, len1, len2) {
        var         i,j,cnt;
        if( !bs ) bs = document.body;
        this.Left = parseInt(lft);                              //Left:カンバスの座標左端（土台左上を原点とした座標）
        this.Top = parseInt(top);                               //Top:カンバスの座標上端（土台左上を原点とした座標）
        this.Len1 = parseInt(len1);                             //Len1:カンバスの横の大きさ
        this.Len2 = parseInt(len2);                             //Len2:カンバスの縦の大きさ
        this.div1 = Len1/48;                                    //div1:カンバスの分割数(ここでは１０)
        this.div2 = Len2/48;                                    //div2:カンバスの分割数(ここでは１０)
        this.cWid = Math.floor(this.Len / this.div);            //cWid:１セルの縦横の大きさ
        this.iNam = lft + "_" + top + "_" + len;                //カンバス画像のファイル名の組み立て処理
        this.bShow = false;
        this.sub = null;
        this.Lock = false;
        this.base = document.createElement("div");              // 制御のための土台のdivタグ作成
        this.base.style.position = 'absolute';
        this.base.style.left = this.Left;
        this.base.style.top = this.Top;
        this.base.style.width = this.Len1;
        this.base.style.height = this.Len2;
        this.ibs = document.createElement("img");               // 画像のimgタグ生成
        this.ibs.style.position = 'absolute';
        this.ibs.style.left = 0;
        this.ibs.style.top = 0;
        this.ibs.style.width = this.Len1;
        this.ibs.style.height = this.Len2;
        this.ibs.style.zIndex = 2;                              //初期画像をここで読み込む。ここでは、
                                                                //その再の奥行きは、zIndex値で２としている。
        this.ibs.src = cgi + "?GetImg&PS=" + this.iNam + "&serial=" + serCnt++ + "&usr=" + user;
                                                                //画像をダウンロードするためのHttpリクエス
                                                                //トの文字列組み立て処理と要求処理
        bs.appendChild(this.base);
        this.base.appendChild(this.ibs);
        cnt = 0;                                                //ここに続くループは、セルを管理するための変数の初期化
        this.cell = new Array;
        for(i=0; i<this.div2; i++) {
                for(j=0; j<this.div1; j++) {
                        this.cell[cnt] = null;
                        cnt++;
                }
        }
        this.celCnt = cnt;                                      //celCnt:セルの総数の代入(実装上は、１０ｘ１０で１００)
}
```

【００３７】

ＨＴＴＰサーバ１２は描画操作対象画像を要求するＨＴＴＰリクエストを受け取ると、ビットマップ画像データ格納領域３０から指定された画像ファイルを読み出し描画操作対象画像として返信する（Ｓ７）。ＷＥＢブラウザはこの描画操作対象画像を受け取って表示する（Ｓ８）。操作者はＷＥＢブラウザの画面に表示されたメニューシステムを操作して筆ツールについて色、全体的な濃度、断面形状、濃度分布パターン、描画サイズ、補間の有無の各属性を指定する（Ｓ９）。ＷＥＢブラウザは指定された筆の属性のうち描画サイズを除く属性を引数に設定してＸＭＬＨＴＴＰリクエストを発行する（Ｓ１０）。このＸＭＬＨＴＴＰリクエストを発行するＪＡＶＡＳＣＲＩＰＴのプログラムリスト例を数２に示す。

【数２】

```
function setBrush() {
        if( xreq ) {
                if( !xreq.readyState ) {                //いま、ＸＭＬＨＴＴＰリクエストが発行できることを確認
                        xreq.onreadystatechange = refXHRdata;
                                                        //状態が変化したら（すなわち、このＸＭＬＨＴＴＰリクエストの
                                                        //リクエストの返事がきたら）、refXHRdata という関数を呼ぶよう
                                                        //に設定している。この関数は、単に返事を受け取る関数。
                        xreq.open("POST", cgi + "?setCol", false);
                                                        //cgiという変数には、ＵＲＬの一部の文字列が
                                                        //（別の場所で）定義されている。
                                                        //setCol は、このコマンド名
                        xreq.setRequestHeader("Content-Type", "text/plain");
                        xreq.send("red=" + pRed         //赤色の指定
                                + "&green=" + pGrn      //緑色の指定
                                + "&blue=" + pBlu       //青色の指定
                                + "&den=" + pDen        //全体的な濃度の指定
                                + "&frm=" + tFrm        //円形、楕円形などの断面の形状の指定
                                + "&pat=" + aPat        //筆の濃度パターンの指定
                                + "&kind=" + pBrKn      //押下した座標に間隔があいた場合の補間の有無の指定
                                + "&usr=" + user);      //利用者を特定する番号の指定
                        xreq.abort();
                }
                else
                                reqSetb = true;         //ＸＭＬＨＴＴＰリクエストが発行できなかった場合は、この
                                                        //フラグを立てて、あとで、再度トライできるようにしておく。
        }
}
```

【００３８】

ＨＴＴＰサーバ１２はこのＸＭＬＨＴＴＰリクエストを受け取ってその中に含まれる筆の属性情報（色、全体的な濃度、断面形状、濃度分布パターン、補間の有無）を共有メモリ２６の筆ツール属性情報格納領域２８に保存する（Ｓ１１）。ＨＴＴＰサーバ１２はＸＭＬＨＴＴＰリクエストに対する応答としてダミー情報を返信する（Ｓ１２）。ＷＥＢブラウザはこの応答を受け取ってそのまま廃棄する（Ｓ１３）。ダミー情報に代えてエラー情報（エラーの有無を通知する情報）を返信することもできる。以上で前処理が終了し、操作者による描画操作が可能な状態となる。

【００３９】

次に、以上のようにして前処理が終了した状態で操作者が実際に描画操作したときの処理について説明する。操作者がＷＥＢブラウザに表示された描画操作対象画像上の描画したい場所にマウスカーソルを位置決めしてマウスボタンを押下すると（Ｓ１４）、これをトリガにしてＷＥＢブラウザは該画像上での筆ツールの座標（マウスカーソルの座標）および前記属性設定で指定された筆ツールの描画サイズ（描画半径）を引数に設定してＸＭＬＨＴＴＰリクエストを発行する（Ｓ１５）。このＸＭＬＨＴＴＰリクエストを発行するＪＡＶＡＳＣＲＩＰＴのプログラムリスト例を数３に示す。

【数３】

```
//サーバへ
//　　筆ツール（マウスカーソル）の座標、
//　　筆ツールの半径、
//　　ユーザＩＤ
//を送信し、筆ツールの計算モジュール２２が実際画像加工した
//　　中心座標、
//　　半径
//の取得をその返信として要求し、さらに、その返信応答があった場合に起動
//する関数（xgetDrawPos）を定義している。ただし、この要求する中心座標、
//半径はこのプログラムコード上に現れず、ここで定義された関数内に現れる。
function sendPoint(xst, yst) {
                                                        //xst,yst:座標情報が格納されている
        if( xreq ) {
                if( (reqCnt < 4 ) && (!xreq.readyState || (!msie && !xtout)) ) {
                        xreq.onreadystatechange = xgetDrawPos;   //その結果は関数「xgetDrawPos」
                                                                 //で処理することを設定
                        xreq.open("POST", cgi + "?putAbrh", true);
                        xreq.setRequestHeader("Content-Type", "text/plain");
                        xreq.send("x=" + xst + "&y=" + yst + "&rad=" + pRad + "&usr=" + user);
                                                                 //xmlhttpリクエストの発行
                                                                 //xst: 渡されてきた筆ツールのＸ座標
                                                                 //yst: 渡されてきた筆ツールのＹ座標
                                                                 //pRad：グローバル変数としてもっている半径の値
                                                                 //user：ユーザＩＤ
                }
        }
}
```

【００４０】

ＨＴＴＰサーバ１２はＸＭＬＨＴＴＰリクエストを受け取ると、先に保存されている筆ツールの属性（色、全体的な濃度、断面形状、濃度分布パターン、補間の有無）と、今回送られてきた筆ツールの座標および描画半径に基づき、画像処理プログラムを使用して、自らが保持管理する描画操作対象画像に対して筆ツールによる画像加工処理を実行する。このようにして画像処理プログラムは描画操作から短時間のうちに（つまり画像加工処理によって画像内容に変更が生じるセルの計算結果が出るのを待つことなく）該描画操作に対応した画像加工処理を開始することができる。そしてＨＴＴＰサーバ１２は実際に加工した（またはこれから加工しようとする）該画像加工処理の中心座標（筆ツールの座標）と描画半径をＸＭＬＨＴＴＰリクエストに対する応答として返信する（Ｓ１６）。なお属性設定で補間の有無について「有り」が指示された場合であって前回のＸＭＬＨＴＴＰリクエストで送られてきた筆ツールの座標と今回のＸＭＬＨＴＴＰリクエストで送られてきた筆ツールの座標との間が離れている場合は該座標間を補間（直線補間、スプライン補間等）して画像加工処理を施し、かつ該補間した複数の中心座標と描画半径を返信する。

【００４１】

ＷＥＢブラウザは画像加工処理の中心座標と描画半径を受け取ると、これらの情報に基づき、描画操作対象画像を仮想的に分割するいずれのセルの画像内容に画像加工処理による変更が生じるかを計算する。そして計算により求められたセルを識別する番号（例えば図３のＸ，Ｙ座標を組み合わせた数値）を引数として該当するセル画像を要求するＨＴＴＰリクエストを発行する（Ｓ１７）。この際ＨＴＴＰリクエストの仕様上、返信として受け取る予定のセル画像の表示位置（ブラウザ画面の原点を基準とした位置）を指定する。なおＨＴＴＰリクエストは要求するセル画像ごとに発行されるので、このＨＴＴＰリクエストは複数になる場合がある。

【００４２】

ＨＴＴＰサーバ１２はＨＴＴＰリクエストを受け取ると、自らが保持管理する描画操作対象画像から該リクエストで指示されるセルの識別番号に対応する矩形領域の画像を切り取って返信する（Ｓ１８）。ＷＥＢブラウザは送られてきたセル画像を、ブラウザの画面の該当する領域（画像要求時に指定した表示位置）に現在表示している画像の手前にｚインデックスの機能で表示する。すなわち１つのセル画像の受信を終了したら、該新たなセル画像のｚインデックスを現在その領域に表示されている画像のｚインデックスよりも大きい値に設定することにより、該新たなセル画像を手前に配置して表示する（Ｓ１９）。以上の処理によればＨＴＴＰサーバ１２からは画像内容に変更が生じたセルの画像のみを送信するので、描画操作対象画像全体を送信する場合に比べて画像表示を高速に更新することができる。画像内容に変更が生じるセルを計算して該計算に基づきＨＴＴＰリクエス

トを発行し、さらに該ＨＴＴＰリクエストに対する応答として送られてきたセル画像を現在表示している画像の手前にｚインデックスの機能で表示するＪＡＶＡＳＣＲＩＰＴのプログラムリスト例を数４～数５に示す。なお数４では計算を容易にするために、画像内容に変更が生じる全てのセルのみを計算するのに代えて、画像内容に変更が生じる全てのセルを含む最小の矩形領域に含まれるセルを計算している。すなわち図３の例で言えば、Ｘ座標が４，５，６，７でＹ座標が３，４，５，６である矩形領域（後述する図７（ｂ）に太線で囲んだ領域と同じ）に含まれる１６個のセルを計算している。その結果この１６個のセルの中には画像内容に変更が生じたセルに隣接する画像内容に変更が生じていないセル｛７，３｝、｛７，６｝も含まれることになるが、この発明はＷＥＢブラウザが画像内容に変更が生じていないセルの画像を一部含んで要求し画像表示を更新する場合を排除するものではない。この場合も描画操作対象画像全体を送信する場合に比べて画像表示を高速に更新することができる。

【数４】

```
//変更座標／筆の半径取得と更新処理の呼び出し
function xgetDrawPos() {
        if (xreq.readyState == 4) {                             //この値が４とは、リクエスト対する返事の
                                                                //読み込みが終わったことをあらわす
                xtout = 0;
                if (xreq.status == 200) {
                        var             _obj,_pos,_rad,x,y,r;
                        _obj = xreq.responseText.split('¥n');
                        _pos = _obj[0].split(',');              //座標値取得
                        _rad = _obj[1] ? _obj[1].split(',') : ''; //筆の直径取得
                        x = parseInt(_pos[0]);
                        y = parseInt(_pos[1]);
                        r = parseInt(_rad[0]);
                        blk.Active(x, y, r);                    //更新処理（関数）の呼び出し部分
                }
                xreq.abort();
                if( reqUpSend ) {
                        reqUpSend = false;
                        sendPointUp();
                }
        }
}


//変更点の座標と筆の直径から変更の加わった可能性のあるセルを求めて、画像を要求する関数を呼び出す関数
bsBlock.prototype.Active = function(xp, yp, rad) {
        var        i,j,x,y,cnt,ss;
        xs = Math.floor((xp - rad) / this.cWid);                //floor():最も小さい整数を返す-切り下げ
        ys = Math.floor((yp - rad) / this.cWid);
        xe = Math.ceil((xp + rad) / this.cWid);                 //ceil():最も大きい整数を返す-切り上げ
        ye = Math.ceil((yp + rad) / this.cWid);
        if( !this.Lock && (this.bShow || (xe > 0 && ye > 0 && xs < this.div && ys < this.div)) ) {
                this.bShow = false;
                for(i=0; i<this.div; i++) {
                        for(j=0; j<this.div; j++) {
                                cnt = i * this.div + j;         //セルは１次元で管理しているので１次元に変換
                                if( i >= xs &&  i < xe && j >= ys && j < ye ) {
                                        if( !this.cell[cnt] ) {    //初回はイメージタグを生成して画像を要求する
                                                x = this.Left + this.cWid*i;
                                                y = this.Top + this.cWid*j;
                                                ss = x + "_" + y + "_" + this.cWid;
                                                this.cell[cnt] = new bsCell(this.base, this.cWid*i, this.cWid*j, this.cWid, ss);
                                                                //この行の new でタグを生成している
                                        }
                                        else if( !this.cell[cnt].Lock )
                                                this.cell[cnt].loadImg();
                                                                //２回目以降は、イメージタグがすでに出来ているので
                                                                //単に画像を要求する関数を呼んでいる
                                        else
                                                this.cell[cnt].reqLd = true;
                                                               //画像更新最中なので読み込み要求を後回しにする
                                        this.bShow = true;
                                }
                                else if( this.cell[cnt] && this.cell[cnt].Lock )
                                        this.bShow = true;
                        }
                }
                if( !this.bShow )
                        this.reqDraw = true;
        }
};

//初回のセル画像要求。
//実装上は、セルＡとセルＢのダブルバッファをしていて、初回はセルＡにロ
//ードしている。
function bsCell(bs, lft, top, len, ist) {                       //セルAに画像読み込み。Z-INDEX=4
        if( !bs ) bs = document.body;
        this.Len = parseInt(len);
        this.xp = parseInt(lft) + Math.floor(this.Len/2);
        this.yp = parseInt(top) + Math.floor(this.Len/2);
        this.iNam = ist;
        this.pNde = bs;
        this.Lock = true;
        this.reqLd = false;
        this.Lcnt = 0;
        this.Bside = null;
        this.Aside = document.createElement("img");             //セルAの画像のタグ生成
        this.Aside.style.position = 'absolute';
        this.Aside.style.left = lft;
        this.Aside.style.top = top;
        this.Aside.style.width = this.Len;
        this.Aside.style.height = this.Len;
        this.Aside.style.zIndex = 4;                            //手前に設定（ｚインデックス値：４）
        this.Aside.style.visibility = 'hidden';
        this.Aside.onload = makeFnc(this.setLoad, this);        //画像の後処理
        this.Aside.src = cgi + "?GetImg&PS=" + this.iNam + "&serial=" + serCnt++ + "&usr=" + user;
                                                                //画像の読み込み指示
        this.Lcnt = 1;
}
```

【００４３】

（数４続き）

【数５】

```
//２回目以降のセル画像を要求。
bsCell.prototype.loadImg = function() {
        this.Lock = true;
        this.Lcnt = 0;
        this.Bside = document.createElement("img");             //セルBの画像のタグ生成
        this.Bside.style.position = 'absolute';
        this.Bside.style.left = this.Aside.style.left;
        this.Bside.style.top = this.Aside.style.top;
        this.Bside.style.width = this.Len;
        this.Bside.style.height = this.Len;
        this.Aside.style.zIndex = 3;                            //セルAのZ-INDEX=3 奥へ
        this.Bside.style.zIndex = 4;                            //セルBのZ-INDEX=4 手前へ
        this.Bside.style.visibility = 'hidden';
        this.Bside.onload = makeFnc(this.setLoad, this);
        this.Bside.src = cgi + "?GetImg&PS=" + this.iNam + "&serial=" + serCnt++ + "&usr=" + user;
                                                                //画像の読み込み指示
        this.Lcnt = 1;
};

//共通のセル画像読み込み後の処理関数。
bsCell.prototype.setLoad = function(e) {
        if( this.Bside ) {
                this.pNde.appendChild(this.Bside);              //この時点でhtmlのツリー構造に追加され表示が行われる
                this.Bside.style.visibility = 'visible';
                rmImage.unshift(this.Aside);
                this.Aside = this.Bside;                        //セルBの情報をセルAに複写
                this.Bside = null;                              //セルBクリア
        }
        else {
                this.pNde.appendChild(this.Aside);              //この時点でhtmlのツリー構造に追加され表示が行われる
                this.Aside.style.visibility = 'visible';
        }
        this.Lock = false;
        if( this.reqLd ) {                                      //画像読み込み最中だった場合の
                this.reqLd = false;                             //後回しにした画像読み込みを処理
                this.loadImg();
        }
};
```

【００４４】

以上の処理によりＷＥＢブラウザの画面では描画操作に伴い描画操作対象画像に描画加工処理が施されたかのように表示が更新される（エコーバックされる）。そして前記ＸＭＬＨＴＴＰリクエストによる筆ツールの座標および描画半径の送信（Ｓ１５）を筆ツールの座標が変化するごと（ｏｎＭｏｕｓｅＭｏｖｅイベントによる筆ツールの検出座標が変化するごと、すなわち筆ツールの移動速度が遅ければ１画素ごと、速ければ検出される飛び飛びの位置の画素ごと）に順次実行する。すなわちそれ以前のＸＭＬＨＴＴＰリクエストにより送信されたカーソル位置の座標情報に基づき加工処理された画像がＷＥＢブラウザの画面に表示されるの待たずにカーソル位置の座標情報の送信を順次実行する。これにより筆ツールの動きに追従して筆ツールによる描画が順次実行される。またこれに合わせてリアルタイムでエコーバックが行われる。

【００４５】

《実施の形態２》

前記実施の形態１では筆ツールの座標が変化するごとに画像内容に変更が生じるセルを計算しかつ該当するセル画像をＨＴＴＰサーバ１２に要求したが、これでは筆ツールの座標が変化するごとにＨＴＴＰサーバ１２から該当するセル画像を送信することになるので、ＷＥＢブラウザの通信セッションが足らなくなり（受信すべきセル画像の数が多くなり、該セル画像の受信が追いつかなくなる）、エコーバックがリアルタイムでブラウザ画面に表示されない場合が生じる可能性がある。そこで筆ツールの座標の変化が複数回生じるごとに、該複数回の座標の変化に基づく画像加工処理全体で画像内容に変更が生じるセルを計算しかつ該当するセル画像（該複数回の座標の変化による画像加工処理が済んだ画像）をＨＴＴＰサーバ１２に要求する。このようにすれば筆ツールの座標が複数回変化するごとにＨＴＴＰサーバ１２から該当するセル画像を送信すれば済むので、筆ツールの座標が変化するごとにＨＴＴＰサーバ１２から該当するセル画像を送信する場合に比べてＨＴＴＰサーバ１２から送信するセル画像の数を減少させることができ、ＷＥＢブラウザの通信セッション不足を緩和することができる。

【００４６】

《実施の形態３》

前記実施の形態１ではＸＭＬＨＴＴＰリクエストによる筆ツールの座標および描画半径

の送信（図４のＳ１５）を筆ツールの座標が変化するごとに行ったが、これではＷＥＢブラウザ側の通信セッションが足らなくなって（つまり通信セッションがＸＭＬＨＴＴＰリクエストによる筆ツールの座標および描画半径の送信のためだけに費やされ、加工処理したセル画像の受信が追いつかなくなる）、エコーバックがリアルタイムでブラウザ画面に表示されない場合が生じる可能性がある。そこでＸＭＬＨＴＴＰリクエストによる筆ツールの座標および描画半径の送信を筆ツールの座標が変化するごとに行うのに代えて、筆ツールの座標が所定の複数回変化するごとに該変化した複数個の座標および描画半径の組をまとめて１回のＸＭＬＨＴＴＰリクエストで送信する。このようにすればＸＭＬＨＴＴＰリクエストによる筆ツールの座標および描画半径の送信（図４のＳ１５）の回数が減少するので、ＷＥＢブラウザの通信セッション不足を緩和することができる。この場合前記実施の形態２の手法も併用して、該１回のＸＭＬＨＴＴＰリクエストで送信した複数個の座標に基づく画像加工処理全体で画像内容に変更が生じるセルを計算しかつ該当するセル画像（該複数個の座標による画像加工処理が済んだ画像）をＨＴＴＰサーバ１２に要求するようにすれば、ＷＥＢブラウザの通信セッション不足をより緩和することができる。

【００４７】

実施の形態３による処理手順を図５を参照して説明する。ここでは前記実施の形態２の手法も併用する。なお前処理は実施の形態１で説明したのと同じであるのでその説明は省略する。前処理が終了した状態で操作者がＷＥＢブラウザに表示された描画操作対象画像上の描画したい場所にマウスカーソルを位置決めしてマウスボタンを押下しながら移動すると（Ｓ２１）、ＷＥＢブラウザは該画像上での筆ツールが移動した複数の座標および属性設定で指定された筆ツールの描画半径の組を引数に設定してＸＭＬＨＴＴＰリクエストを発行する（Ｓ２２）。この引数の組の数は例えば２～４個程度である。このＸＭＬＨＴＴＰリクエストを発行するＪＡＶＡＳＣＲＩＰＴのプログラムリスト例を数６～数７に示す。

【数６】

```
//変更座標／筆の半径取得と更新処理の呼び出し
function xgetDrawPos() {
        if (xreq.readyState == 4) {                                  //この値が４とは、リクエスト対する返事の
                                                                     //読み込みが終わったことをあらわす
                xtout = 0;
                if (xreq.status == 200) {

                        var        i,j,_obj,_pos,_rad,x,y,r;
                        _obj = xreq.responseText.split('¥n');
                        _pos = _obj[0].split(',');                   //座標値取得
                        _rad = _obj[1] ? _obj[1].split(',') : ''; //筆の直径取得
                        for(i=0; i<_pos.length/2; i++) {             //座標と直径はカンマ形式のリストで送られてくる
                                x = parseInt(_pos[i*2]);
                                y = parseInt(_pos[i*2+1]);
                                r = _rad.length > i ? parseInt(_rad[i]) : pRad;
                                blk.Active(x, y, r);                 //更新処理（関数）の呼び出し部分
                        }

                }
                xreq.abort();
                if( reqUpSend ) {
                        reqUpSend = false;
                        sendPointUp();
                }
        }
}


//変更点の座標と筆の直径から変更の加わった可能性のあるセルを求めて、画像を要求する関数を呼び出す関数
bsBlock.prototype.Active = function(xp, yp, rad) {
        var        i,j,x,y,cnt,ss;
        xs = Math.floor((xp - rad) / this.cWid);                     //floor():最も小さい整数を返す-切り下げ
        ys = Math.floor((yp - rad) / this.cWid);
        xe = Math.ceil((xp + rad) / this.cWid);                      //ceil():最も大きい整数を返す-切り上げ
        ye = Math.ceil((yp + rad) / this.cWid);
        if( !this.Lock && (this.bShow || (xe > 0 && ye > 0 && xs < this.div && ys < this.div)) ) {
                this.bShow = false;
                for(i=0; i<this.div; i++) {
                        for(j=0; j<this.div; j++) {
                                cnt = i * this.div + j;              //セルは１次元で管理しているので１次元に変換
                                if( i >= xs &&  i < xe && j >= ys && j < ye ) {
                                        if( !this.cell[cnt] ) {      //初回はイメージタグを生成して画像を要求する
                                                x = this.Left + this.cWid*i;
                                                y = this.Top + this.cWid*j;
                                                ss = x + "_" + y + "_" + this.cWid;
                                                this.cell[cnt] = new bsCell(this.base, this.cWid*i, this.cWid*j, this.cWid, ss);
                                                                     //この行の new でタグを生成している
                                        }
                                        else if( !this.cell[cnt].Lock )
                                                this.cell[cnt].loadImg();
                                                                     //２回目以降は、イメージタグがすでに出来ているので
                                                                     //単に画像を要求する関数を呼んでいる
                                        else
                                                this.cell[cnt].reqLd = true;
                                                                     //画像更新最中なので読み込み要求を後回しにする
                                        this.bShow = true;
                                }
                                else if( this.cell[cnt] && this.cell[cnt].Lock )
                                        this.bShow = true;
                        }
                }
                if( !this.bShow )
                        this.reqDraw = true;
        }
};

//初回のセル画像要求。
//実装上は、セルＡとセルＢのダブルバッファをしていて、初回はセルＡにロ
//ードしている。
function bsCell(bs, lft, top, len, ist) {                            //セルAに画像読み込み。Z-INDEX=4
        if( !bs ) bs = document.body;
        this.Len = parseInt(len);
        this.xp = parseInt(lft) + Math.floor(this.Len/2);
        this.yp = parseInt(top) + Math.floor(this.Len/2);
        this.iNam = ist;
        this.pNde = bs;
        this.Lock = true;
        this.reqLd = false;
        this.Lcnt = 0;
        this.Bside = null;
        this.Aside = document.createElement("img");                  //セルAの画像のタグ生成
        this.Aside.style.position = 'absolute';
        this.Aside.style.left = lft;
        this.Aside.style.top = top;
        this.Aside.style.width = this.Len;
        this.Aside.style.height = this.Len;
        this.Aside.style.zIndex = 4;                                 //手前に設定（ｚインデックス値：４）
        this.Aside.style.visibility = 'hidden';
        this.Aside.onload = makeFnc(this.setLoad, this);             //画像の後処理
        this.Aside.src = cgi + "?GetImg&PS=" + this.iNam + "&serial=" + serCnt++ + "&usr=" + user;
                                                                     //画像の読み込み指示
        this.Lcnt = 1;
}
```

数４との違いはここだけ

【００４８】

（数６続き）

【数７】

```
//２回目以降のセル画像を要求。
bsCell.prototype.loadImg = function() {
        this.Lock = true;
        this.Lcnt = 0;
        this.Bside = document.createElement("img");             //セルBの画像のタグ生成
        this.Bside.style.position = 'absolute';
        this.Bside.style.left = this.Aside.style.left;
        this.Bside.style.top = this.Aside.style.top;
        this.Bside.style.width = this.Len;
        this.Bside.style.height = this.Len;
        this.Aside.style.zIndex = 3;                            //セルAのZ-INDEX=3 奥へ
        this.Bside.style.zIndex = 4;                            //セルBのZ-INDEX=4 手前へ
        this.Bside.style.visibility = 'hidden';
        this.Bside.onload = makeFnc(this.setLoad, this);
        this.Bside.src = cgi + "?GetImg&PS=" + this.iNam + "&serial=" + serCnt++ + "&usr=" + user;
                                                                //画像の読み込み指示
        this.Lcnt = 1;
};

//共通のセル画像読み込み後の処理関数。
bsCell.prototype.setLoad = function(e) {
        if( this.Bside ) {
                this.pNde.appendChild(this.Bside);              //この時点でhtmlのツリー構造に追加され表示が行われる
                this.Bside.style.visibility = 'visible';
                rmImage.unshift(this.Aside);
                this.Aside = this.Bside;                        //セルBの情報をセルAに複写
                this.Bside = null;                              //セルBクリア
        }
        else {
                this.pNde.appendChild(this.Aside);              //この時点でhtmlのツリー構造に追加され表示が行われる
                this.Aside.style.visibility = 'visible';
        }
        this.Lock = false;
        if( this.reqLd ) {                                      //画像読み込み最中だった場合の
                this.reqLd = false;                             //後回しにした画像読み込みを処理
                this.loadImg();
        }
};
```

【００４９】

ＨＴＴＰサーバ１２はＸＭＬＨＴＴＰリクエストを受け取ると、先に保存されている筆ツールの属性（色、全体的な濃度、断面形状、濃度分布パターン、補間の有無）と、今回送られてきた複数の筆ツールの座標および描画半径の組に基づき、画像処理プログラムを使用して、自らが保持管理する前記描画操作対象画像に対して筆ツールによる画像加工処理を実行する。そして実際に加工した（またはこれから加工しようとする）複数の該画像加工処理の中心座標（筆ツールの座標）と描画半径をＸＭＬＨＴＴＰリクエストに対する応答として返信する（Ｓ２３）。なお属性設定で補間の有無について「有り」が指示された場合であって今回送られてきた筆ツールの複数の座標間が離れている場合は該座標間を補間して画像加工処理を施し、かつ該補間により数が増えた中心座標と描画半径を返信する。また前回のＸＭＬＨＴＴＰリクエストで送られてきた筆ツールの最後の座標と今回のＸＭＬＨＴＴＰリクエストで送られてきた筆ツールの最初の座標との間が離れている場合も該座標間を補間して画像加工処理を施し、かつ該補間された分の中心座標と描画半径を併せて返信する。

【００５０】

ＷＥＢブラウザは複数の画像加工処理の中心座標と描画半径を受け取ると、これらの情報に基づき、これら複数の中心座標全体で、描画操作対象画像を分割するいずれのセルの画像内容に画像加工処理による変更が生じるかを計算する。そして計算により求められたセルの識別番号（例えば図３のＸ，Ｙ座標を組み合わせた数値）を引数として該当するセル画像を要求するＨＴＴＰリクエストを発行する（Ｓ２４）。この際ＨＴＴＰリクエストの仕様上、返信として受け取る予定のセル画像の表示位置（ブラウザ画面の原点を基準とした位置）を指定する。このＨＴＴＰリクエストはそのリクエスト数が増える可能性がある以外は前記実施の形態１のステップＳ１７（図４）における処理と同じである。ＨＴＴＰサーバ１２は今回の複数の筆ツールの座標および描画半径により加工処理された最終的な画像（例えば筆ツールの座標の４個を１組にする場合は、この４個の座標全部による加工処理が済んだ画像）から、要求されたセル画像を切り出してＷＥＢブラウザに送信する（Ｓ２５）。このＨＴＴＰリクエストに対するＨＴＴＰサーバ１２によるセル画像の送信処理およびＷＥＢブラウザによる該セル画像の受信および表示処理（Ｓ２６）は、実施の形態１のステップＳ１８，Ｓ１９とそれぞれ同じである。

【００５１】
前記ＸＭＬＨＴＴＰリクエストによる複数の筆ツールの座標および描画半径の組の送信（Ｓ２１）を筆ツールの座標が所定の複数回変化するごとに順次実行する（前回送信してから所定時間内に該所定の複数回に達しない場合は該所定時間経過後に送信する）ことにより、筆ツールの動きに追従して筆ツールによる描画が順次実行されていく。この実施の形態３によれば、筆ツールの座標が変化するごとにＸＭＬＨＴＴＰリクエストを発行する場合に比べてＸＭＬＨＴＴＰリクエストの回数が減少する。また複数の筆ツールの座標全部による加工処理が済んだ画像からセル画像を切り出してＷＥＢブラウザに送信するので、筆ツールの座標が変化するごとにセル画像を切り出してＷＥＢブラウザに送信する場合に比べてＨＴＴＰサーバ１２から送信するセル画像数も減少する。これによりＷＥＢブラウザ側の通信セッション不足を防止することができる。

【００５２】
《実施の形態４》
前記実施の形態では端末装置１０からの描画操作により該端末装置１０にエコーバックする場合について説明したが、描画操作対象画像に対し端末装置１０と他の端末装置で同時に描画操作を行って（または他の端末装置単独で描画操作を行って）、他の端末装置による描画操作により加工処理された画像を端末装置１０にエコーバックさせることもできる。そのようにする場合の端末装置１０とＨＴＴＰサーバ１２間の処理手順を図６を参照して説明する。なお前処理は実施の形態１で説明したのと同じであるのでその説明は省略する。

【００５３】
ＷＥＢブラウザは、前処理が終了した状態で、実施の形態１と同様にして端末装置１０における描画操作に基づきＸＭＬＨＴＴＰリクエストによる筆ツールの座標および描画半径を送信する。そして該描画操作に基づくＸＭＬＨＴＴＰリクエストの合間に、所定の設定時間が経過するごとにタイマーイベント（ｓｅｔＩｎｔｅｒｖａｌあるいはｓｅｔＴｉｍｅｏｕｔ。ｓｅｔＴｉｍｅｏｕｔを使用する場合は設定時間後に再びｓｅｔＴｉｍｅｏｕｔを設定する）によるトリガで（Ｓ３１）他の端末装置のＷＥＢブラウザからの描画操作に基づいてされた画像加工処理の中心座標と描画半径の組を要求するＸＭＬＨＴＴＰリクエストを発行する（Ｓ３２）。ＨＴＴＰサーバ１２はこのリクエストを受信して、該他の端末装置のＷＥＢブラウザからの操作で実際に画像加工した（またはこれから加工しようとする）中心座標と描画半径のうち、前回のタイマートリガによるＸＭＬＨＴＴＰリクエストの返信の際に返信していないものを返信する（Ｓ３３）。この返信を受け取ったときのＷＥＢブラウザによる画像内容に変更が生じるセルの計算および該当するセル画像を要求するＨＴＴＰリクエストの送信処理（Ｓ３４）、このＨＴＴＰリクエストに対するＨＴＴＰサーバ１２によるセル画像の送信処理（Ｓ３５）、ＷＥＢブラウザによる該セル画像の受信および表示処理（Ｓ３６）は、実施の形態１のステップＳ１７，Ｓ１８，Ｓ１９と同じである。以上の処理により、描画操作対象画像に対し端末装置１０と他の端末装置で同時に描画操作を行って（または他の端末装置単独で描画操作を行って）、該描画操作により加工処理された画像を端末装置１０にエコーバックさせることができる。なお、他の端末装置においてもタイマーイベントによるトリガに基づき端末装置１０と同様に処理することにより、端末装置１０における描画操作により加工された画像を他の端末装置にエコーバックさせることができ、これにより各端末装置で共通の画像を表示しながら同時に描画操作をすることができる。タイマートリガによりＸＭＬＨＴＴＰリクエストを発行するＪＡＶＡＳＣＲＩＰＴのプログラムリスト例を数８に示す。

【数８】

```
//タイマーによる画面更新のための変更座標／筆の半径取得要求
function checkInterval() {
        if( xreq && (!xreq.readyState || !msie && !xtout) ) {
                xreq.onreadystatechange = xgetDrawPos;          //その結果は関数「xgetDrawPos」
                                                                //で処理することを設定
                xreq.open("POST", cgi + "?getChg", true);
                xreq.setRequestHeader("Content-Type", "text/plain");
                xreq.send("usr=" + user);                       //xmlhttpリクエストの発行
        }
}
```

【００５４】

《その他の実施の形態》

前記実施の形態の一部を以下のように様々に変更することができる。

［Ａ］前記実施の形態では筆ツールの描画サイズ（半径または直径）の情報を属性設定時にＨＴＴＰサーバ１２に送信せずに実際に描画操作が行われたときに筆ツールの座標と組にして送信するようにしたが、これに代えて描画サイズの情報を属性設定時に他の属性（色、全体的な濃度、断面形状、濃度分布パターン、補間の有無）とともにＨＴＴＰサーバ１２に送信し、ＨＴＴＰサーバ１２の共有メモリ２６の筆ツール属性情報格納領域２８に格納して画像加工処理に利用することができる。

【００５５】

［Ｂ］前記実施の形態ではＷＥＢブラウザはＨＴＴＰサーバ１２から返信される画像加工処理の中心座標と描画半径に基づいて画像内容に変更が生じるセルを計算したが、これに代えてＷＥＢブラウザが自ら保持する描画操作時の筆ツールの座標と描画サイズの情報に基づいて画像内容に変更が生じるセルを計算することもできる。あるいはＷＥＢブラウザはＨＴＴＰサーバ１２から返信される画像加工処理の中心座標とＷＥＢブラウザが自ら保持する描画サイズの情報に基づいて画像内容に変更が生じるセルを計算することもできる。ただし前者の場合、ＨＴＴＰサーバ１２側で座標を補間処理して画像処理を行う場合は、ＷＥＢブラウザが画像内容に変更が生じるセルを正しく計算するためには、該ＷＥＢブラウザ側でも同じ補間計算をしたうえで画像内容に変更が生じるセルの計算をする必要がある。

【００５６】

［Ｃ］前記実施の形態ではＷＥＢブラウザ側で画像内容に変更が生じるセルを計算したが、これに代えてＨＴＴＰサーバ１２側で画像加工処理の中心座標と描画半径に基づいて画像内容に変更が生じるセルを計算し、該計算により求められたセルの識別番号（例えば図３のＸ，Ｙ座標を組み合わせた数値）をＷＥＢブラウザに送信することもできる。

【００５７】

［Ｄ］あるいはＨＴＴＰサーバ１２側で画像加工処理の中心座標と描画半径に基づいて画像内容に変更が生じる領域全体を含む１つの矩形領域を特定する座標情報をＷＥＢブラウザに送信し、ＷＥＢブラウザ側で該矩形領域が掛かる全てのセルを計算することもできる。図７はその説明図である。図７は前記図２（ａ）と同じ描画状態を示す。すなわち図７（ａ）のようにマウスカーソルをカンバス３６上の任意の位置Ｐ１（x_1，y_1）に位置決めしてマウスボタンを押下すると、マウスカーソル位置Ｐ１（x_1，y_1）を中心とする半径ｒの円形に筆による画像３８が描かれる。このとき画像３８全体を含む最小の矩形領域は同図に点線４２で囲んだ領域となる。この矩形領域４２はその対角位置の２つの座標Ｐ１１（x_{11}，y_{11}）、Ｐ１２（x_{12}，y_{12}）で特定される。そこでＨＴＴＰサーバ１２はこの２つの座標Ｐ１１（x_{11}，y_{11}）、Ｐ１２（x_{12}，y_{12}）をＷＥＢブラウザに送信する。ＷＥＢブラウザはこの２つの座標Ｐ１１，Ｐ１２を受信して対応する矩形領域４２が掛かる全てのセルを計算して求める。すなわち図７（ｂ）に示すように２つの座標Ｐ１１，Ｐ１２で特定される矩形領域４２が掛かる全てのセルとして、Ｘ座標が４，５，６，７でＹ座標が３，４，５，６である同図に太線で囲んだ領域の１６個のセルを求める。そしてＷＥＢブラウザはＨＴＴＰサーバ１２に対してＨＴＴＰリクエス

トによりこの１６個のセルの画像を要求する。ＨＴＴＰサーバ１２はこのリクエストを受けて、保持している画像加工処理が行われた画像からこの１６個のセルの画像を切り出してＷＥＢブラウザに送信する。ＷＥＢブラウザは各セル画像を受信しブラウザ画面に表示されている画像の該当するセル領域に、それまで該矩形領域に表示されていた画像に代えて表示する。この１６個のセルの中には画像内容に変更が生じたセルに隣接する画像内容に変更が生じていないセル｛７，３｝、｛７，６｝も含まれているが、この発明はＷＥＢブラウザが画像内容に変更が生じていないセルの画像を一部含んで要求し画像表示を更新する場合を排除するものではない。この場合も描画操作対象画像全体を送信する場合に比べて画像表示を高速に更新することができる。

【００５８】

［Ｅ］前記実施の形態では操作者が筆の描画サイズを設定可能としたが、これに代えてＨＴＴＰサーバ側で筆の描画サイズを固定値として初めから保持することもできる。この場合、ＷＥＢブラウザ側で画像内容に変更が生じるセルを計算する場合はＨＴＴＰサーバから筆の描画サイズの固定値をＷＥＢブラウザに送信する。またＨＴＴＰサーバ側で画像内容に変更が生じるセルを計算しＷＥＢブラウザに送信する場合はＨＴＴＰサーバから筆の描画サイズの固定値をＷＥＢブラウザに送信する必要はない。

【００５９】

［Ｆ］前記実施の形態では新たなセル画像のｚインデックスを現在その領域に表示されている画像のｚインデックスよりも大きい値に設定して該新たなセル画像を手前に配置して表示することにより表示画像の更新を行うようにしたが、これに代えて現在表示されているセル画像を新しいセル画像で完全に差し替える（今まで表示されていたセル画像は差し替えと同時にＨＴＭＬドキュメントとのリンクが解除される）ことにより表示画像の更新を行うこともできる。

【００６０】

［Ｇ］前記実施の形態では描画操作対象画像を仮想的に複数のセルに分割して（画像ファイルは１つ）管理したが、これに代えて描画操作対象画像を実際にスライス画像に分割して（画像ファイルはスライス画像ごと。個々のスライス画像の大きさは例えば縦・横各方向とも４８０画素）管理することもできる。また描画操作対象画像を実際にスライス画像に分割したうえで（画像ファイルはスライス画像ごと。個々のスライス画像の大きさは例えば縦・横各方向とも４８０画素）、さらに個々のスライス画像を仮想的に複数のセルに分割して（個々のセル画像の大きさは例えば縦・横各方向とも４８画素）管理することもできる。

【００６１】

［Ｈ］前記実施の形態ではマウスボタンを押したままカーソルを移動させるときの移動速度に応じて描画濃度を変化させる（つまり「描画操作態様」をカーソルの移動速度とする）ようにしたが、カーソルを１箇所に留まらせたままマウスボタンを押し続ける時間等に応じて描画濃度を変化させる（つまり「描画操作態様」をマウスボタンを押し続ける時間その他とする）こともできる。この場合はｏｎＭｏｕｓｅＭｏｖｅイベントに代えてタイマーイベント（ｓｅｔＩｎｔｅｒｖａｌあるいはｓｅｔＴｉｍｅｏｕｔ。ｓｅｔＴｉｍｅｏｕｔを使用する場合は設定時間後に再びｓｅｔＴｉｍｅｏｕｔを設定する）を使用する。なおタイマーイベントではカーソルの座標位置情報は取得できないので、処理に必要な筆ツールの座標としては直前のｏｎＭｏｕｓｅＭｏｖｅイベントで取得した筆ツールの座標を用い、処理に必要な画像加工処理の中心座標としては直前のＸＭＬＨＴＴＰリクエストに対する応答として取得した画像加工処理の中心座標を用いる。

【００６２】

［Ｉ］前記実施の形態ではポインティングデバイスとしてマウスを使用した場合について説明したが、タブレットその他のポインティングデバイスを使用することもできる。タブレットを使用する場合はスタイラスペンをタブレットに押しつける操作がマウスボタンを押下する操作に対応する。

【００６３】

［Ｊ］前記実施の形態では筆ツールの描画形状を円形としたが、楕円形、角形その他の任意の形状にすることもできる。

【図面の簡単な説明】

【００６４】

【図１】この発明の実施の形態のシステム構成を示すブロック図である。

【図２】図１のシステム構成において描画操作対象画像のカンバスに対して筆ツールにより描画を行う状態を示す図である。

【図３】カンバスを仮想的に矩形（セル）に分割した一例および図２（ａ）の描画状態に対応して画像表示を更新するセル画像の範囲を示す図である。

【図４】この発明の実施の形態１による処理手順を示すフロー図である。

【図５】この発明の実施の形態３による処理手順を示すフロー図である。

【図６】この発明の実施の形態４による処理手順を示すフロー図である。

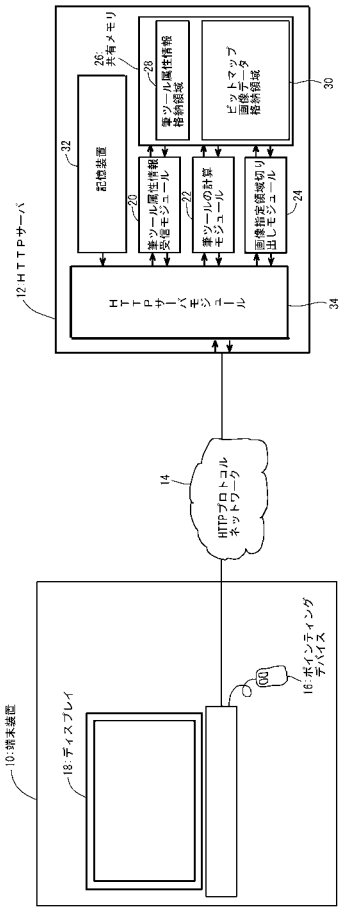
【図７】図２（ａ）と同じ描画状態について画像表示を更新するセル画像の範囲が図３と異なる例を示す図である。

【符号の説明】

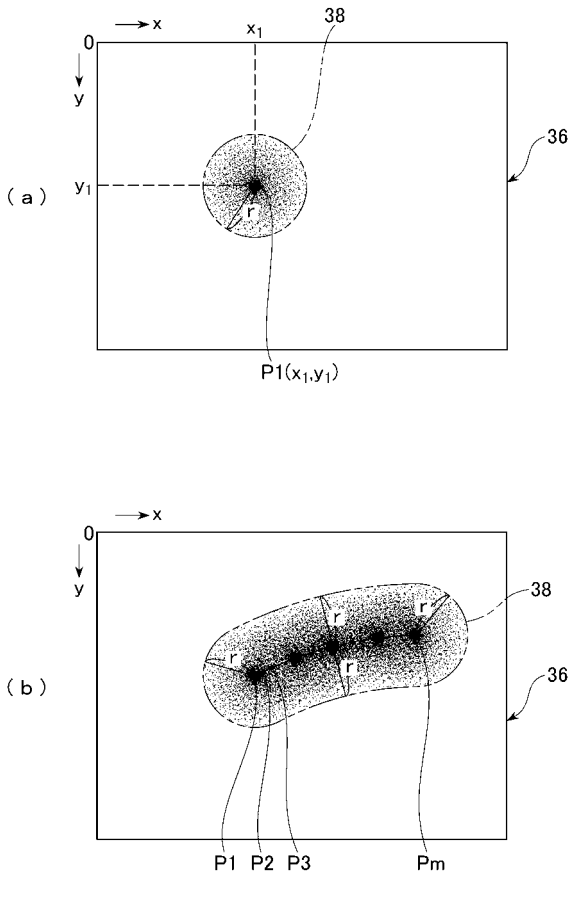
【００６５】

１０…端末装置、１６…ポインティングデバイス、１２…ＨＴＴＰサーバ、１４…ＨＴＴＰプロトコルネットワーク、３６…描画操作対象画像の全体を構成するカンバス、３８…筆ツールによる画像、４０…セル、Ｐ１，Ｐ２，・・・，Ｐｍ…マウスカーソルの位置

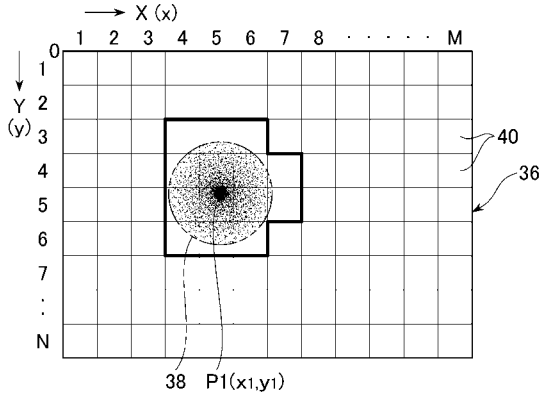
【図１】



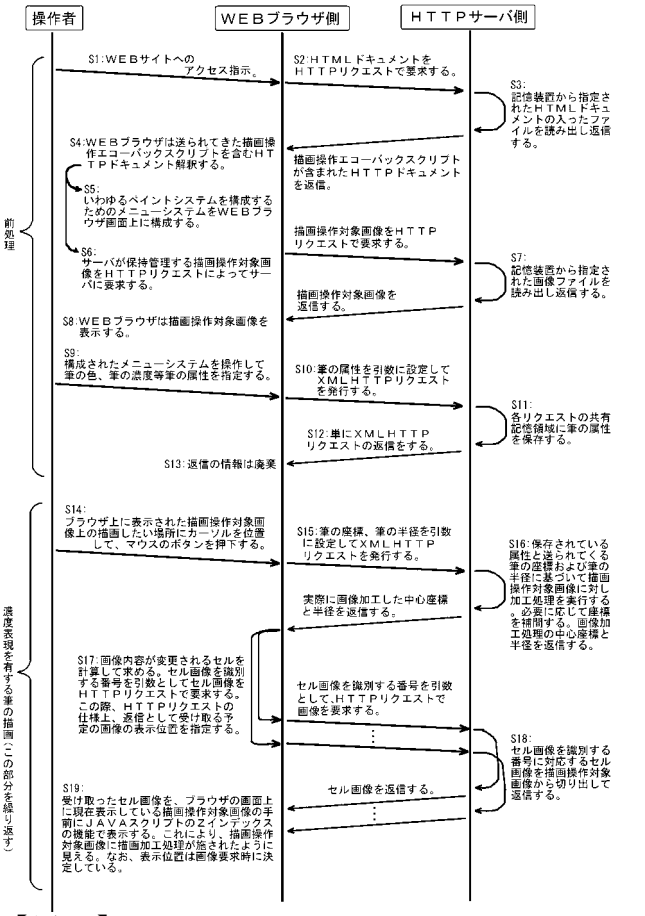
【図２】



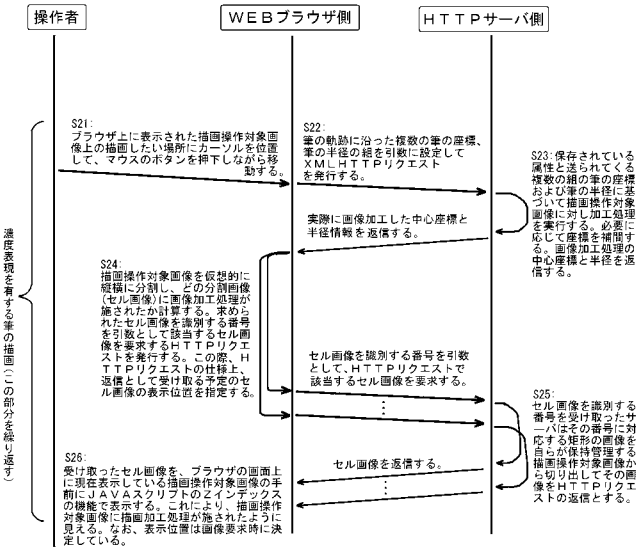
【図３】


X (x)
1 2 3 4 5 6 7 8 ・・・・・ M
Y (y)
0 1 2 3 4 5 6 7 ・・ N
40
36
38
P1(x1,y1)

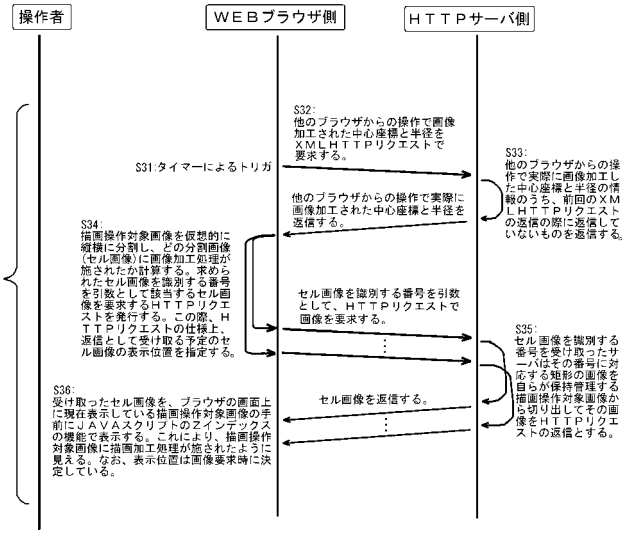
【図４】


操作者
WEBブラウザ側
HTTPサーバ側
前処理
S1:WEBサイトへのアクセス指示。
S2:HTMLドキュメントをHTTPリクエストで要求する。
S3:記憶装置から指定されたHTMLドキュメントの入ったファイルを読み出し返信する。
S4:WEBブラウザは送られてきた描画操作エコーバックスクリプトを含むHTTPドキュメント解釈する。
描画操作エコーバックスクリプトが含まれたHTTPドキュメントを返信。
S5:いわゆるペイントシステムを構成するためのメニューシステムをWEBブラウザ画面上に構成する。
S6:サーバが保持管理する描画操作対象画像をHTTPリクエストによってサーバに要求する。
描画操作対象画像をHTTPリクエストで要求する。
S7:記憶装置から指定された画像ファイルを読み出し返信する。
描画操作対象画像を返信する。
S8:WEBブラウザは描画操作対象画像を表示する。
S9:構成されたメニューシステムを操作して筆の色、筆の濃度等筆の属性を指定する。
S10:筆の属性を引数に設定してXMLHTTPリクエストを発行する。
S11:各リクエストの共有記憶領域に筆の属性を保存する。
S12:単にXMLHTTPリクエストの返信をする。
S13:返信の情報は廃棄
濃度表現を有する筆の描画（この部分を繰り返す）
S14:ブラウザ上に表示された描画操作対象画像上の描画したい場所にカーソルを位置して、マウスのボタンを押下する。
S15:筆の座標、筆の半径を引数に設定してXMLHTTPリクエストを発行する。
S16:保存されている属性と送られてくる筆の座標および筆の半径に基づいて描画操作対象画像に対し加工処理を実行する。必要に応じて座標を補間する。画像加工処理の中心座標と半径を返信する。
実際に画像加工した中心座標と半径を返信する。
S17:画像内容が変更されるセルを計算して求める。セル画像を識別する番号を引数としてセル画像をHTTPリクエストで要求する。この際、HTTPリクエストの仕様上、返信として受け取る予定の画像の表示位置を指定する。
セル画像を識別する番号を引数として、HTTPリクエストで画像を要求する。
S18:セル画像を識別する番号に対応するセル画像を描画操作対象画像から切り出して返信する。
セル画像を返信する。
S19:受け取ったセル画像を、ブラウザの画面上に現在表示している描画操作対象画像の手前にJAVAスクリプトのZインデックスの機能で表示する。これにより、描画操作対象画像に描画加工処理が施されたように見える。なお、表示位置は画像要求時に決定している。

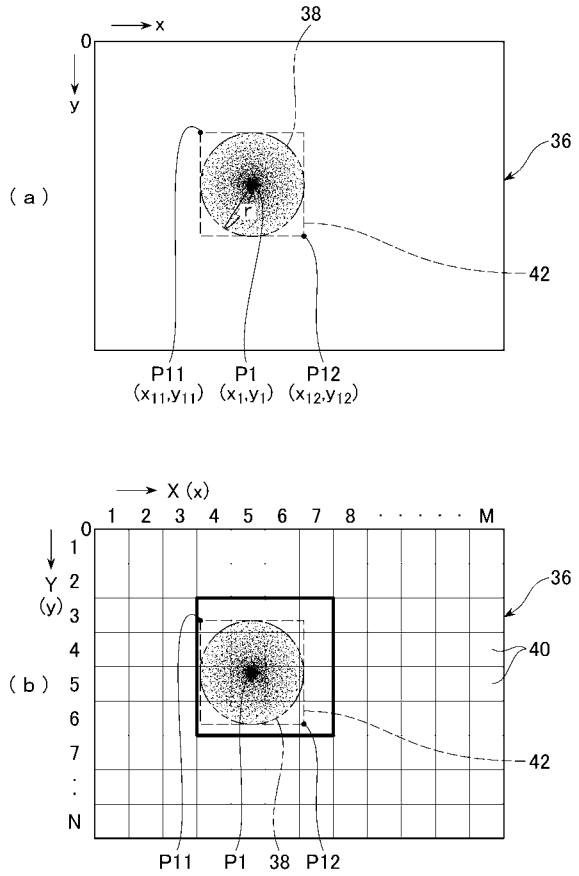
【図５】


操作者
WEBブラウザ側
HTTPサーバ側
濃度表現を有する筆の描画（この部分を繰り返す）
S21:ブラウザ上に表示された描画操作対象画像上の描画したい場所にカーソルを位置して、マウスのボタンを押下しながら移動する。
S22:筆の軌跡に沿った複数の筆の座標、筆の半径の組を引数に設定してXMLHTTPリクエストを発行する。
S23:保存されている属性と送られてくる複数の組の筆の座標および筆の半径に基づいて描画操作対象画像に対し加工処理を実行する。必要に応じて座標を補間する。画像加工処理の中心座標と半径を返信する。
実際に画像加工した中心座標と半径情報を返信する。
S24:描画操作対象画像を仮想的に縦横に分割し、どの分割画像（セル画像）に画像加工処理が施されたか計算する。求められたセル画像を識別する番号を引数として該当するセル画像を要求するHTTPリクエストを発行する。この際、HTTPリクエストの仕様上、返信として受け取る予定のセル画像の表示位置を指定する。
セル画像を識別する番号を引数として、HTTPリクエストで該当するセル画像を要求する。
S25:セル画像を識別する番号を受け取ったサーバはその番号に対応する矩形の画像を自らが保持管理する描画操作対象画像から切り出してその画像をHTTPリクエストの返信とする。
S26:受け取ったセル画像を、ブラウザの画面上に現在表示している描画操作対象画像の手前にJAVAスクリプトのZインデックスの機能で表示する。これにより、描画操作対象画像に描画加工処理が施されたように見える。なお、表示位置は画像要求時に決定している。
セル画像を返信する。

【図６】


操作者
WEBブラウザ側
HTTPサーバ側
濃度表現を有する筆の描画（この部分を繰り返す）
S31:タイマーによるトリガ
S32:他のブラウザからの操作で画像加工された中心座標と半径をXMLHTTPリクエストで要求する。
S33:他のブラウザからの操作で実際に画像加工した中心座標と半径の情報のうち、前回のXMLHTTPリクエストの返信の際に返信していないものを返信する。
他のブラウザからの操作で実際に画像加工された中心座標と半径を返信する。
S34:描画操作対象画像を仮想的に縦横に分割し、どの分割画像（セル画像）に画像加工処理が施されたか計算する。求められたセル画像を識別する番号を引数として該当するセル画像を要求するHTTPリクエストを発行する。この際、HTTPリクエストの仕様上、返信として受け取る予定のセル画像の表示位置を指定する。
セル画像を識別する番号を引数として、HTTPリクエストで画像を要求する。
S35:セル画像を識別する番号を受け取ったサーバはその番号に対応する矩形の画像を自らが保持管理する描画操作対象画像から切り出してその画像をHTTPリクエストの返信とする。
S36:受け取ったセル画像を、ブラウザの画面上に現在表示している描画操作対象画像の手前にJAVAスクリプトのZインデックスの機能で表示する。これにより、描画操作対象画像に描画加工処理が施されたように見える。なお、表示位置は画像要求時に決定している。
セル画像を返信する。

【図７】



【手続補正書】
【提出日】平成19年11月5日(2007.11.5)
【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】
端末装置にインストールされたＷＥＢブラウザの画面に画像を表示し、該画面上で該端末装置の操作者がポインティングデバイスで描画操作をすることにより、該ポインティングデバイスによるカーソル位置の画像の表示に濃度表現を有する筆ツールによる変更を加える画像表示更新方法であって、
ここに「濃度表現を有する筆ツール」とは、ポインティングデバイスの描画操作態様に応じて描画濃度が変化する描画を実現する描画ツールであって、
サーバに、前記操作者による描画操作対象となる画像のデータと、該描画操作に応じて該画像データに対して画像加工処理を実行する画像処理プログラムを保持管理し、
前記端末装置と前記サーバをネットワークを介して相互に通信可能な状態に設定し、
前記端末装置のＷＥＢブラウザが前記サーバに対しＨＴＭＬドキュメントをＨＴＴＰリクエストによって要求すると、該サーバは該ＷＥＢブラウザによって解釈可能な描画操作エコーバックスクリプトが内部に記述されたＨＴＭＬドキュメントまたは該描画操作エコーバックスクリプトを外部ファイルとして読み込むことが記述されたＨＴＭＬドキュメントを該端末装置に該ＨＴＴＰリクエストに対する応答として返信し、該端末装置のＷＥＢブラウザは該ＨＴＭＬドキュメントを受信して、該ＷＥＢブラウザに前記描画操作エコーバックスクリプトを付加し、

前記ＨＴＭＬドキュメントは、前処理として、前記サーバが保持管理する前記描画操作対象画像をＨＴＴＰリクエストによって該サーバに要求し、その返信として受け取った画像データによる画像を前記ＷＥＢブラウザの画面に表示し、

前記端末装置のＷＥＢブラウザに付加された前記描画操作エコーバックスクリプトは、前記画像が表示された画面上で前記描画操作がされたときに、該描画操作に応じたカーソル位置の座標情報をＸＭＬＨＴＴＰリクエストによって前記サーバに送信し、

前記サーバの画像処理プログラムは、受信した該カーソル位置の座標情報と、別途取得しまたは初めから保持する筆の描画サイズの情報に基づき、自らが保持管理する前記描画操作対象画像に対して、前記濃度表現を有する筆ツールによる画像加工処理を実行し、

前記描画操作エコーバックスクリプトは前記カーソル位置の座標情報を送信後に、前記ＷＥＢブラウザの画面に表示された画像を複数の領域に分割する矩形領域のうち前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を取得し、かつ該単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信し、

前記サーバは該単一または複数のＨＴＴＰリクエストを受信して、前記ＷＥＢブラウザの画面に表示された画像の該当する単一または複数の矩形領域の前記画像加工処理が終了した画像データを選択的に返信し、

前記ＷＥＢブラウザは受信した単一または複数の矩形領域の画像データによる画像を、該ＷＥＢブラウザの画面の該当する矩形領域に、それまで該矩形領域に表示されていた画像に代えて表示する画像表示更新方法。

【請求項２】

端末装置にインストールされたＷＥＢブラウザの画面に画像を表示し、該画面上で該端末装置の操作者がポインティングデバイスで描画操作をすることにより、該ポインティングデバイスによるカーソル位置の画像の表示に濃度表現を有する筆ツールによる変更を加える画像表示更新方法であって、

ここに「濃度表現を有する筆ツール」とは、ポインティングデバイスの描画操作態様に応じて描画濃度が変化する描画を実現する描画ツールであって、

サーバに、前記操作者による描画操作対象となる画像のデータと、該描画操作に応じて該画像データに対して画像加工処理を実行する画像処理プログラムを保持管理し、

前記端末装置と前記サーバをネットワークを介して相互に通信可能な状態に設定し、

前記端末装置のＷＥＢブラウザが前記サーバに対しＨＴＭＬドキュメントをＨＴＴＰリクエストによって要求すると、該サーバは該ＷＥＢブラウザによって解釈可能な描画操作エコーバックスクリプトが内部に記述されたＨＴＭＬドキュメントまたは該描画操作エコーバックスクリプトを外部ファイルとして読み込むことが記述されたＨＴＭＬドキュメントを該端末装置に該ＨＴＴＰリクエストに対する応答として返信し、該端末装置のＷＥＢブラウザは該ＨＴＭＬドキュメントを受信して、該ＷＥＢブラウザに前記描画操作エコーバックスクリプトを付加し、

前記ＨＴＭＬドキュメントは、前処理として、前記サーバが保持管理する前記描画操作対象画像をＨＴＴＰリクエストによって該サーバに要求し、その返信として受け取った画像データによる画像を前記ＷＥＢブラウザの画面に表示し、

前記端末装置のＷＥＢブラウザに付加された前記描画操作エコーバックスクリプトは、前記画像が表示された画面上で前記描画操作がされたときに、該描画操作に応じたカーソル位置の座標情報をＸＭＬＨＴＴＰリクエストによって前記サーバに送信し、

前記サーバの画像処理プログラムは、受信した該カーソル位置の座標情報と、別途取得しまたは初めから保持する筆の描画サイズの情報に基づき、自らが保持管理する前記描画操作対象画像に対して、前記濃度表現を有する筆ツールによる画像加工処理を実行し、

前記描画操作エコーバックスクリプトは前記カーソル位置の座標情報を送信後に、前記ＷＥＢブラウザの画面に表示された画像を複数の領域に分割する矩形領域のうち前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を取得し、かつ該単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰ

リクエストを該サーバに送信し、
前記サーバは該単一または複数のＨＴＴＰリクエストを受信して、前記ＷＥＢブラウザの画面に表示された画像の該当する単一または複数の矩形領域の前記画像加工処理が終了した画像データを選択的に返信し、
前記ＷＥＢブラウザは受信した単一または複数の矩形領域の画像データによる画像を、該ＷＥＢブラウザの画面の該当する矩形領域に、それまで該矩形領域に表示されていた画像に代えて表示し、
該ＷＥＢブラウザは前記ＸＭＬＨＴＴＰリクエストによるカーソル位置の座標情報の送信を、それ以前のＸＭＬＨＴＴＰリクエストにより送信されたカーソル位置の座標情報に基づき加工処理された画像が該ＷＥＢブラウザの画面に表示されるのを待たずに、前回のＸＭＬＨＴＴＰリクエストに対する応答を受信後に順次実行し、
前記サーバの画像処理プログラムは該ＸＭＬＨＴＴＰリクエストに基づき、前記画像加工処理がされた画像に対しさらに該画像加工処理を繰り返す画像表示更新方法。

【請求項３】
前記描画操作エコーバックスクリプトは前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を自ら計算して取得する請求項１または２記載の画像表示更新方法。

【請求項４】
前記描画操作エコーバックスクリプトは前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を前記サーバの画像処理プログラムが計算した結果を受信して取得する請求項１または２記載の画像表示更新方法。

【請求項５】
前記描画操作エコーバックスクリプトは前記筆の描画サイズの情報を前記カーソル位置の座標情報と同時に前記サーバに送信する請求項１から４のいずれか１つに記載の画像表示更新方法。

【請求項６】
前記描画操作エコーバックスクリプトは前記筆の描画サイズの情報を、操作者により筆の描画サイズの設定操作がされたときに前記サーバに送信する請求項１から４のいずれか１つに記載の画像表示更新方法。

【請求項７】
前記サーバの画像処理プログラムは前記端末装置から受信した前記カーソル位置の座標情報と、該端末装置から該カーソル位置の座標情報と同時に受信しまたは該サーバが該カーソル位置の座標情報を受信する以前から保持している筆の描画サイズの情報に基づき前記画像加工処理を実行し、かつ該画像描画処理により描画するまたは描画した筆の描画中心位置の座標情報をＸＭＬＨＴＴＰリクエストに対する応答として前記端末装置のＷＥＢブラウザに返信し、前記描画操作エコーバックスクリプトは該サーバから受信する筆の描画中心位置の座標情報と、該サーバから該筆の描画中心位置の座標情報と同時に受信しまたは該描画操作エコーバックスクリプトが該筆の描画中心位置の座標情報を受信する以前から保持している筆の描画サイズの情報を利用して、前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を自ら計算して取得する請求項１または２記載の画像表示更新方法。

【請求項８】
前記筆の描画中心位置の座標情報は前記カーソル位置の座標情報そのものまたは該カーソル位置の座標間を該カーソル位置の座標情報で補間処理した座標情報である請求項７記載の画像表示更新方法。

【請求項９】
前記描画操作エコーバックスクリプトは自ら取得しかつ前記サーバを経由しない前記カーソル位置の座標情報と、該描画操作エコーバックスクリプトが該カーソル位置の座標情報を取得する以前から保持している筆の描画サイズの情報を利用して、前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を自ら計算して取

得する請求項１または２記載の画像表示更新方法。
【請求項１０】
前記画像処理プログラムは前記描画操作対象画像に対して他の端末装置から同時に描画操作がされたときに該描画操作に基づく画像加工処理を併せて実行するものであり、
前記描画操作エコーバックスクリプトはタイマー駆動により定期的にＸＭＬＨＴＴＰリクエストを発行し、
前記画像処理プログラムは前記他の端末装置からの描画操作に基づく画像加工処理に対応して、該画像処理プログラムで計算した画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報または前記描画操作エコーバックスクリプトで該矩形領域を計算するのに必要な情報を該ＸＭＬＨＴＴＰリクエストに対する応答として返信し、
前記描画操作エコーバックスクリプトは該サーバから受信して取得した情報による単一または複数の矩形領域または該サーバから受信した情報に基づき自ら計算して取得した情報による単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信し、
前記サーバは該ＨＴＴＰリクエストを受信して、前記ＷＥＢブラウザの画面に表示された画像の該当する単一または複数の矩形領域の画像データを選択的に返信し、
前記ＷＥＢブラウザは受信した単一または複数の矩形領域の画像データによる画像を、該ＷＥＢブラウザの画面の該当する矩形領域に、それまで該矩形領域に表示されていた画像に代えて表示する請求項１から９のいずれか１つに記載の画像表示更新方法。
【請求項１１】
前記サーバが保持管理する描画操作対象画像はその画像全体に相当する単一の画像ファイルまたは該画像全体を実際に矩形に分割した複数のスライス画像に相当する複数の画像ファイルで構成され、前記矩形領域は１つの画像ファイルによる画像を仮想的に複数の矩形に分割した該仮想分割画像単位の領域である請求項１から１０のいずれか１つに記載の画像表示更新方法。
【請求項１２】
前記サーバが保持管理する描画操作対象画像はその画像全体を実際に複数の矩形に分割した複数のスライス画像の画像ファイルで構成され、前記矩形領域は該スライス画像単位の領域である請求項１から１０のいずれか１つに記載の画像表示更新方法。
【請求項１３】
前記端末装置の描画操作エコーバックスクリプトは、受信した単一または複数の矩形領域の画像を、前記ＷＥＢブラウザの該当する領域に、それまで該領域に表示されていた画像の手前にｚインデックスを変更することにより配置して表示する請求項１から１２のいずれか１つに記載の画像表示更新方法。
【請求項１４】
前記端末装置の描画操作エコーバックスクリプトは、前記画像が表示された画面上で前記描画操作がされたときに、１回のＸＭＬＨＴＴＰリクエストにつき、該描画操作によるカーソルの移動軌跡に応じた該カーソル位置の複数の座標位置の情報を前記サーバに送信し、かつ該複数の座標位置に基づく画像加工処理全体で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を取得し、かつ該単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信する請求項１から１３のいずれか１つに記載の画像表示更新方法。
【請求項１５】
前記ポインティングデバイスの描画操作態様が該ポインティングデバイスの移動速度である請求項１から１４のいずれか１つに記載の画像表示更新方法。
【請求項１６】
請求項１から１５のいずれか１つに記載の画像表示更新方法を実行するサーバ・クライアントシステム。
【請求項１７】
請求項１から１５のいずれかに記載の画像表示更新方法を実行するために前記端末装置

で行う処理を該端末装置に実行させるプログラムであって、前記サーバから送信され前記端末装置の前記ＷＥＢブラウザに付加される描画操作エコーバックスクリプト。

【手続補正３】
【補正対象書類名】明細書
【補正対象項目名】０００１
【補正方法】変更
【補正の内容】
【０００１】
この発明は端末装置のＷＥＢブラウザ画面上での描画操作に基づきサーバ側に保管されている画像に対し濃度表現を有する筆ツールによる加工を施す方法において、描画操作に応じて加工される画像が該描画操作から大きく遅れることなくＷＥＢブラウザの画面に反映されるようにしたものである。またこの発明は端末装置のＷＥＢブラウザ画面上での描画操作に基づきサーバ側に保管されている画像に対し濃度表現を有する筆ツールによる加工を施す方法において、描画操作に応じて時々刻々加工されていく画像が該描画操作から大きく遅れることなくＷＥＢブラウザの画面に反映されるようにして、端末装置の操作者が思い通りの描画操作をできるようにしたものである。

【手続補正４】
【補正対象書類名】明細書
【補正対象項目名】０００８
【補正方法】変更
【補正の内容】
【０００８】
この発明は上述の点に鑑みてなされたもので、端末装置のＷＥＢブラウザ画面上での描画操作に基づきサーバ側に保管されている画像に対し濃度表現を有する筆ツールによる加工を施す方法において、描画操作に応じて加工される画像が該描画操作から大きく遅れることなくＷＥＢブラウザの画面に反映されるようにした手法を提供しようとするものである。またこの発明は端末装置のＷＥＢブラウザ画面上での描画操作に基づきサーバ側に保管されている画像に対し濃度表現を有する筆ツールによる加工を施す方法において、描画操作に応じて時々刻々加工されていく画像が該描画操作から大きく遅れることなくＷＥＢブラウザの画面に反映されるようにして、端末装置の操作者が思い通りの描画操作をできるようにした手法を提供しようとするものである。

【手続補正５】
【補正対象書類名】明細書
【補正対象項目名】０００９
【補正方法】変更
【補正の内容】
【０００９】
この発明の画像表示更新方法は、端末装置にインストールされたＷＥＢブラウザの画面に画像を表示し、該画面上で該端末装置の操作者がポインティングデバイスで描画操作をすることにより、該ポインティングデバイスによるカーソル位置の画像の表示に濃度表現を有する筆ツールによる変更を加える画像表示更新方法であって、サーバに、前記操作者による描画操作対象となる画像のデータと、該描画操作に応じて該画像データに対して画像加工処理を実行する画像処理プログラムを保持管理し、前記端末装置と前記サーバをネットワークを介して相互に通信可能な状態に設定し、前記端末装置のＷＥＢブラウザが前記サーバに対しＨＴＭＬドキュメントをＨＴＴＰリクエストによって要求すると、該サーバは該ＷＥＢブラウザによって解釈可能な描画操作エコーバックスクリプトが内部に記述されたＨＴＭＬドキュメントまたは該描画操作エコーバックスクリプトを外部ファイルとして読み込むことが記述されたＨＴＭＬドキュメントを該端末装置に該ＨＴＴＰリクエストに対する応答として返信し、該端末装置のＷＥＢブラウザは該ＨＴＭＬドキュメントを受信して、該ＷＥＢブラウザに前記描画操作エコーバックスクリプトを付加し、前記ＨＴ

ＭＬドキュメントは、前処理として、前記サーバが保持管理する前記描画操作対象画像をＨＴＴＰリクエストによって該サーバに要求し、その返信として受け取った画像データによる画像を前記ＷＥＢブラウザの画面に表示し、前記端末装置のＷＥＢブラウザに付加された前記描画操作エコーバックスクリプトは、前記画像が表示された画面上で前記描画操作がされたときに、該描画操作に応じたカーソル位置の座標情報をＸＭＬＨＴＴＰリクエストによって前記サーバに送信し、前記サーバの画像処理プログラムは、受信した該カーソル位置の座標情報と、別途取得しまたは初めから保持する筆の描画サイズの情報に基づき、自らが保持管理する前記描画操作対象画像に対して、前記濃度表現を有する筆ツールによる画像加工処理を実行し、前記描画操作エコーバックスクリプトは前記カーソル位置の座標情報を送信後に、前記ＷＥＢブラウザの画面に表示された画像を複数の領域に分割する矩形領域のうち前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を取得し、かつ該単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信し、前記サーバは該単一または複数のＨＴＴＰリクエストを受信して、前記ＷＥＢブラウザの画面に表示された画像の該当する単一または複数の矩形領域の前記画像加工処理が終了した画像データを選択的に返信し、前記ＷＥＢブラウザは受信した単一または複数の矩形領域の画像データによる画像を、該ＷＥＢブラウザの画面の該当する矩形領域に、それまで該矩形領域に表示されていた画像に代えて表示するものである。

【手続補正６】
【補正対象書類名】明細書
【補正対象項目名】００１１
【補正方法】変更
【補正の内容】
【００１１】

この発明によれば、ＷＥＢブラウザの画面に表示された画像を複数の領域に分割する矩形領域のうち、描画操作に基づく画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の画像を選択的にサーバからＷＥＢブラウザに送信して該ＷＥＢブラウザの画面表示を更新するので、ＷＥＢブラウザの画面に表示された画像全体分の画像をサーバからＷＥＢブラウザに送信して該ＷＥＢブラウザの画面に表示された画像全体の表示を更新する場合に比べて、更新画像のダウンロードに要する時間を短縮することができる。その反面、この手法によればＷＥＢブラウザの画面に表示された画像全体の表示を更新する場合に比べて、画像内容に変更が生じる領域を計算する分の時間が必要となる。そこでこの発明では描画操作エコーバックスクリプトは画像が表示された画面上で描画操作がされたときに、該描画操作に応じたカーソル位置の座標情報をＸＭＬＨＴＴＰリクエストによってサーバにいち早く送信するようにしている（加工処理された画像を要求するＨＴＴＰリクエストはこの時点では送信せず、後に別途送信する。画像加工処理で画像内容に変更が生じる領域の計算は該カーソル位置の座標情報をサーバに送信した後に端末装置側またはサーバ側で行うことができる）。したがってサーバの画像処理プログラムは描画操作から短時間のうちに該描画操作に対応した画像加工処理を開始することができる。したがって描画操作がされたときに、加工が行われる領域の計算が済んでから画像加工処理を実行するのに必要な情報をサーバに送る場合に比べて、描画操作がされてから画像加工処理が終了するまでの時間を短縮することができる。これにより描画操作エコーバックスクリプトは前記ＸＭＬＨＴＴＰリクエストを送信した後に、加工処理された画像を要求するＨＴＴＰリクエストを送信すると、該当する加工処理された画像を少ない待ち時間で取得することができる。

【手続補正７】
【補正対象書類名】明細書
【補正対象項目名】００１２
【補正方法】変更
【補正の内容】

【００１２】

またこの発明の画像表示更新方法は、端末装置にインストールされたＷＥＢブラウザの画面に画像を表示し、該画面上で該端末装置の操作者がポインティングデバイスで描画操作をすることにより、該ポインティングデバイスによるカーソル位置の画像の表示に濃度表現を有する筆ツールによる変更を加える画像表示更新方法であって、サーバに、前記操作者による描画操作対象となる画像のデータと、該描画操作に応じて該画像データに対して画像加工処理を実行する画像処理プログラムを保持管理し、前記端末装置と前記サーバをネットワークを介して相互に通信可能な状態に設定し、前記端末装置のＷＥＢブラウザが前記サーバに対しＨＴＭＬドキュメントをＨＴＴＰリクエストによって要求すると、該サーバは該ＷＥＢブラウザによって解釈可能な描画操作エコーバックスクリプトが内部に記述されたＨＴＭＬドキュメントまたは該描画操作エコーバックスクリプトを外部ファイルとして読み込むことが記述されたＨＴＭＬドキュメントを該端末装置に該ＨＴＴＰリクエストに対する応答として返信し、該端末装置のＷＥＢブラウザは該ＨＴＭＬドキュメントを受信して、該ＷＥＢブラウザに前記描画操作エコーバックスクリプトを付加し、前記ＨＴＭＬドキュメントは、前処理として、前記サーバが保持管理する前記描画操作対象画像をＨＴＴＰリクエストによって該サーバに要求し、その返信として受け取った画像データによる画像を前記ＷＥＢブラウザの画面に表示し、前記端末装置のＷＥＢブラウザに付加された前記描画操作エコーバックスクリプトは、前記画像が表示された画面上で前記描画操作がされたときに、該描画操作に応じたカーソル位置の座標情報をＸＭＬＨＴＴＰリクエストによって前記サーバに送信し、前記サーバの画像処理プログラムは、受信した該カーソル位置の座標情報と、別途取得しまたは初めから保持する筆の描画サイズの情報に基づき、自らが保持管理する前記描画操作対象画像に対して、前記濃度表現を有する筆ツールによる画像加工処理を実行し、前記描画操作エコーバックスクリプトは前記カーソル位置の座標情報を送信後に、前記ＷＥＢブラウザの画面に表示された画像を複数の領域に分割する矩形領域のうち前記画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を取得し、かつ該単一または複数の矩形領域の画像を選択的に要求する単一または複数のＨＴＴＰリクエストを該サーバに送信し、前記サーバは該単一または複数のＨＴＴＰリクエストを受信して、前記ＷＥＢブラウザの画面に表示された画像の該当する単一または複数の矩形領域の前記画像加工処理が終了した画像データを選択的に返信し、前記ＷＥＢブラウザは受信した単一または複数の矩形領域の画像データによる画像を、該ＷＥＢブラウザの画面の該当する矩形領域に、それまで該矩形領域に表示されていた画像に代えて表示し、該ＷＥＢブラウザは前記ＸＭＬＨＴＴＰリクエストによるカーソル位置の座標情報の送信を、それ以前のＸＭＬＨＴＴＰリクエストにより送信されたカーソル位置の座標情報に基づき加工処理された画像が該ＷＥＢブラウザの画面に表示されるのを待たずに、前回のＸＭＬＨＴＴＰリクエストに対する応答を受信後に順次実行し、前記サーバの画像処理プログラムは該ＸＭＬＨＴＴＰリクエストに基づき、前記画像加工処理がされた画像に対しさらに該画像加工処理を繰り返すものである。

この発明によれば前述の効果に加えて、描画操作に応じて時々刻々加工されていく画像を該描画操作から大きく遅れることなくＷＥＢブラウザの画面に反映させることが可能になり、端末装置の操作者は濃度表現を有する筆ツールを用いた描画を自分の思い通りに実行することが可能になる。また描画操作エコーバックスクリプトはＸＭＬＨＴＴＰリクエストによるカーソル位置の座標情報の送信を、それ以前のＸＭＬＨＴＴＰリクエストにより送信されたカーソル位置の座標情報に基づき加工処理された画像がブラウザの画面に表示されるの待たずに順次実行するので、サーバの画像処理プログラムは描画操作に追従して画像加工処理を実行することができる。このことも、描画操作に応じて時々刻々加工されていく画像が該描画操作から大きく遅れることなくＷＥＢブラウザの画面に反映されることに寄与している。

【手続補正８】

【補正対象書類名】明細書

【補正対象項目名】００１５

【補正方法】変更
【補正の内容】
【００１５】
この発明の画像表示更新方法において、サーバの画像処理プログラムは端末装置から受信したカーソル位置の座標情報と、該端末装置から該カーソル位置の座標情報と同時に受信しまたは該サーバが該カーソル位置の座標情報を受信する以前から保持している筆の描画サイズの情報に基づき画像加工処理を実行し、かつ該画像描画処理により描画するまたは描画した筆の描画中心位置の座標情報をＸＭＬＨＴＴＰリクエストに対する応答として端末装置のＷＥＢブラウザに返信し、描画操作エコーバックスクリプトは該サーバから受信する筆の描画中心位置の座標情報と、該サーバから該筆の描画中心位置の座標情報と同時に受信しまたは該描画操作エコーバックスクリプトが該筆の描画中心位置の座標情報を受信する以前から保持している筆の描画サイズの情報を利用して、画像加工処理で画像内容に変更が生じる領域を含む単一または複数の矩形領域の情報を自ら計算して取得することができる。この場合筆の描画中心位置の座標情報はカーソル位置の座標情報そのものまたは該カーソル位置の座標間を該カーソル位置の座標情報で補間処理した座標情報とすることができる。
【手続補正９】
【補正対象書類名】明細書
【補正対象項目名】００２２
【補正方法】変更
【補正の内容】
【００２２】
この発明のサーバ・クライアントシステムはこの発明の画像表示更新方法を実行するものである。この発明の描画操作エコーバックスクリプトはこの発明の画像表示更新方法を実行するために前記端末装置で行う処理を該端末装置に実行させるプログラムであって、前記サーバから送信され前記端末装置の前記ＷＥＢブラウザに付加されるものである。