(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2016-14918
(P2016-14918A)
(43)公開日　平成28年1月28日(2016.1.28)

(51)Int.Cl.		F I			テーマコード（参考）
G06T 1/20	(2006.01)	G06T	1/20	A	5B057

審査請求 未請求　請求項の数 12　ＯＬ　（全 34 頁）

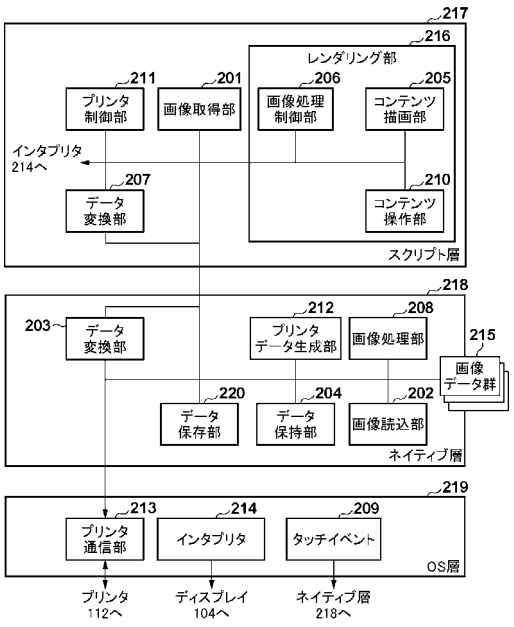
(21)出願番号　特願2014-135176 (P2014-135176)
(22)出願日　平成26年6月30日 (2014.6.30)

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二
(74)代理人　100130409
弁理士　下山　治
(74)代理人　100134175
弁理士　永川　行光

最終頁に続く

(54)【発明の名称】情報処理装置、制御方法、及びプログラム

(57)【要約】
【課題】　より柔軟かつ利便性の高い装置の使用環境を提供する。
【解決手段】　プロセッサで翻訳され実行されるための命令セットを含む第１のプログラム層と、プロセッサ以外で予め翻訳された命令セットを含む第２のプログラム層とを包含するプログラムにおいて、第１のプログラム層で指定される処理対象の画像データに対して色変換を含む画像処理を実行する。色変換された画像データを、第１のプログラム層が利用できる形式の画像データに変換する。
【選択図】　図２



【特許請求の範囲】
【請求項１】
プロセッサで翻訳され実行されるための命令セットを含む第１のプログラム層と、前記プロセッサ以外で予め翻訳された命令セットを含む第２のプログラム層とを包含するプログラムを、前記プロセッサで実行する情報処理装置であって、
前記第１のプログラム層で指定される処理対象の画像データに対して色変換を含む画像処理を実行する画像処理手段と、
前記画像処理手段で色変換された画像データを、前記第１のプログラム層が利用できる形式の画像データに変換する変換手段と
を備え、
前記第２のプログラム層が、前記画像処理手段を備える
ことを特徴とする情報処理装置。
【請求項２】
プロセッサで翻訳され実行されるための命令セットを含む第１のプログラム層と、前記プロセッサ以外で予め翻訳された命令セットを含む第２のプログラム層とを包含するプログラムを、前記プロセッサで実行する情報処理装置であって、
前記第１のプログラム層で指定される処理対象の画像データを第２のプログラム層が利用できる形式の画像データに変換する変換手段と、
前記変換手段で変換された画像データに対して色変換を含む画像処理を実行する画像処理手段を備え、
前記第２のプログラム層が、前記画像処理手段を備える
ことを特徴とする情報処理装置。
【請求項３】
前記第１のプログラム層は、ウェブ標準言語で記述されている
ことを特徴とする請求項１または２に記載の情報処理装置。
【請求項４】
前記第２のプログラム層は、前記画像データを色変換するための、出力先のデバイスに対応する色変換データを保持する保持手段を備え、
前記画像処理手段は、前記色変換データに基づいて、前記画像データの色変換を行う
ことを特徴する請求項１乃至３のいずれか１項に記載の情報処理装置。
【請求項５】
前記第２のプログラム層は、前記画像データを色変換するための、出力先のデバイスに対応する色変換データを取得して、保存する保存手段を備え、
前記画像処理手段は、前記色変換データに基づいて、前記画像データの色変換を行う
ことを特徴とする請求項１乃至４のいずれか１項に記載の情報処理装置。
【請求項６】
前記第１のプログラム層は、出力先のデバイスに対応する色変換データを選択する選択手段を備え、
前記画像処理手段は、前記選択手段で選択された色変換データに基づいて、前記画像データの色変換を行う
ことを特徴とする請求項１乃至５のいずれか１項に記載の情報処理装置。
【請求項７】
前記画像処理手段は、前記画像データに含まれる色変換データに基づいて、該画像データの色変換を行う
ことを請求項１乃至６のいずれか１項に記載の情報処理装置。
【請求項８】
前記画像処理手段は、当該情報処理装置のオペレーティングシステムに含まれる、出力先のデバイスに対応する色変換データに基づいて、前記画像データの色変換を行う
ことを特徴とする請求項１乃至７のいずれか１項に記載の情報処理装置。
【請求項９】

前記画像処理手段は、当該情報処理装置と接続する外部デバイスが保持する、出力先のデバイスに対応する色変換データに基づいて、前記画像データの色変換を行う
ことを特徴とする請求項１乃至７のいずれか１項に記載の情報処理装置。
【請求項１０】
プロセッサで翻訳され実行されるための命令セットを含む第１のプログラム層と、前記プロセッサ以外で予め翻訳された命令セットを含む第２のプログラム層とを包含するプログラムを、前記プロセッサで実行する情報処理装置の制御方法であって、
前記第１のプログラム層で指定される処理対象の画像データに対して色変換を含む画像処理を実行する画像処理工程と、
前記画像処理工程で色変換された画像データを、前記第１のプログラム層が利用できる形式の画像データに変換する変換工程と
を備え、
前記第２のプログラム層が、前記画像処理工程を備える
ことを特徴とする情報処理装置の制御方法。
【請求項１１】
プロセッサで翻訳され実行されるための命令セットを含む第１のプログラム層と、前記プロセッサ以外で予め翻訳された命令セットを含む第２のプログラム層とを包含するプログラムを、前記プロセッサで実行する情報処理装置の制御方法であって、
前記第１のプログラム層で指定される処理対象の画像データを第２のプログラム層が利用できる形式の画像データに変換する変換工程と、
前記変換工程で変換された画像データに対して色変換を含む画像処理を実行する画像処理工程を備え、
前記第２のプログラム層が、前記画像処理工程を備える
ことを特徴とする情報処理装置の制御方法。
【請求項１２】
コンピュータを、請求項１乃至９のいずれか１項に記載の情報処理装置の各手段として機能させるための、または請求項１０または１１に記載の情報処理装置の制御方法をコンピュータに実行させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、情報処理技術に関するものである。
【背景技術】
【０００２】
可搬型多機能携帯端末（以下、モバイルコンピュータ）は、基本的には３つの要素で成り立っている。即ち、コンピュータ自身であるハードウェアと、該ハードウェア上で動作するオペレーティングシステム（以下、ＯＳ）と、該ＯＳ上で動作するアプリケーションである。ユーザは、該アプリケーションを用いて、地図やメール、インターネット上のウェブサイトの閲覧等の機能を使用することが可能である。
【０００３】
このようなモバイルコンピュータ上で動作するアプリケーションの形態としては、主に二つのものが存在する。即ち、ネイティブアプリケーションとウェブアプリケーションである。以下、それぞれの特徴を説明する。
【０００４】
まず、ネイティブアプリケーションとは、通常、ＯＳ毎に用意される開発環境、及び開発言語を用いて開発される。例えば、Ａ社が提供するＯＳ上ではＣ／Ｃ＋＋言語、Ｂ社が提供するＯＳ上ではＪａｖａ（登録商標）言語、Ｃ社が提供するＯＳ上では更に異なる開発言語を用いる、という具合である。通常、ネイティブアプリケーションは、各開発環境において予めコンパイルされ、人間が理解可能ないわゆる高水準言語から、コンピュータのＣＰＵが解釈可能なアセンブラ等の命令セット群に変換される。このように、通常のネ

イティブアプリケーションでは、命令をＣＰＵが直接解釈するために、高速動作が可能である、というメリットが存在する。
【０００５】
一方、ウェブアプリケーションとは、近年では、各コンピュータ上のＯＳに標準的に組み込まれているウェブブラウザ上で動作するアプリケーションのことである。アプリケーションはウェブブラウザが解釈できるよう、一般的には、ＨＴＭＬ５及びＣＳＳ、さらにＪａｖａＳｃｒｉｐｔ（登録商標）等の言語を用いて開発される。これらはウェブ標準言語であるため、これらのウェブ標準言語でウェブアプリケーションを一旦記述すれば、ウェブブラウザが動作する環境であれば、どこでも動作可能というメリットが存在する。
【０００６】
特許文献１には、ウェブブラウザ上で動作するウェブアプリケーションにより印刷を実行することが開示されている。
【先行技術文献】
【特許文献】
【０００７】
【特許文献１】特開２０１１－２３３０３４号公報
【発明の概要】
【発明が解決しようとする課題】
【０００８】
上述のように、近年のモバイルコンピュータに写真等の画像が記憶されることがある。これらの画像に対して、例えば、モノクロ・セピア調にするフィルタ処理を施したり、写真の明るさ、色バランス等を修正するための画像処理は、ユーザにとっても大変重要で欠かせないアプリケーションとなっている。このような画像処理をユーザに対して、ストレスなく、簡単に提供できるようなアプリケーションを、上記２つのアプリケーション形態で作成することを考えると、以下のようなことが発生し得る。
【０００９】
まず、ネイティブアプリケーションでは、上述のように処理を高速実行できるというメリットがある。しかし、本形態は、ＯＳ毎に異なる開発言語で別々に開発する必要があるため、開発コストと開発時間が増大し、ユーザに迅速に提供できない場合がある。また、ネイティブアプリケーションは、予めコンパイル（翻訳）する必要がある。そのため、例えば、アプリケーションのＵＩ（ユーザインタフェース）デザインを動作時に変更したり、機能を動的に追加する等は難しく、柔軟性に欠ける場合がある。
【００１０】
また、ウェブアプリケーションでは、ＨＴＭＬ５、ＣＳＳ、ＪａｖａＳｃｒｉｐｔで記述されたウェブアプリケーションの本体が、モバイルコンピュータ外部のサーバ上に存在することが考えられる。ウェブアプリケーションは、利用時に動的にサーバからモバイルコンピュータにインターネット回線を介してダウンロードするため、ＵＩデザイン等を予めコンパイルすることなく動的に変更することが可能である。
【００１１】
しかし、高度で複雑な処理を実行する場合、ウェブアプリケーションは、ブラウザのセキュリティ上の制約から、ブラウザ上でＪａｖａＳｃｒｉｐｔによって実行するか、サーバ上で実行するという２つの選択肢しか存在しない。従来、ＪａｖａＳｃｒｉｐｔは、人間が視認可能な文字列のスクリプトとして記述され、該スクリプトを動作時に随時コンパイルすることで実行することができる。このことから、複雑な処理をＪａｖａＳｃｒｉｐｔで記述すると動作が遅くなる場合がある。
【００１２】
一方、この複雑な処理をサーバで実行するように構築した場合、モバイルコンピュータ内部に存在する写真等のデータを、インターネット回線を介してサーバ上にアップロードし、サーバで処理後の結果を、今度はサーバからダウンロードする時間が必要となる。このような構成は、モバイルアプリケーションに対してストレスの少ない、即時的な処理を

実現することができない場合がある。
【００１３】
一方、従来から、カメラ、スキャナ、ディスプレイ、プリンタ等の異なるデバイス間の色が異なることが問題となることから、統一的に色を管理するためのカラーマネージメントシステム（ＣＭＳ）がある。モバイルコンピュータにおいても、同様に、ＣＭＳへの対応が求められる。しかしながら、ＯＳ毎にＣＭＳ対応の有無や、色変換を行うためのカラーマネージメントモジュールが異なっている場合がある。また、アプリケーション毎の特色を活かすために、独自のカラーマネージメントを実施したい要望がある。
【００１４】
ウェブアプリケーションでは、このＣＭＳに関する処理を高速に実行できず、本来ならば、ネイティブアプリケーションで実行することが望ましい。一方、ウェブアプリケーションにて扱う画像データのデータ形式は、ネイティブアプリケーションが扱う画像データの形式とは異なる。そのため、ウェブアプリケーションにて表示されていた画像データについて画像処理が要求された場合、ウェブアプリケーションが取り扱っていた画像データがネイティブアプリケーションに送信されても、ネイティブアプリケーションが画像処理を実行できないおそれがあった。また、ネイティブアプリケーションが画像データに対して画像処理を行ったとしても、画像処理が適用された画像データをウェブアプリケーションが表示できないおそれがあった。
【００１５】
本発明は上記の課題を解決するためになされたものであり、より柔軟かつ利便性の高い装置の使用環境を提供することを目的とする。
【課題を解決するための手段】
【００１６】
上記の目的を達成するための本発明による情報処理装置の構成は以下の構成を備える。即ち、
プロセッサで翻訳され実行されるための命令セットを含む第１のプログラム層と、前記プロセッサ以外で予め翻訳された命令セットを含む第２のプログラム層とを包含するプログラムを、前記プロセッサで実行する情報処理装置であって、
前記第１のプログラム層で指定される処理対象の画像データに対して色変換を含む画像処理を実行する画像処理手段と、
前記画像処理手段で色変換された画像データを、前記第１のプログラム層が利用できる形式の画像データに変換する変換手段と
を備え、
前記第２のプログラム層が、前記画像処理手段を備える。
【発明の効果】
【００１７】
本発明によれば、より柔軟かつ利便性の高い装置の使用環境を提供できる。
【図面の簡単な説明】
【００１８】
【図１】情報処理装置の構成を示すブロック図である。
【図２】情報処理装置のソフトウェアの構成を示すブロック図である。
【図３】ユーザ操作に伴う処理を示すフローチャートである。
【図４】写真画像選択の詳細を示すフローチャートである。
【図５】画像処理の詳細を示すフローチャートである。
【図６】スタンプ追加の詳細を示すフローチャートである。
【図７】スタンプ特定の詳細を示すフローチャートである。
【図８】スタンプ操作の詳細を示すフローチャートである。
【図９】プリンタ設定の詳細を示すフローチャートである。
【図１０】レンダリングの詳細を示すフローチャートである。
【図１１】プリントの詳細を示すフローチャートである。

【図１２】アプリケーション画面の一例を示す図である。
【図１３】設定画面の一例を示す図である。
【図１４】設定画面の一例を示す図である。
【図１５】設定画面の一例を示す図である。
【発明を実施するための形態】
【００１９】
以下、本発明の実施の形態について図面を用いて詳細に説明する。
【００２０】
＜実施形態１＞
本実施形態では、情報処理装置上で、後述するハイブリッド型アプリケーションを動作させ、ユーザが選択した画像に対して、様々な画像処理を適用した後に、その画像を印刷する構成について説明する。
【００２１】
＜ハードウェア構成の説明＞
図１は情報処理装置１１５として、例えば、スマートフォンや携帯電話等の携帯型情報端末の構成例を説明するブロック図である。同図において、１００はＣＰＵ（中央演算装置／プロセッサ）であり、以下で説明する各種処理をプログラムに従って実行する。図中のＣＰＵ１００は１つであるが、複数のＣＰＵあるいはＣＰＵコアによって構成されていても良い。１０１はＲＯＭであり、ＣＰＵ１００により実行されるプログラムが記憶されている。１０２はＲＡＭであり、ＣＰＵ１００によるプログラムの実行時に、各種情報を一時的に記憶するためのメモリである。
【００２２】
１０３はハードディスクやフラッシュメモリ等の２次記憶装置であり、ファイルや画像解析等の処理結果を保持するデータベース等のデータや、各種プログラムを記憶するための記憶媒体である。１０４はディスプレイであり、各種処理を実現するための操作を受け付けるためのＵＩ（ユーザインタフェース）や、実行された処理による処理結果等の各種情報を表示する。ディスプレイ１０４は、タッチセンサ１０５を備えても良い。
【００２３】
情報処理装置１１５は、内部撮像デバイス１１０を備えてもよい。内部撮像デバイス１１０による撮像によって得られた画像データは、所定の画像処理を経た後、２次記憶装置１０３に保存される。また、画像データは、外部Ｉ／Ｆ１０８を介して接続された外部撮像デバイス１１１から読み込むこともできる。
【００２４】
情報処理装置１１５は、外部Ｉ／Ｆ１０９を備え、インターネット等のネットワーク１１３を介して通信を行うことができる。情報処理装置１１５は、この通信Ｉ／Ｆ１０９を介して、ネットワーク１１３に接続されたサーバ１１４より画像データを取得することもできる。
【００２５】
情報処理装置１１５は、加速度センサ１０６を備え、情報処理装置１１５自身の位置姿勢に関する加速度情報を取得することができる。情報処理装置１１５は、外部Ｉ／Ｆ１０７を介し、プリンタ１１２と接続されており、画像データ等のデータを出力することができる。プリンタ１１２は、ネットワーク１１３にも接続されており、通信Ｉ／Ｆ１０９経由で、画像データを送受信することができる。
【００２６】
外部Ｉ／Ｆ１０７～１０９は、有線通信と無線通信の内、少なくともいずれかの通信形態を有するインタフェースであり、利用する通信形態に応じて外部デバイス（プリンタ１１２あるいはサーバ１１４）との通信を行う。有線通信には、例えば、ＵＳＢ、イーサネット（登録商標）等があり、無線通信には、無線ＬＡＮ、ＮＦＣ、ブルートゥース、赤外線通信等がある。また、無線通信として、無線ＬＡＮを利用する場合には、装置同士が直接接続する形態もあれば、無線ＬＡＮルータ等の中継装置を介して接続する形態もある。

また、外部Ｉ／Ｆ１０７～１０９は、図では別々に構成されているが、一体となって構成されていても良い。

【００２７】

情報処理装置１１５の動作に必要な電源は、バッテリ１１７によって供給される。情報処理装置１１５が備える各種構成要素は、制御バス／データバス１１６を介して相互に接続され、ＣＰＵ１００は、この制御バス／データバス１１６を介して、各種構成要素を制御する。

【００２８】

尚、本実施形態では、情報処理装置１１５が、その情報処理装置１１５が備える制御部（ＣＰＵ１００）によって実行されるプログラム等のソフトウェアの実行場所（ソフトウェア実行環境）となる。

【００２９】

＜ソフトウェアのブロック図＞

図２は情報処理装置１１５で動作するソフトウェア構成のブロック図である。

【００３０】

情報処理装置１１５は、スクリプト層２１７、ネイティブ層２１８、及びＯＳ層２１９のプログラムを実行する。これらの各層は、ＣＰＵ１００がＲＯＭ１０１あるいは２次記憶装置１０３に記憶されている対応するプログラムを読み出し実行することにより実現される。

【００３１】

スクリプト層２１７は、ＨＴＭＬ５やＣＳＳ３、及びＪａｖａＳｃｒｉｐｔ等のウェブ標準言語を使って、テキストデータで命令セット（コンテンツの描画や画像の表示、動画の再生等）が記述されているプログラム層である。このスクリプト層２１７では、アプリケーション実行環境上で、そのアプリケーション実行環境に存在するプロセッサ（例えば、ＣＰＵ１００）を用いて、テキストデータの各種命令セットを翻訳して実行することになる。形態としては、実行の度に命令文を一行ずつ動的に翻訳する場合や、アプリケーションを起動したときに翻訳する場合、アプリケーションを情報処理装置１１５にインストールしたときに翻訳する場合等が考えられる。

【００３２】

以後、スクリプト層２１７で処理することや内容をスクリプトと呼ぶ。スクリプトの命令を情報処理装置１１５内で翻訳する形態の例として、ネイティブ層２１８やＯＳ層２１９が備えるインタプリタの機能を使用することが挙げられる。尚、本実施形態においては、アプリケーションのＵＩの大部分が、このスクリプト層２１７で記述されていることを想定している。

【００３３】

ネイティブ層２１８は、アプリケーション実行環境以外で予め翻訳（コンパイル）された命令セットが記述されているプログラム層である。形態としては、ＣもしくはＣ＋＋といった高水準言語で記述されたコードが、予めアプリケーションの開発者のＰＣやサーバ上でコンパイルされ、ＣＰＵ１００が解釈できる命令の集合体となっている。以後、ネイティブ層２１８で処理することや内容、後述するＯＳ層２１９の機能をネイティブ層２１８から呼び出すことを含め、ネイティブと呼ぶこととする。尚、ネイティブ層２１８の別の実装系として、Ｊａｖａが挙げられる。Ｊａｖａは、Ｃ／Ｃ＋＋と類似の高水準言語であり、予めアプリケーション開発時の開発環境上で中間コードに翻訳される。翻訳された中間コードは、各ＯＳが備えるＪａｖａ仮想環境上で動作する。本実施形態においては、このようなプログラム形態も、ネイティブ層２１８の一種に含める。

【００３４】

ＯＳ層２１９は、情報処理装置１１５のオペレーティングシステム（Ｏｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ：ＯＳ）に対応する。ＯＳ層２１９は、ハードウェア機能の使用をアプリケーションに提供する役割及び固有の機能を有する。ＯＳ層２１９は、ＡＰＩを備え、スクリプト層２１７やネイティブ層２１８から機能を使用することができる。

【００３５】
本実施形態では、スクリプト層２１７からネイティブ層２１８の呼び出しを可能にすることをバインディング、もしくはバインドと呼ぶ。各種ネイティブの機能は、ＡＰＩを備え、該ＡＰＩをスクリプトが呼び出すことでネイティブの機能を使用することができる。このようなバインディング機能は、通常、各種ＯＳが標準的に備えている機能である。
【００３６】
尚、本実施形態では、スクリプト層２１７とネイティブ層２１８を含むアプリケーションのことをハイブリッド型アプリケーションと呼ぶ。
【００３７】
スクリプト層２１７の画像取得部２０１は、ネイティブ層２１８に対し画像データの取得を依頼する。取得依頼時に、画像取得部２０１は、一意なＩＤを生成し、ネイティブ層２１８に送信する。このＩＤと、ネイティブ層２１８の画像読込部２０２で読み込まれた画像データは、対となって、ネイティブ層２１８のデータ保持部２０４に記憶される。これ以外にも、例えば、絶対パスを指定する方法や、ダイアログ表示を促す方法等が挙げられる。
【００３８】
ネイティブ層２１８の画像読込部２０２は、画像データ群２１５から画像データを取得する。画像データ群２１５からの画像データの取得方法は、スクリプト層２１７の画像取得部２０１の依頼に依存する。依頼方法は、ＵＩ上に提供されているダイアログボックスから選択する、ファイルのパスから直接画像を選択する等が挙げられる。
【００３９】
ネイティブ層２１８のデータ変換部２０３は、ネイティブ層２１８のデータ（例：バイナリ形式の画像データ）をスクリプト層２１７で利用できる形式のデータ（例：テキスト形式（ＢＡＳＥ６４）の画像データ）に変換する。一方で、データ変換部２０３は、スクリプト層２１７から送られてきたデータ（例：テキスト形式（ＢＡＳＥ６４）の画像データ）をネイティブ層２１８で利用できる形式（例：バイナリ形式の画像データ）に変換する。
【００４０】
スクリプト層２１７のデータ変換部２０７は、スクリプト層２１７のデータ（例：テキスト形式の処理パラメータ）をネイティブ層２１８で利用できる形式のデータ（例：テキスト形式（ＪＳＯＮ形式）の処理パラメータ）に変換する。一方で、データ変換部２０７は、ネイティブ層２１８から送られてきたデータをスクリプト層２１７で利用できる形式にする変換も行う。
【００４１】
ネイティブ層２１８のデータ保持部２０４は、画像読込部２０２で読み込んだ画像データ、画像処理部２０８で画像処理が施された画像データ、画像処理部２０８で色変換に利用する色変換データを保持する。保持される画像データは、例えば、ＲＧＢ画像信号に展開されており、すぐに画像処理が実行できる形式になっている。また、保持されている画像データは、スクリプト層２１７の画像取得部２０１で生成されたＩＤと対になっており、このＩＤを指定することで、データ保持部２０４から対応する画像データを取得することができる。
【００４２】
スクリプト層２１７のコンテンツ描画部２０５は、ネイティブ層２１８のデータ変換部２０３を経由して取得した画像データをディスプレイ１０４に表示する。また、コンテンツ描画部２０５は、コンテンツ操作部２１０で操作された画像データの再描画も行う。コンテンツ操作部２１０は、スクリプト層２１７で画像データを操作する。この操作には、画像データの拡大、移動、回転等が挙げられる。また、コンテンツ描画部２０５は、プリントのためのコンテンツをウェブ標準言語を利用して記述する。この記述には、コンテンツ操作部２１０で操作されたスクリプトも反映される。コンテンツ描画部２０５で記述されたコンテンツのスクリプトは、ＯＳ層２１９のインタプリタ２１４で解釈され、ディス

プレイ１０４に表示される。
【００４３】
画像処理制御部２０６は、画像処理に用いる補正パラメータを決定し、ネイティブ層２１８の画像処理部２０８に画像処理を依頼する。まず、画像処理制御部２０６は、スクリプト層２１７で補正パラメータを設定する。設定された補正パラメータは、データ変換部２０７でネイティブ層２１８へ送信できる形式へ変換される。その後、変換された補正パラメータは、処理対象となる画像データのＩＤと共にネイティブ層２１８へ送信される。
【００４４】
スクリプト層２１７の画像処理部２０８は、画像処理制御部２０６で指定されたＩＤと対応する画像をネイティブ層２１８のデータ保持部２０４から取得し、画像処理を施す。その際、どのような画像処理を施すかは、画像処理制御部２０６で設定された補正パラメータにより決定される。
【００４５】
ＯＳ層２１９のタッチイベント２０９は、ディスプレイ１０４のタッチに関する情報を取得する。タッチに関する情報とは、ディスプレイ１０４のタッチ検知、タッチされた位置情報等が挙げられる。取得したタッチに関する情報は、ネイティブ層２１８経由でスクリプト層２１７のコンテンツ操作部２１０に送信される。
【００４６】
スクリプト層２１７のプリンタ制御部２１１は、レンダリング部２１６へのレンダリング開始依頼、プリンタ検知の依頼、プリンタ設定画面の表示、プリント情報の生成と送信を制御する。プリンタ設定画面では、用紙のサイズ、用紙の種類、カラー・モノクロ等のプリンタ設定がなされる。ここで設定された項目を基に、プリンタデータ生成部２１２でプリンタデータが生成される。
【００４７】
プリンタデータ生成部２１２は、プリンタ制御部２１１からの依頼を基に、プリンタ通信に必要なデータ、コマンドを生成する。プリンタ通信に必要なデータとは、通信プロトコルに則ったデータであり、コマンドとは、印刷やスキャン等、プリンタの動作を決定するためのデータである。
【００４８】
ＯＳ層２１９のプリンタ通信部２１３は、プリンタデータ生成部２１２から受信したプリンタデータを接続しているプリンタ１１２に送信したり、プリンタ１１２からプリンタ１１２に関する情報を受信する。ＯＳ層２１９のインタプリタ２１４は、スクリプト層２１７で生成された命令を解釈・実行する。例えば、画像の描画等の命令は、インタプリタ２１４を通して実行され、ディスプレイ１０４に表示される。
【００４９】
画像データ群２１５は、画像データを保持している領域である。データ保存部２２０は、必要に応じて、データ保持部２０４が保持する画像データを画像データ群２１５に保存する。
【００５０】
レンダリング部２１６は、コンテンツ描画部２０５、画像処理制御部２０６、及びコンテンツ操作部２１０を制御して、処理対象の画像データのレンダリングを行う。このレンダリングには、例えば、スクリプト層２１７で出力解像度の画像を生成することが含まれる。また、スクリプト層２１７におけるレンダリング結果、及び、スクリプト層２１７が生成途中の画像はディスプレイ１０４に表示しない。レンダリング結果は、ネイティブ層２１８のデータ変換部２０３に送信され、プリンタ１１２が利用できる形式の画像データに変換される。
【００５１】
＜ユーザ操作に伴う処理＞
図３はユーザ操作を含む処理を示すフローチャートである。まず、図３を用いて、Ｓ２１からＳ２８の各処理の概要を説明し、詳細は後述する。尚、このフローチャートの各ス

テップの処理は、情報処理装置１１５のＣＰＵ１００が、ＲＯＭ１０１あるいは２次記憶装置１０３に記憶されているプログラムを実行することにより実現される。また、図３に示す各ステップは、ＵＩの１つである図１２に示すアプリケーション画面１２００に対するユーザ操作に従って遷移する。このアプリケーション画面１２００は、スクリプト層２１７によって生成される。このアプリケーション画面１２００の操作は、例えば、タッチセンサ１０５を介して実現される。

【００５２】

Ｓ２１で、ＣＰＵ１００は、アプリケーション画面１２００の写真画像選択ボタン１２０１に対するユーザ操作（タッチ操作も含む。以後も同様）を検知すると、その操作に応じて、任意の画像を選択する。画像を選択すると、ＣＰＵ１００は、アプリケーション画面１２００の描画領域１２０６の全体に選択された画像を表示する。

【００５３】

Ｓ２２では、ＣＰＵ１００は、表示されている画像の輝度を調整するためのスライドバー１２０２に対するユーザ操作を検知すると、そのユーザ操作に応じて、画像処理時に利用する補正パラメータを設定する。そして、ＣＰＵ１００は、設定した補正パラメータに従って、表示されている画像に画像処理を施し、その処理内容及び処理結果を描画領域１２０６に表示する。

【００５４】

Ｓ２３は、ＣＰＵ１００は、スタンプ追加ボタン１２０３に対するユーザ操作を検知すると、スタンプ一覧１２０７を表示する。スタンプ一覧１２０７に対するユーザ操作によってスタンプの選択を検知すると、ＣＰＵ１００は、描画領域１２０６に選択されたスタンプを追加・表示する。

【００５５】

Ｓ２４で、ＣＰＵ１００は、アプリケーション画面１２００に対するユーザ操作に応じて、スタンプを特定する。スタンプの特定とは、ディスプレイ１０４にユーザ操作によってタッチされた座標とスタンプの座標より、スタンプがタッチされたか否かを判断するものである。スタンプがタッチされた場合、そのスタンプは操作受付状態となる。ここでは、ユーザ操作に応じて、スタンプが操作受付状態になっているものとする。操作受付状態については後述する。

【００５６】

Ｓ２５で、ＣＰＵ１００は、操作受付状態になっているスタンプを回転するためのスライドバー１２０４に対するユーザ操作を検知すると、そのユーザ操作に応じて、スタンプ操作として、操作受付状態にあるスタンプを回転する。

【００５７】

Ｓ２６で、ＣＰＵ１００は、プリントボタン１２０５に対するユーザ操作を検知すると、プリントに必要な情報を設定するための設定画面１３０１（図１３）を表示する。プリントに必要な情報とは、例えば、図１３の設定画面１３０１に示されるように、用紙サイズ、用紙種類、印刷品位、縁あり／なしの設定項目がある。これ以外にも、両面／片面、モノクロ・カラー等、使用するプリンタが有する機能に応じて、設定可能な設定項目が構成される。

【００５８】

また、本実施形態では、設定画面１３０１の設定完了ボタン１３０２に対するユーザ操作を検知すると、プリントに必要な更なる情報を設定するための設定画面１４０１（図１４）を表示する。プリントに必要な更なる情報とは、図１４の設定画面１４０１に示されるように、ソースプロファイル、ディスティネーションプロファイル、マッチング方法の設定項目がある。

【００５９】

尚、（カラー）プロファイルとは、デバイスの色特性や見た目の要求仕様を記述したものであり、デバイスの入力と出力の色空間のマッピングや「プロファイル接続空間（ＰＣＳ）」で定義される。ＰＣＳとしては、ＣＩＥＬＡＢ（Ｌ＊ａ＊ｂ＊）やＣＩＥＸＹＺが

ある。マッピングは表形式で示され、表にない値には内挿を行うか、変換のための一連のパラメータを用意することがある。そして、あるプロファイルの画像を別のプロファイルの画像へ変換する場合に、その変換元のプロファイルをソースプロファイル、変換先のプロファイルをディスティネーションプロファイルと呼んでいる。

【００６０】

Ｓ２７で、ＣＰＵ１００は、設定画面１４０１の設定完了ボタン１４０２に対するユーザ操作を検知すると、描画領域に表示されている画像を、プリンタに出力するためのプリント解像度に変換するためのレンダリングを実行する。

【００６１】

Ｓ２８で、ＣＰＵ１００は、プリントの解像度に変換された画像を、プリンタ制御のコマンドと共にプリンタに送信する。以上の処理により、ユーザにより選択された画像がプリンタ１１２でプリントされる。

【００６２】

尚、図３に示す処理は、一例であり、処理内容はこれに限定されず、ステップ群の処理順序もこれに限定されるものではない。また、本実施形態において、プロセッサで翻訳され実行されるための命令セットを含む第１のプログラム層をスクリプト層２１７、プロセッサ以外で予め翻訳された命令セットを含む第２のプログラム層をネイティブ層２１８と定義する。そして、これらの第１のプログラム層と第２のプログラム層とを包含するプログラムがハイブリッド型アプリケーションを実現する。文字列データを第１の形式、バイナリデータを第２の形式と定義する。尚、スクリプト層２１７は、テキスト形式のデータを保持することが可能であり、ネイティブ層２１８は、バイナリ形式のデータを保持することが可能である。

【００６３】

＜プリンタの選択＞

最初に、ユーザ操作によって、図３に示す処理を実現するアプリケーションが起動される際には、アプリケーションは、まず、接続可能な外部デバイス（プリンタ１１２）のディスカバリ処理（不図示）を行う。ディスカバリ処理とは、情報処理装置１１５が存在するネットワーク１１３内において、接続可能なプリンタ１１２のＩＰアドレスを特定する処理のことである。

【００６４】

情報処理装置１１５は、ディスカバリ処理によって取得したＩＰアドレス（場合によっては、複数個）に対して、各種プリンタの属性情報取得の命令を送信して、その返信を取得することができる。生成されたコマンドは、プリンタの通信プロトコルに従った形式で、ネットワーク上のネットワーク機器（例えば、Ｗｉ－Ｆｉ（登録商標）接続されたルータ）に対しブロードキャストされる。ここで、通信方法は、Ｗｉ－Ｆｉダイレクトや電話回線を利用する形態が考えられるが、これに限定されるものではない。コマンドの送信の結果、ネイティブ層２１８は、プリンタからプリンタ名称、機種名等のプリンタに関する情報を含む応答を受信する。ネイティブ層２１８で取得された機種名は、スクリプト層２１７に伝達されて、プリンタのリストを表示することができる。

【００６５】

＜写真画像選択の詳細＞

図３のＳ２１の写真画像選択の詳細について、図４を用いて説明する。尚、Ｓ３０１－Ｓ３０２、Ｓ３０９－Ｓ３１１は、ＣＰＵ１００がスクリプト層２１７のプログラムを用いて実行する処理であり、Ｓ３０３－Ｓ３０８は、ＣＰＵ１００がネイティブ層２１８のプログラムを用いて実行する処理である。

【００６６】

Ｓ３０１で、ＣＰＵ１００は、一意なＩＤを生成する。このＩＤは、数値、文字列等、スクリプト層２１７からネイティブ層２１８へ送信できる形式であればどのような形でも良い。Ｓ３０２で、ＣＰＵ１００は、生成したＩＤとともに、ネイティブ層２１８に、写真画像選択ボタン１２０１に対するユーザ操作に応じて画像選択を依頼する。依頼方法は

、バインディング機能によりスクリプト層２１７からネイティブ層２１８固有の画像選択ＡＰＩを呼び出す。ネイティブ層２１８には、スクリプト層２１７から直接呼び出せる関数もしくはその関数を間接的に呼び出す、いわゆる、ラッパーが予め用意してある。

【００６７】

Ｓ３０３で、ＣＰＵ１００は、デバイス固有の画像選択ＵＩをディスプレイ１０４に表示する。表示された画像選択ＵＩに対するユーザ操作に基づいて、任意の画像を１枚選択する。画像の選択は、例えば、情報処理装置１１５内で管理されるフォルダから画像を１枚選択するものとするが、これに限定されるものではない。例えば、インターネット上の画像や、脱着可能な記憶媒体内の画像を選択しても良いし、情報処理装置１１５の内部撮像デバイス１１０を利用して撮影した画像を取得しても良い。

【００６８】

Ｓ３０４で、ＣＰＵ１００は、選択された画像を取得する。例えば、選択した画像が画像ファイルの状態であれば、ＣＰＵ１００は、その画像ファイルを開き、その内容を読み取る。Ｓ３０５で、ＣＰＵ１００は、取得した画像をＲＧＢ展開する。Ｓ３０６で、ＣＰＵ１００は、展開したＲＧＢ画像と取得した画像のソースプロファイルを、スクリプト層２１７から取得したＩＤと関連付けてデータ保持部２０４に保持する。関連付け方法は、例えば、ＩＤとＲＧＢ画像を有するオブジェクトを作成することで、ＩＤによりＲＧＢ画像の特定を可能にするという方法が考えられる。ここで、関連付け方法は、これに限らず、ＩＤと選択された画像のアクセス先であるパスや、ＩＤとＲＧＢ展開に応じて実行される関数やクラス等も考えられる。

【００６９】

Ｓ３０７で、ＣＰＵ１００は、展開したＲＧＢ画像をスクリプト層２１７でサポート可能な形式のデータに変換する。本実施形態では、Ｓ３０７において、ＲＧＢ画像がＪＰＥＧ（Ｊｏｉｎｔ　Ｐｈｏｔｏｇｒａｐｈｙ　Ｅｘｐｅｒｔ　Ｇｒｏｕｐ）形式のデータに変換される。Ｓ３０８で、ＣＰＵ１００は、ＪＰＥＧ形式のデータをＢＡＳＥ６４データに変換し、スクリプト層２１７へ送信する。これは、スクリプト層２１７では、ＲＧＢ画像のデータ配列をそのまま利用できないため、ネイティブ層２１８において、スクリプト層２１７で利用できる形式に変換する必要があるためである。ＪａｖａＳｃｒｉｐｔ（登録商標）では、文字列しか扱えないため、本実施形態では、文字列としてデータを表現するＢＡＳＥ６４の形式を利用する。

【００７０】

Ｓ３０９で、ＣＰＵ１００は、ネイティブ層２１８で変換されたＢＡＳＥ６４データを受信するとともに、そのＢＡＳＥ６４データを表示するための描画領域をＲＡＭ１０２に確保する。本実施形態では、描画領域の確保の一例としてＨＴＭＬのＣａｎｖａｓ機能を利用し、画像の描画は、Ｃａｎｖａｓの有するＣｏｎｔｅｘｔオブジェクトのＡＰＩを利用する。

【００７１】

Ｓ３１０で、ＣＰＵ１００は、補正パラメータを生成し、初期化する。ここで、補正パラメータとは、Ｓ２２の画像処理の内容を決定するパラメータ群を保持するオブジェクトである。ＪａｖａＳｃｒｉｐｔ（登録商標）で保持する補正パラメータの一例として、下記のような形態が考えられる。

【００７２】

```
var CorrectionParam = function(){
    this.brightness = 0;
}
```

【００７３】

この補正パラメータは、ＣｏｒｒｅｃｔｉｏｎＰａｒａｍオブジェクトの中に、明るさ補正用にｂｒｉｇｈｔｎｅｓｓという名前の変数が設けられており、０という値を格納するということを表している。

【００７４】

本実施形態では、説明の簡略化のために、補正パラメータは明るさ（輝度）補正用のみであるが、その他の補正処理用のパラメータ（ぼかしフィルタの強度、セピア変換のオン・オフ等）を追加しても良い。

【００７５】

Ｓ３１１で、ＣＰＵ１００は、描画領域で描画するデータにＢＡＳＥ６４データを指定し、それに従って、その描画領域に画像を描画する。具体的には、インタプリタ２１４がＢＡＳＥ６４データのスクリプトを解釈し、描画領域に画像として表示する。ここで、描画領域に、ＢＡＳＥ６４データを反映させるサンプルコードの一例を示す。

【００７６】

```
-------------------------------------------------
var base64Data = ネイティブ層からのＢＡＳＥ６４データ
var canvas = document.createElement("canvas");  //画像の描画領域確保
canvas.setAttribute("width", 100);              //描画領域の大きさ設定
canvas.setAttribute("height", 100);
var context = canvas.getContext("2d");  //描画領域に描画するＡＰＩを持つオブジェクトの生成
var img = new Image();     //Imageオブジェクトの生成
img.src = base64Data;      //画像のＵＲＩを受け取ったＢＡＳＥ６４データとする
img.onload = function(){  //画像のロードが終わってから処理を開始する
  context.drawImage(img, 0, 0, img.width, img.height, 0, 0, canvas.width, canvas.height);  //contextオブジェクトのメソッドを用いて画像を描画領域に描画
  document.getElementById("div").appendChild(canvas);
  //本フローチャートではＣａｎｖａｓが何層にもなるレイヤー構造を想定している。
}
```

これらのＣａｎｖａｓは至る所好き勝手に存在するわけではなく、描画、移動、拡大等の操作は、特定の領域内（図１２の描画領域１２０６）で完結する。その領域を指定しているものが「ｄｉｖ」であり、Ｃａｎｖａｓはそのｄｉｖに追加されてゆく形態をとる。

```
-------------------------------------------------
```

【００７７】

＜画像処理の詳細＞

図３のＳ２２の画像処理の詳細について、図５を用いて説明する。尚、Ｓ４０１－Ｓ４０３、Ｓ４０９、Ｓ４１１は、ＣＰＵ１００がスクリプト層２１７のプログラムを用いて実行する処理であり、Ｓ４０４－Ｓ４０８、Ｓ４１０はＣＰＵ１００がネイティブ層２１８のプログラムを用いて実行する処理である。

【００７８】

Ｓ４０１で、ＣＰＵ１００は、補正パラメータを設定する。ここでは、図３のＳ３１０で生成した補正パラメータのｂｒｉｇｈｔｎｅｓｓの値が、スライドバー１２０２に対するユーザ操作に応じて設定される値に更新される。Ｓ４０２で、ＣＰＵ１００は、スクリプト層２１７で、インジケータを起動し、ディスプレイ１０４に表示する。ここで、インジケータとは、ユーザに処理中である旨を伝える表示であり、一般には、プログレスバーや、時計マーク、図形の点滅や回転等の画像で表現する。Ｓ４０３で、ＣＰＵ１００は、設定した補正パラメータをネイティブ層２１８で利用できる形式に変換する。ここで、補正パラメータはオブジェクトの形態を取っており、そのままではネイティブ層２１８で利用できないため、設定した補正パラメータをＪＳＯＮ文字列に変換する。そして、ＣＰＵ１００は、ＪＳＯＮ文字列に変換された補正パラメータを、図３のＳ３０１で生成したＩＤと共にネイティブ層２１８へ送信する。

【００７９】

Ｓ４０４で、ＣＰＵ１００は、ＪＳＯＮ文字列に変換された補正パラメータをデコードし、補正パラメータを取得する。より具体的には、補正パラメータをＯＳ層２１９に備えられているパーサを利用してパースする。パース後に、上記の例の場合、補正パラメータ

内のｂｒｉｇｈｔｎｅｓｓが取得される。
【００８０】
Ｓ４０５で、ＣＰＵ１００は、スクリプト層２１７から取得したＩＤを基に、図３のＳ３０５で展開されたＲＧＢ画像を特定する。尚、上述のように、ＩＤと画像の対応付けは、ＩＤとＲＧＢ画像と対にすることに限定されるわけではなく、例えば、ＩＤと画像のパスを関連付ける方法を用いてもよい。その他にＩＤと関連付ける例として、ネイティブ層２１８のオブジェクト、画像データの先頭アドレス、画像を呼び出す関数等、様々なものが考えられる。
【００８１】
Ｓ４０６で、ＣＰＵ１００は、取得した補正パラメータから施すべき画像処理を判断し、Ｓ４０５において特定されたＲＧＢ画像に対して補正パラメータに対応する画像処理を行う。本実施形態では、明るさ補正のパラメータにより、全ての画素のＲＧＢの値に１０が加算される。
【００８２】
Ｓ４０７で、ＣＰＵ１００は、画像処理が施されたＲＧＢ画像をスクリプト層２１７でサポート可能な形式のデータに変換する。ここでは、図３のＳ３０７と同様に、ＪＰＥＧ形式のデータに変換する。Ｓ４０８で、ＣＰＵ１００は、スクリプト層２１７にインジケータの停止を依頼する。これは、ネイティブ層２１８がスクリプト層２１７で定義されているインジケータ停止の関数を呼び出すことで実現する。
【００８３】
Ｓ４０９で、ＣＰＵ１００は、インジケータを停止して、ディスプレイ１０４の表示からインジケータを消去する。
【００８４】
一方、Ｓ４１０で、ＣＰＵ１００は、変換されたＪＰＥＧ形式のデータをＢＡＳＥ６４データに変換し、スクリプト層２１７へ送信する。
【００８５】
Ｓ４１１で、ＣＰＵ１００は、ネイティブ層２１８で変換されたＢＡＳＥ６４データを受信し、それに従って、図３のＳ３０９で確保した描画領域に画像を描画する。この画像がＯＳ層２１９で解釈されることで、指定した表示領域内に画像として表示される。
【００８６】
＜スタンプ追加の詳細＞
図３のＳ２３のスタンプ追加の詳細について、図６を用いて説明する。ここでは、ユーザ操作によって、図１２のアプリケーション画面１２００のスタンプ追加ボタン１２０３が押下されてスタンプ一覧が表示された後、ハートスタンプ１２０８が選択された場合を例に挙げて説明する。尚、Ｓ５０１－Ｓ５０２、Ｓ５０８－Ｓ５１０は、ＣＰＵ１００がスクリプト層２１７のプログラムを用いて実行する処理であり、Ｓ５０３－Ｓ５０７はＣＰＵ１００がネイティブ層２１８のプログラムを用いて実行する処理である。
【００８７】
Ｓ５０１で、ＣＰＵ１００は、一意なＩＤを生成する。このＩＤは、図３のＳ３０１で生成するＩＤと同等の性質を有する。Ｓ５０２で、ＣＰＵ１００は、Ｓ５０１で生成されたＩＤとともに、スタンプとして利用される画像のアクセス先（絶対パス）をネイティブ層２１８に送信することで、スタンプに対応するスタンプ画像の画像選択を依頼する。
【００８８】
Ｓ５０３で、ＣＰＵ１００は、スクリプト層２１７から受信したスタンプ画像の絶対パスとデバイス固有の画像選択ＡＰＩを利用して、スタンプ画像を取得する。Ｓ５０４で、ＣＰＵ１００は、取得したスタンプ画像をＲＧＢ展開する。Ｓ５０５で、ＣＰＵ１００は、展開したＲＧＢ画像を、スクリプト層２１７から取得したＩＤと関連付けてデータ保持部２０４に保持する。関連付け方法は、図３のＳ３０６と同様である。Ｓ５０６で、ＣＰＵ１００は、展開したＲＧＢ画像をスクリプト層２１７でサポート可能な形式のデータに変換する。ここでの変換は、図３のＳ３０７と同様に、ＪＰＥＧ形式のデータに変換する

。Ｓ５０７で、ＣＰＵ１００は、ＪＰＥＧ形式のデータをＢＡＳＥ６４データに変換し、スクリプト層２１７へ送信する。

【００８９】

Ｓ５０８で、ＣＰＵ１００は、ネイティブ層２１８で変換されたＢＡＳＥ６４データを受信するとともに、そのＢＡＳＥ６４データを表示するための描画領域をＲＡＭ１０２に確保する。Ｓ５０９で、ＣＰＵ１００は、オブジェクトパラメータを生成し、初期化する。ここで、オブジェクトパラメータとは、図３のＳ２７のレンダリングの際、レンダリング後のスタンプの回転角度を決定するために用いられるパラメータを保持するオブジェクトである。ＪａｖａＳｃｒｉｐｔ（登録商標）で保持するオブジェクトパラメータの一例としては、下記のような形態が考えられる。

【００９０】

```
var ObjectParam = function(){
      this.theta = 0;
      this.posX = 0;
      this.posY = 0;
      this.width = 100;
      this.height = 100;
}
```

【００９１】

このオブジェクトパラメータは、ＯｂｊｅｃｔＰａｒａｍオブジェクトの中に、回転角度を示すｔｈｅｔａという変数名と、ｔｈｅｔａには０という値が格納されているということを表している。同様に、ｐｏｓＸは描画領域の左上を基準点とした時のｘ座標、ｐｏｓＹは描画領域の左上を基準点とした時のｙ座標、ｗｉｄｔｈは描画領域の横幅、ｈｅｉｇｈｔは描画領域の縦幅を表している。尚、本実施形態では、説明を簡単にするため、オブジェクトパラメータは必要最小限であるが、その他のパラメータ（平行移動量、拡大倍率等）を追加し、描画時やレンダリング時に利用できることは明らかである。

【００９２】

Ｓ５１０で、ＣＰＵ１００は、ＢＡＳＥ６４データを、生成したオブジェクトパラメータを基に、描画領域１２０６に画像として表示する。具体的には、ＣＰＵ１００は、選択されたスタンプに対応するＢＡＳＥ６４データをＯＳ層２１９のインタプリタ２１４に送信する。そして、インタプリタ２１４がＢＡＳＥ６４データのスクリプトを解釈し、描画領域にスタンプ画像として表示する。尚、本実施形態では、説明を簡単にするために、スタンプを１つ選択した場合を例に挙げているが、複数個のスタンプを選択できる。また、本実施形態では、スタンプに予め用意した画像を利用しているが、Ｃｏｎｔｅｘｔオブジェクトを利用して描画物をスクリプト層で生成する方法を用いても良い。

【００９３】

＜スタンプ特定の詳細＞

図３のＳ２４のスタンプ特定の詳細について、図７を用いて説明する。尚、Ｓ６０２－６０３は、ＣＰＵ１００がスクリプト層２１７のプログラムを用いて実行する処理であり、Ｓ６０１はＣＰＵ１００がネイティブ層２１８のプログラムを用いて実行する処理である。

【００９４】

Ｓ６０１で、ＣＰＵ１００は、ディスプレイ１０４上でタッチされた座標を取得し、スクリプト層２１７に送信する。

【００９５】

Ｓ６０２で、ＣＰＵ１００は、ネイティブ層２１８から受信した座標と、図５のＳ５０９で生成したオブジェクトパラメータの情報より、図３のＳ２３で追加したスタンプがタッチされたかどうかを判断する。追加したスタンプにおいて、オブジェクトパラメータは初期値のままである。そのため、上記のオブジェクトパラメータの一例に従えば、スタンプは、描画領域１２０６の左上を（０，０）とし、そこからｘ方向に１００、ｙ方向に１

００の領域に描画されていることになる。これより、受信した座標の座標（ｘ，ｙ）の内、」ｘ座標から描画領域１２０６のｘ座標分を差し引いた値が０～１００の範囲、かつ、ｙ座標から描画領域１２０６のｙ座標分を差し引いた値が０～１００の範囲であれば、スタンプはタッチされたと判断できる。スタンプがタッチされたと判断された場合、そのスタンプは、操作受付状態となる。以降の説明では、Ｓ２３で追加したスタンプがタッチされたものとして説明する。

【００９６】

Ｓ６０３で、ＣＰＵ１００は、判断結果に応じて、スタンプを操作受付状態に設定する。ここで、操作受付状態に設定することは、タッチされたスタンプのＩＤを注目スタンプＩＤとして、一時的にスクリプト層２１７に記録しておくことである。これは、データ保持部２０４には、画像とＩＤが対となって記憶されているため、ＩＤさえ把握しておけば一意な画像を特定できるためである。

【００９７】

＜スタンプ操作の詳細＞

図３のＳ２５のスタンプ操作の詳細について、図８を用いて説明する。尚、図８の各ステップは、ＣＰＵ１００がスクリプト層２１７のプログラムを用いて実行する処理である。

【００９８】

Ｓ７０１で、ＣＰＵ１００は、スタンプのオブジェクトパラメータのｒｏｔａｔｅの値を更新する。例えば、図１２のスライドバー１２０４で設定した値に更新する。Ｓ７０２で、ＣＰＵ１００は、オブジェクトパラメータを用いて、図７のＳ６０３で操作受付状態に設定されたスタンプを描画領域１２０６に再描画する。例えば、スタンプ画像の描画をＨＴＭＬのＣａｎｖａｓで行っている場合、ＣａｎｖａｓのＣｏｎｔｅｘｔオブジェクトのｒｏｔａｔｅメソッドを利用することで、Ｃａｎｖａｓ内の画像を回転することが可能である。

【００９９】

尚、本実施形態では、スタンプの操作は回転のみであるが、拡大・縮小、平行移動等の他の操作も考えられる。また、写真画像に対してオブジェクトパラメータを持たせれば、スタンプ操作と同様の操作が可能となることも、明らかである。

【０１００】

＜プリンタ設定の詳細＞

図３のＳ２６のプリンタ設定の詳細について、図９を用いて説明する。尚、Ｓ８０１、Ｓ８０７－Ｓ８０９は、ＣＰＵ１００がスクリプト層２１７のプログラムを用いて実行する処理であり、Ｓ８０２－Ｓ８０６、Ｓ８１０はＣＰＵ１００がネイティブ層２１８のプログラムを用いて実行する処理である。

【０１０１】

Ｓ８０１で、ＣＰＵ１００は、スクリプト層２１７からネイティブ層２１８へデバイス情報としてのプリンタ情報取得を依頼する。これは、プリンタ１１２と通信を行うためのスクリプト層２１７からの要求にあたる。依頼の方法は、画像選択時と同様に、バインディング機能によりスクリプトからネイティブ固有のＡＰＩを呼び出す。ネイティブ層２１８には、スクリプト層２１７から直接呼び出せる関数もしくはその関数を間接的に呼び出す、いわゆる、ラッパーが予め用意してある。例えば、ＧｅｔＰｒｉｎｔｅｒＩｎｆｏというネイティブ関数を用意しておき、スクリプト側からその関数を呼び出す。このように、ネイティブ層はスクリプトからの外部デバイスとの通信の要求を取得する。

【０１０２】

通常、スクリプト層２１７からはセキュリティ上の制限で外部デバイスと直接通信することはできない。そのため、上記のように、スクリプト層２１７から、一旦、ネイティブ層２１８へ外部デバイス情報の取得を依頼し、ネイティブ層２１８を介して外部デバイスと通信を行う。ここで、ネイティブ層２１８は、ＯＳ層２１９を介して、外部デバイス（例えば、プリンタ１１２）と通信する機能を備えている。

【０１０３】
Ｓ８０２で、ＣＰＵ１００は、スクリプト層２１７からの依頼に従って、プリンタ情報取得のための関数を呼び出し、プリンタの検知、いわゆる、ディスカバリを行う。この検知には、例として、同一の無線ＬＡＮルータで接続しているプリンタを検知するという方法がある。ここでは、通信可能なプリンタの検知を行うため、例えば、Ｂｏｎｊｏｕｒ等のプロトコルにより、ブロードキャストやマルチキャスト等の方法で応答の要求を行い、プリンタからの応答を待機することでプリンタを検知する。
【０１０４】
Ｓ８０３で、ＣＰＵ１００は、プリンタ１１２からの応答に含まれるＩＰアドレスを記憶する。Ｓ８０４で、ＣＰＵ１００は、記憶したＩＰアドレスに対応するプリンタ１１２へ、デバイス情報取得コマンドとしてプリンタ情報取得コマンドを送信する。ここで、応答したプリンタ１１２が複数である場合、ＣＰＵ１００は、全てのプリンタ１１２に対して、プリンタ情報取得コマンドを送信する。プリンタ情報取得コマンドとは、プリンタの動作を指定する命令であり、その一例としては、以下のようなＸＭＬで表現される。
【０１０５】

```
01:      <?xml version="1.0" encoding="utf-8" ?>
02:      <cmd xmlns:trans="http://www.xxxx/yyyyy/">
03:           <contents>
04:                <operation>GetCapability</operation>
05:           </contents>
06:      </cmd>
```

【０１０６】
上記各行の左側に書かれている「０１：」等の数値は説明を行うために付加した行番号であり、本来のＸＭＬ形式のテキストには記載されない。
【０１０７】
１行目はヘッダであり、ＸＭＬ形式で記述していることを表している。
【０１０８】
２行目のｃｍｄはコマンドの開始を意味する。ｘｍｌｎｓで名前空間を指定し、コマンドの解釈の定義を指定している。尚、６行目の＜／ｃｍｄ＞でコマンドの終了を示している。
【０１０９】
３行目は以後に内容を記載する宣言であり、５行目でその終了を示している。
【０１１０】
４行目には要求する命令が記載されており、＜ｏｐｅｒａｔｉｏｎ＞と＜／ｏｐｅｒａｔｉｏｎ＞の間に実際の命令文言が存在する。命令文言であるＧｅｔＣａｐａｂｉｌｉｔｙは、外部デバイスであるプリンタの情報を取得する命令である。例えば、プリンタが対応している用紙種類、サイズ、縁なし印刷機能の有無、印刷品位、等のケーパビリティ情報を提供するよう要求する。
【０１１１】
尚、プリンタ情報取得コマンドの生成は、例えば、ＲＯＭ１０１に予め記憶した固定のテキストを読み込んでも構わない。また、ＸＭＬ等のテキスト形式に限らず、バイナリ形式で記述し、それに沿ったプロトコルで通信しても良い。また、生成したプリンタ情報取得コマンドは、ＨＴＴＰ等のプリンタが対応している通信プロトコルに従った形式で、プリンタ通信部２１３を介してプリンタ１１２へ送信される。
【０１１２】
また、通信方法はこれに限定されるものではない。Ｗｉ－Ｆｉ（登録商標）ダイレクトやＢｌｕｅｔｏｏｔｈ（登録商標）、赤外線通信、電話回線、有線ＬＡＮ、ＵＳＢを用いた接続でも良く、その方法に沿ったプロトコルで通信を行えば同様の効果を得ることがで

きる。

【０１１３】

図９では、ネイティブ層２１８で、プリンタ情報取得コマンドを生成する構成としているが、スクリプト層２１７で、プリンタ情報取得コマンドを生成する構成でも、同様の効果が得られる。その場合、スクリプト層２１７で、例えば、上記のＸＭＬ形式の命令文を含むプリンタ情報取得コマンドを作成し、ネイティブ層２１８へ渡す。それを受けて、ネイティブ層２１８は、通信プロトコルに従った形式でプリンタ１１２へプリンタ情報取得コマンドを送信する。

【０１１４】

プリンタ１１２は、情報処理装置１１５からプリンタ情報取得コマンドを受信すると、デバイス情報であるプリンタ情報をＸＭＬ形式で通信プロトコルに沿って、情報処理装置１１５へ送信する。以下に、プリンタの情報の一例を示す。

【０１１５】

```
-------------------------------------------
01:      <?xml version="1.0" encoding="utf-8" ?>
02:      <cmd xmlns:trans="http://www.xxxx/yyyyy/">
03:           <contents>
04:                <device id=”Printer001” />
05:                <mode = 1>
06:                     <media>GlossyPaper</media>
07:                     <size>A4</size>
08:                     <quality>1</quality>
09:                     <border>no</border>
10:                     <src>sRGB.icc</src>
11:                     <dst>001GlossyPaper.icc</dst>
12:                     <intent>perceptual</intent>
13:                </mode>
14:                <mode = 2>
                        ～中略～
                   </mode>
                   <mode = 3>
                        ～中略～
                   </mode>

                   ～中略～

              </contents>
         </cmd>
-------------------------------------------
```

【０１１６】

１行目はヘッダであり、ＸＭＬ形式で記述していることを表している。

【０１１７】

２行目のｃｍｄはコマンドの開始を意味する。ｘｍｌｎｓで名前空間を指定し、コマンドの解釈の定義を指定している。尚、最下行の＜／ｃｍｄ＞でコマンドの終了を示している。

【０１１８】

３行目は以後に内容を記載する宣言であり、下の＜／ｃｏｎｔｅｎｔｓ＞までその内容は継続する。

【０１１９】

４行目でデバイスＩＤを示している。ここでは、プリンタ１１２の機種名が「Ｐｒｉｎ

ｔｅｒ００１」であることを表している。
【０１２０】
５行目以降はプリンタ１１２が有する各モードについての記述である。＜ｍｏｄｅ＞から＜／ｍｏｄｅ＞までで、１つのモードにおける情報が記述されている。５行目では、モードの番号が１である。以降の＜ｍｅｄｉａ＞は印刷用紙の種類、＜ｓｉｚｅ＞は用紙サイズ、＜ｑｕａｌｉｔｙ＞は印刷品位、＜ｂｏｒｄｅｒ＞は縁あり／なしの情報をそれぞれ記述している。
【０１２１】
１０行目以降は、＜ｓｒｃ＞はソースプロファイル名、＜ｄｓｔ＞はディスティネーションプロファイル名、＜ｉｎｔｅｎｔ＞はマッチング方法の情報をそれぞれ記載してある。
【０１２２】
１４行目以降は他のモードであるモード２についての情報が記述されている。このように、プリンタ１１２の機種名と、そのプリンタが対応している全てのモードがこのＸＭＬに記述されている。
【０１２３】
尚、プリンタ情報の記述方法はこれに限定されることはなく、タグ形式でないテキストや、バイナリ形式等の他の形式であっても構わない。
【０１２４】
Ｓ８０５で、ＣＰＵ１００は、プリンタ１１２からプリンタ情報を受信し、そのプリンタ情報からプリンタ１１２の機能一覧を取得する。例えば、プリンタ１１２が有する全てのモードにおける印刷用紙の種類、サイズ、印刷品位、縁あり／なしの項目と項目数等を含むプリンタ機能一覧が取得される。更に、本実施形態のプリンタ情報として、対応するソースプロファイル、ディスディネーションプロファイル、マッチング方法の一覧を取得する。
【０１２５】
Ｓ８０６で、ＣＰＵ１００は、取得したプリンタ機能の一覧に関するプリンタ情報をスクリプト層２１７が解釈できる形式に変換して、スクリプト層２１７へ送信する。つまり、プリンタ１１２との通信によって得られた情報をスクリプト層２１７へ渡す。具体的には、バインディング機能を用いてネイティブ層２１７からスクリプト層２１８にプリンタ機能の一覧に関するプリント情報が送信される。スクリプト層２１７からそのネイティブ関数を呼び出し、戻り値として情報を受け渡す。取得するモード等を引数とし、スクリプト層２１７はそのモードの戻り値を受け取るようにしても良い。その他の例として、受け取ったＸＭＬ形式のプリンタ情報で送信したり、タグなしのテキスト形式に変えて送信する等の方法がある。加えて、上述のＪＳＯＮ文字列を用いた受け渡しや、データ変換部２０７及び２０３を用いてＢＡＳＥ６４等の文字列で受け渡しを行ってもよい。
【０１２６】
Ｓ８０７で、ＣＰＵ１００は、ネイティブ層２１８から受信したプリンタ情報に基づいて、プリンタ１１２で利用できる機能を含む設定画面（図１３）を形成し、表示する。これを、本実施形態では、表示制御と呼ぶ。本実施形態では、最初にプリンタを選択している。しかし、接続可能なプリンタが複数ある場合は、ＣＰＵ１００が、このタイミングでプリンタ名を表示し、ユーザに印刷するプリンタを選択させるための表示画面を生成する（表示内容を制御する）。尚、プリンタの選択は、上記に限らず、一番早く応答してきたプリンタや、より機能が多いプリンタ、印刷ジョブが混んでいないプリンタを選択する、等の方法も考えられる。
【０１２７】
このように、ＣＰＵ１００は、印刷用紙の種類・サイズ、印刷品位、縁あり／なし等のプリンタで利用できる機能を選択させる設定画面１３０１（図１３）を表示する。設定画面の形成方法の一例として、以下に、ＨＴＭＬ記述のサンプルを示す。
【０１２８】

```
------------------------------------------------
<!DOCTYPE html>
    <head>
    <title>印刷設定 </title>
    <script>
        <!-- 用紙サイズ -->
        var PaperSizeNum = GetPaperSizeNum();
        var p = document.getElementById("PaperList");
        var i;
        for(i=0; i<PaperSizeNum; i++){
              p.options[i] = new Option(GetPaperSizeT(i), GetPaperSizeV(i));
        }

        <!-- 用紙種類-->
        var MediaTypeNum = GetMediaTypeNum();
        var m = document.getElementById("MediaList");
        var j;
        for(j=0; j<MediaTypeNum; j++){
              m.options[i] = new Option(GetMediaTypeT(j), GetMediaTypeV(j));
        }

        <!-- 印刷品位 -->
        var QualityNum = GetQualityNum();
        var q = document.getElementById("QualityList");
        var k;
        for(k=0; k< QualityNum; k++){
              q.options[i] = new Option(GetQualityT(k), GetQualityV(k));
        }

        <!-- 縁あり/なし-->
        var BorderNum = GetBorderNum();
        var b = document.getElementById("BorderList");
        var l;
        for(l=0; l<BorderNum; l++){
              b.options[i] = new Option(GetBorderT(l), GetBorderV(l));
        }

        <!-- 印刷関数-->
        function printer() {
              SetPrint(document.getElementById("PaperList").value,
                    document.getElementById("MediaList").value,
                    document.getElementById("QualityList").value,
                    document.getElementById("BorderList").value);
        }
    </script>
    </head>

        <!-- 表示部 -->
        <body>
        用紙サイズ  <select id="PaperList"></select><br />
```

```
用紙種類　　＜select id="MediaList"＞＜/select＞＜br /＞
印刷品位　　＜select id="QualityList"＞＜/select＞＜br /＞
縁あり/なし ＜select id="BorderList"＞＜/select＞＜br /＞
＜br /＞

＜button id="btn1" onclick="printer()"＞設定完了＜/button＞
＜/body＞
＜/html＞
```

【０１２９】

上記のＧｅｔＰａｐｅｒＳｉｚｅＮｕｍ（　）、ＧｅｔＭｅｄｉａＴｙｐｅＮｕｍ（　）、ＧｅｔＱｕａｌｉｔｙＮｕｍ（　）、ＧｅｔＢｏｒｄｅｒＮｕｍ（　）はネイティブ関数であり、それぞれの項目数を取得する機能を備える。例えば、プリンタが対応している用紙サイズがＡ４、Ａ５、Ｂ５、Ｌ判の４種類である場合、ＧｅｔＰａｐｅｒＳｉｚｅＮｕｍ（　）は４を返す。

【０１３０】

ＧｅｔＰａｐｅｒＳｉｚｅＴ（ｎ）、ＧｅｔＭｅｄｉａＴｙｐｅＴ（ｎ）、ＧｅｔＱｕａｌｉｔｙＴ（ｎ）、ＧｅｔＢｏｒｄｅｒＴ（ｎ）もネイティブ関数であり、引数ｎの値番目の文字列を返す。例えば、用紙サイズのテキストを返す関数のＧｅｔＰａｐｅｒＳｉｚｅ（０）の返り値は「Ａ４」、ＧｅｔＰａｐｅｒＳｉｚｅ（１）の返り値は「Ａ５」となる。これらの値は、プリンタから受信するプリンタ情報からネイティブ関数が取り出す。

【０１３１】

ＧｅｔＰａｐｅｒＳｉｚｅＶ（ｎ）、ＧｅｔＭｅｄｉａＴｙｐｅＶ（ｎ）、ＧｅｔＱｕａｌｉｔｙＶ（ｎ）、ＧｅｔＢｏｒｄｅｒＶ（ｎ）もネイティブ関数であり、引数ｎの値に対応する値を返す。例えば、用紙種類のテキストを返す関数のＧｅｔＭｅｄｉａＴｙｐｅＴ（０）の返り値は「光沢紙」のように、表示してユーザに示す文言である。一方、ＧｅｔＭｅｄｉａＴｙｐｅＶ（０）の返り値は「ＧｌｏｓｓｙＰａｐｅｒ」と、プリンタが解釈できる文言となっている。これらの文言は、プリンタ情報と結び付けてネイティブ関数が決定する。例えば、プリンタ情報より取り出した値が「ＧｌｏｓｓｙＰａｐｅｒ」である場合、表示するテキストは「光沢紙」と決定する。決定方法として、ネイティブ関数はこれらの対応表を予め保持しておき、その対応表に沿ってテキストを決定すれば良い。

【０１３２】

尚、上記では、例として、用紙サイズ、用紙種類、印刷品位、縁あり／なしの設定を行う仕様であるが、これに限定されるものではない。他の例として、両面／片面、カラー／モノクロ、画像補正のオン／オフ等の他の設定項目が挙げられる。また、印刷機能のみではなく、プリンタが処理可能な画像処理や解析処理、サイレントモードの有無、メモリカードの利用有無、インク残量等のステータス等の情報を表示しても良い。

【０１３３】

Ｓ８０８で、ＣＰＵ１００は、設定画面１３０１に対するユーザ操作に基づいて、プリンタに設定する機能を選択する。上記の例で示したＨＴＭＬ記述を、レンダリング部２１６を用いてレンダリングしてディスプレイ１０４に表示した例が、図１３の設定画面１３０１である。ネイティブ層２１８を介してプリンタ情報を要求し、プリンタ情報から、上記のネイティブ関数を用いて取得した情報を基に設定画面１３０１が形成されている。

【０１３４】

尚、上記では、ＨＴＭＬ記述はスクリプト層２１７で形成する構成で説明しているが、これに限定されるものではない。例えば、ネイティブ層２１８でＨＴＭＬ記述を形成し、スクリプト層２１７でそれをレンダリングしてディスプレイ１０４に表示するようにしても良い。

【０１３５】

また、設定画面１３０１の用紙サイズ等の設定項目はそれぞれプルダウンメニューになっており、ユーザ操作によって各設定項目を選択することができる。ここで、設定画面１３０１は、プルダウンメニューによって、用紙サイズの設定項目として選択可能な項目の一覧が表示されている状態を示しており、ユーザ操作によってＡ４やＡ５等の用紙サイズを選択することができる。

【０１３６】

設定完了ボタン１３０２が押下されると、ソースプロファイル、ディスティネーションプロファイル、マッチング方法を選択させるカラー設定用の設定画面を表示する。カラー設定用の設定画面の形成方法の一例として、以下にＨＴＭＬのサンプルを示す。

【０１３７】

```
-----------------------------------------------
<!DOCTYPE html>
    <head>
    <title>カラー設定 </title>
    <script>
        <!-- ソースプロファイル -->
        var SourceProfileNum = GetSourceProfileNum();
        var p = document.getElementById("SourceProfileList");
        var i;
        for(i=0; i<SourceProfileNum; i++){
              p.options[i] = new Option(GetSourceProfileT(i), GetSourceProfile
V(i));
        }

        <!-- ディスティネーションプロファイル-->
        var DestinationProfileNum = GetDestinationProfileNum();
        var m = document.getElementById("DestinationProfileList");
        var j;
        for(j=0; j<DestinationProfileNum; j++){
              m.options[i] = new Option(GetDestinationProfileT(j),GetDestinati
onProfileV(j));
        }

        <!-- マッチング方法 -->
        var IntentNum = GetIntentNum();
        var q = document.getElementById("IntentList");
        var k
        for(k=0; k< IntentNum; k++){
              q.options[i] = new Option(GetIntentT(k), GetIntentV(k));
        }

        <!-- カラーマッチング関数-->
        function printer() {
              SetColor(document.getElementById("SourceProfileList").value,
                  document.getElementById("DestinationProfileList").value,
                  document.getElementById("IntentList ").value);
        }
    </script>
    </head>
        <!-- 表示部 -->
```

```
        <body>
        ソースプロファイル　<select id="SourceProfileList"></select><br />
        ディスティネーションプロファイル　　<select id="DestinationProfileList"></select><br />
        マッチング方法　　<select id="IntentList"></select><br />
        <br />
    <button id="btn1" onclick="printer()">設定完了</button>
        </body>
</html>
```

【０１３８】

上記のＧｅｔＳｏｕｒｃｅＰｒｏｆｉｌｅＮｕｍ（　）、ＧｅｔＤｅｓｔｉｎａｔｉｏｎＰｒｏｆｉｌｅＮｕｍ（　）、ＧｅｔＩｎｔｅｎｔＮｕｍ（　）はネイティブ関数であり、それぞれの項目がいくつあるかを得る機能を備える。例えば、プリンタが対応しているソースプロファイルがｓＲＧＢ、ＡｄｏｂｅＲＧＢ、ＰｒｏＰｈｏｔｏＲＧＢの３種類ある場合、ＧｅｔＳｏｕｒｃｅＰｒｏｆｉｌｅＮｕｍ（　）は３を返す。

【０１３９】

ＧｅｔＳｏｕｒｃｅＰｒｏｆｉｌｅＴ（ｎ）、ＧｅｔＤｅｓｔｉｎａｔｉｏｎＰｒｏｆｉｌｅＴ（ｎ）、ＧｅｔＩｎｔｅｎｔＴ（ｎ）もネイティブ関数であり、引数ｎの値番目の文字列を返す。例えば、ソースプロファイルのテキストを返す関数のＧｅｔＳｏｕｒｃｅＰｒｏｆｉｌｅＴ（０）の返り値は「ｓＲＧＢ」、ＧｅｔＳｏｕｒｃｅＰｒｏｆｉｌｅＴ（１）の返り値は「ＡｄｏｂｅＲＧＢ」となる。これらの値は、プリンタから受信するプリンタ情報からネイティブ関数が取り出す。

【０１４０】

ＧｅｔＳｏｕｒｃｅＰｒｏｆｉｌｅＶ（ｎ）、ＧｅｔＤｅｓｔｉｎａｔｉｏｎＰｒｏｆｉｌｅＶ（ｎ）、ＧｅｔＩｎｔｅｎｔＶ（ｎ）もネイティブ関数であり、引数ｎの値に対応する値を返す。例えば、用紙種類のテキストを返す関数のＧｅｔＳｏｕｒｃｅＰｒｏｆｉｌｅＴ（０）の返り値は「ｓＲＧＢ」のように、表示してユーザに示す文言である。一方、ＧｅｔＳｏｕｒｃｅＰｒｏｆｉｌｅＶ（０）の返り値は「ｓＲＧＢ．ｉｃｃ」と、プリンタが解釈できる文言となっている。これらの文言は、プリンタ情報と結び付けてネイティブ関数が決定する。

【０１４１】

上記の例で示したＨＴＭＬ記述を、レンダリング部２１６を用いてレンダリングしてディスプレイ１０４に表示した例が、図１４のカラー設定用の設定画面１４０１である。ネイティブ層２１８を介してプリンタ情報を要求し、上記のネイティブ関数を用いて取得した情報を基に設定画面１４０１が形成される。

【０１４２】

尚、上記では、ＨＴＭＬ記述はスクリプト層２１７で形成する構成で説明しているが、これに限定されるものではない。例えば、ネイティブ層２１８でＨＴＭＬ記述を形成し、スクリプト層２１７でそれをレンダリングしてディスプレイ１０４に表示するようにしても良い。

【０１４３】

また、設定画面１４０１のソースプロファイル等の設定項目はそれぞれプルダウンメニューになっており、ユーザ操作によって各設定項目を選択することができる。ここで、設定画面１４０１は、プルダウンメニューによって、ソースプロファイルの設定項目として選択可能な項目の一覧が表示されている状態を示しており、ユーザ操作によってｓＲＧＢやＡｄｏｂｅＲＧＢ等のカラープロファイルを選択することができる。

【０１４４】

Ｓ８０９で、ＣＰＵ１００は、設定完了ボタン１４０２に対するユーザ操作を検知する

と、ユーザ操作によって選択された設定項目を含む設定情報を作成して、ネイティブ層２１８へ送信する。上記ＨＴＭＬ記述の例にあるＳｅｔＰｒｉｎｔ（　）もバインディング機能を有するネイティブ関数である。上記の例では、ＳｅｔＰｒｉｎｔ（　）を用いて、用紙サイズ、用紙種類、印刷品位、縁あり／なしの印刷設定を文字列としてネイティブ層２１８へ渡している。また、ＳｅｔＣｏｌｏｒ（）を用いて、ソースプロファイル、ディスディネーションプロファイル、マッチング方法のカラー設定を文字列としてネイティブ層２１８へ渡している。

【０１４５】

Ｓ８１０で、ＣＰＵ１００は、バインディング機能によりスクリプト層２１７から設定情報を受信する。ネイティブ層２１８では、後に、受信した設定情報、印刷対象の画像データ、さらにはスタンプの画像データを基に、プリンタ１１２の通信プロトコルに従って、プリントコマンドを生成する。そして、プリンタコマンドは、プリンタ通信部２１３を介してプリンタ１１２へ送信される。

【０１４６】

以上のように、ＣＰＵ１００は、スクリプト層２１７からの要求によりネイティブ層２１８を介してプリンタ１１２から情報を取得する通信制御を行う。これにより、ＣＰＵ１００は、プリンタ１１２の性能を取得して、スクリプト層２１７で表示されるＵＩを制御することができる。

【０１４７】

＜レンダリングの詳細＞

図３のＳ２７のレンダリングの詳細について、図１０を用いて説明する。Ｓ９０１、Ｓ９０３、Ｓ９０５、Ｓ９０８、Ｓ９１２、Ｓ９１４は、ＣＰＵ１００がスクリプト層２１７のプログラムを実行することで実現される。Ｓ９０２、Ｓ９０４、Ｓ９０６、Ｓ９０７、Ｓ９１０、Ｓ９１１、Ｓ９１３、Ｓ９１８は、ＣＰＵ１００がＯＳ層２１９のプログラムを実行することで実現される。Ｓ９１５－Ｓ９１７は、ＣＰＵ１００がネイティブ層２１８のプログラムを実行することで実現される。

【０１４８】

Ｓ９０１で、ＣＰＵ１００は、インジケータの起動をＯＳ層２１９に依頼する。

【０１４９】

Ｓ９０２で、ＣＰＵ１００は、依頼によって起動したインジケータをディスプレイ１０４に表示する。

【０１５０】

Ｓ９０３で、ＣＰＵ１００は、Ｓ８０９で作成した設定情報で設定されている用紙サイズに対応する出力サイズを決定し、出力画像の描画領域を算出する。

【０１５１】

Ｓ９０４で、ＣＰＵ１００は、算出した出力画像の描画領域をＲＡＭ１０２に確保する。

【０１５２】

Ｓ９０５で、ＣＰＵ１００は、ＯＳ層２１９に写真画像の描画を依頼する。

【０１５３】

Ｓ９０６で、ＣＰＵ１００は、写真画像の描画の依頼のスクリプトを解釈し、写真画像をレンダリングし、Ｓ９０４で確保した描画領域に対し、レンダリング後の写真画像を反映する（Ｓ９０７）。ここでの反映とは、描画領域に変更を加えることであり、ディスプレイ１０４上に画像を表示することではない。

【０１５４】

Ｓ９０８で、ＣＰＵ１００は、スタンプ画像の描画を依頼する前に、オブジェクトパラメータによるレンダリング条件の変更をスクリプトで記述して設定する。Ｓ９０９で、ＣＰＵ１００は、ＯＳ層２１９にスタンプ画像の描画を依頼する。

【０１５５】

Ｓ９１０で、ＣＰＵ１００は、レンダリング条件に従って、スタンプ画像をレンダリン

グする。つまり、画像の回転処理やリサイズは、ＯＳ層で実行されている。レンダリング終了後、Ｓ９１１で、Ｓ９０４で確保した描画領域に対し、レンダリング後のスタンプ画像を反映する。

【０１５６】

Ｓ９１２で、ＣＰＵ１００は、ＯＳ層２１９に出力画像データの取得を依頼する。

【０１５７】

Ｓ９１３で、ＣＰＵ１００は、描画領域に存在する画像データを、ＢＡＳＥ６４データに変換して、スクリプト層２１７へ送信する。

【０１５８】

Ｓ９１４で、ＣＰＵ１００は、ネイティブ層２１８に対し、ＯＳ層２１９から受信したＢＡＳＥ６４データを送信して、印刷を依頼する。

【０１５９】

Ｓ９１５で、ＣＰＵ１００は、スクリプト層２１７から受信したＢＡＳＥ６４データをデコードする。Ｓ９１６で、ＣＰＵ１００は、ＢＡＳＥ６４データをＲＧＢ画像に変換する。Ｓ９１７で、ＣＰＵ１００は、インジケータの停止をＯＳ層２１９へ依頼する。

【０１６０】

Ｓ９１８で、ＣＰＵ１００は、インジケータを停止して、ディスプレイ１０４の表示から消去する。

【０１６１】

尚、Ｓ９１５において、ＢＡＳＥ６４データを受信したネイティブ層２１８は、レンダリングされた画像データに対し、設定画面１４０１で設定されたカラー設定に基づく色変換データを用いて、画像処理部２０８で色変換を施す。これにより、ディスプレイ１０４に表示されている画像に対応する画像データを、プリンタ１１２でプリントする場合に、プリンタ１１２の色空間特性に適した色変換が施された画像データを生成することができる。

【０１６２】

本実施形態では、色変換データとなる、ＩＣＣプロファイルと呼ばれる個々のデバイスの色空間特性を記録した定義ファイルを用いて、ＰＣＳと呼ばれる機器に依存しない色空間を介して、デバイス間の色の違いを調整する。まず、ソースプロファイルからＲＧＢをＣＩＥＬ＊ａ＊ｂ＊等のデバイスに非依存な色空間に変換する。次に、変換先となるディスティネーションプロファイルからＣＩＥＬ’＊ａ’＊ｂ’＊に変換する。マッチング方法に応じて、それぞれの色再現範囲のミスマッチをマッピングする色空間変換を実行する。最後に、ＣＩＥＬ’＊ａ’＊ｂ‘＊を変換先の画像データを示すＲ’Ｇ’Ｂ’へ変換する。

【０１６３】

このとき使用する色変換方法は、ＩＣＣプロファイルによる方法であるが、他の方法でもよい。例えば、ＲＧＢからＲ’Ｇ’Ｂ’に変換するルックアップテーブルによる方法でも、行列演算による方法でもよく、その場合、カラー設定に対応する色変換データがデータ保持部２０４に保持されているものとする。また、画像に指定された（含まれる）プロファイル（色変換データ）とユーザの指定プロファイル（色変換データ）が異なる場合には、画像のプロファイルからユーザ指定のプロファイルへの色変換を行ってから、同様の色変換を行うこととする。

【０１６４】

＜プリントの詳細＞

図３のＳ２８のプリントの詳細について、図１１を用いて説明する。尚、Ｓ１００１－Ｓ１００３は、ＣＰＵ１００がネイティブ層２１８のプログラムを実行することで実現される。

【０１６５】

Ｓ１００１で、ＣＰＵ１００は、Ｓ８０９で作成した設定情報、Ｓ９１６で変換したＲＧＢ画像をプリンタ１１２で利用可能な形式に変換する。プリンタが利用可能な形式は、

ＲＧＢやＪＰＥＧ、ＣＭＹＫ、ＰＤＦ等のプリンタベンダーの独自フォーマット等の画像データである。Ｓ１００２で、ＣＰＵ１００は、Ｓ１００１の変換結果を用いてプリンタ１１２へ送信するコマンドを生成する。Ｓ１００３で、ＣＰＵ１００は、プリンタの利用できる通信プロトコルに則り、Ｓ１００２で生成したコマンドを、印刷用に選択されたプリンタ１１２に対し、プリンタ通信部２１３を利用して送信する。

【０１６６】

Ｓ１００４で、プリンタ１１２は、情報処理装置１１５から受信したコマンドに従って、プリントを実行する。

【０１６７】

以上説明したように、本実施形態によれば、ハイブリッドアプリケーションを利用する環境において、色空間特性が異なるデバイス間で画像データを入出力する場合でも、各デバイスの色空間特性に適した色変換を行うことができる。また、スクリプト層とネイティブ層で、色変換を伴う画像データのやり取りを実現することができる。多くのＯＳはＪａｖａＳｃｒｉｐｔを解釈できる機能を標準的に備えているので、ネイティブ言語の違いを意識せずとも、一度の開発で多くのＯＳのユーザに対してのアプリケーション提供が可能となる。それに加え、実行速度で優位であるネイティブ層を利用することが可能となり、ユーザビリティが向上する。また、ネイティブ層を外部サーバとの通信を行わずに実現できるため、オフラインでの動作も可能となる。

【０１６８】

＜実施形態２＞

実施形態１では、プリンタ１１２で画像を印刷するための色変換を行う構成について説明しているが、ディスプレイ１０４で画像を表示するための色変換を行う場合にも適用できる。これは、例えば、ディスプレイ１０４の色空間特性とは異なる色空間特性の外部デバイスから入力された画像データを、ディスプレイ１０４で出力（表示）する場合に有効である。

【０１６９】

実施形態２では、実施形態１の図３の処理の内、特に、実施形態１のＳ２２の画像処理の詳細、つまり、図５のフローチャートの処理の内、Ｓ４０６の処理が実施形態１と異なる。

【０１７０】

実施形態２では、Ｓ４０６において、ＣＰＵ１００は、ネイティブ層２１８で、取得した補正パラメータから施すべき画像処理を判断し、ＲＧＢ画像に対して補正パラメータに対応する画像処理を行う。ここで、予め保存されたディスプレイ１０４に対応するモニタプロファイルを、ディスティネーションプロファイルとして参照することで色変換を行う。このとき使用する色変換方法は、実施形態１に図１０のＳ９１５で説明する同様の色変換方法であるため、その詳細については省略する。尚、実施形態２では、画像処理が適用された後の画像データがＳ４１０においてＢＡＳＥ６４データに変換されたスクリプト層に送信される。その結果、スクリプト層２１７では、ネイティブ層２１８の画像処理の結果が反映された画像データを表示することが可能となる。

【０１７１】

また、スクリプト層２１７で、モニタ設定用の設定画面を表示して、ユーザに、所望のプロファイルを選択させることもできる。実施形態１の例で示したＨＴＭＬ記述と同様に、レンダリング部２１６を用いてレンダリングしてディスプレイ１０４に表示した例が、図１５のモニタ設定用の設定画面１５０１である。スクリプト層２１７で、設定完了ボタン１５０２を含む設定画面１５０１を提供することで、ネイティブ層２１８を介し、ネイティブ関数を用いて取得した情報を基にカラー設定用の設定画面１５０１が形成される。ここで、選択されたモニタプロファイルを、ディスティネーションプロファイルとして参照することで色変換を行う。設定画面１５０１の形成方法の一例として、以下にＨＴＭＬのサンプルを示す。

【０１７２】

```
---------------------------------------------
<!DOCTYPE html>
    <head>
    <title>モニタ設定 </title>
    <script>
        <!-- ソースプロファイル -->
        var SourceProfileNum = GetSourceProfileNum();
        var p = document.getElementById("SourceProfileList");
        var i;
        for(i=0; i<SourceProfileNum; i++){
            p.options[i] = new Option(GetSourceProfileT(i), GetSourceProfile
V(i));
        }

        <!-- ディスティネーションプロファイル-->
        var DestinationProfileNum = GetDestinationProfileNum();
        var m = document.getElementById("DestinationProfileList");
        var j;
        for(j=0; j<DestinationProfileNum; j++){
            m.options[i] = new Option(GetDestinationProfileT(j),GetDestinati
onProfileV(j));
        }

        <!-- マッチング方法 -->
        var IntentNum = GetIntentNum();
        var q = document.getElementById("IntentList");
        var k
        for(k=0; k< IntentNum; k++){
            q.options[i] = new Option(GetIntentT(k), GetIntentV(k));
        }
        <!-- カラーマッチング関数-->
        function printer() {
            SetColor(document.getElementById("SourceProfileList").value,
                document.getElementById("DestinationProfileList").value,
                document.getElementById("IntentList ").value);
        }
    </script>
    </head>
        <!-- 表示部 -->
        <body>
        ソースプロファイル　<select id="SourceProfileList"></select><br /
>
        ディスティネーションプロファイル　　<select id="DestinationProfileLis
t"></select><br />
        マッチング方法　　<select id="IntentList"></select><br />
        <br />
    <button id="btn1" onclick="monitor ()">設定完了</button>
        </body>
</html>
---------------------------------------------
```

【０１７３】
以上説明したように、実施形態２によれば、ディスプレイに対する色変換においても、実施形態１と同様の効果を得ることができる。
【０１７４】
＜実施形態３＞
上記実施形態の色変換を適用するデバイスは、プリンタあるいはディスプレイとしているが、これらに限定されない。例えば、プロジェクタ、スキャナ、カメラ等の、色空間特性が異なるデバイス（入力デバイスと出力デバイス）間で画像データを入力／出力する場合にも適用することが可能である。
【０１７５】
また、上記実施形態では、処理対象の画像データを出力（プリンタ１１２へ印刷、あるいはディスプレイ１０４へ表示）する場合に、その出力先のデバイスの色空間特性に適した色変換を行う構成としているが、これに限定されない。
【０１７６】
例えば、プリンタ１１２へ画像データを出力する前に、ディスプレイ１０４で、その画像データによってプリンタ１１２で印刷される画像のプレビューを表示する場合にも、上記実施形態の色変換を行うようにしても良い。
【０１７７】
また、処理対象の画像データをスキャナから入力して、ディスプレイ１０４でその画像データ（あるいはプレスキャンによって入力される画像データ）を表示する場合にも、上記実施形態の色変換を行うようにしても良い。
【０１７８】
更には、スキャナとプリンタを組み合わせて、あるいはスキャナ機能とプリンタ機能を有する複合機による画像のコピーを行う場合にも、上記実施形態の色変換を行うようにしても良い。この場合、例えば、スキャナ機能によって入力された画像データをプレビューする時点、プリンタ機能よって印刷する前の画像データをプレビューする時点の少なくとも一方の時点に色変換を行うこともできる。また、プレビューを介在させない場合には、最終的な出力先のデバイスに対する色変換だけを行うこともできる。
【０１７９】
上記実施形態の各実施形態で示す色変換データは、ネイティブ層２１８のデータ保持部２０４に予め保持されている例を示しているが、これに限定されるものではない。例えば、色変換データは、図１のプリンタ１１２やサーバ１１４に保存されていることも考えられる。その場合、ネイティブ層２１８のデータ保存部２２０を介して、アプリケーションが起動した際に、インターネットの標準的通信機能を用いて、色変換データを取得する構成とすることや、ＯＳ内部に予め包含させる構成とすることも可能である。更に、色変換データの取得先が複数個ある場合には、それらの色変換データに対して優先度を付与しておき、優先度の高い色変換データのみを使用することもできるし、各色変換データを用いて適宜色変換を行うこともできる。
【０１８０】
上記実施形態では、情報処理装置１１５として、携帯型情報端末上でハイブリッド型アプリケーションを動作させる場合を例に挙げて説明しているが、これに限定されるものではない。例えば、ハイブリッド型アプリケーションが動作する環境は、スマートフォンやタブレットＰＣを始めとする情報端末の他に、ＰＣやサーバ、ゲーム機、デジタルカメラ等の他の電子機器とすることもできる。
【０１８１】
加えて、上記実施形態では、外部デバイスとしてプリンタを例に挙げて説明しているが、これに限定されるものではない。例えば、外部デバイスとして、他のスマートフォンやタブレットＰＣ、ＰＣやサーバ、ゲーム機、スキャナ等の、自身に関する情報を提供可能な他の電子機器も対象となる。例えば、スクリプト層から他の携帯型情報端末のバッテリ残量、通信状態、無線ＬＡＮの接続有無、ＧＰＳや温度、湿度、加速度等、その電子機器

が有する機能等の電子機器に関する情報を取得することも可能となる。

【０１８２】

また、外部デバイスの例として、電化製品や自動車も含まれる。例えば、携帯型情報端末上のスクリプト層から外部の冷蔵庫、洗濯機、エアコン、照明、掃除機、ポット等の電化機器の情報を取得したり、その電子機器の機能のオン／オフや出力の調整等も可能となる。

【０１８３】

加えて、上記実施形態では、コンテンツ（写真画像やスタンプ画像）の描画として、ＪａｖａＳｃｒｉｐｔのＣａｎｖａｓ機能で例に挙げて説明しているが、コンテンツの描画は、これに限定されるものではない。例えば、ＳＶＧ（Ｓｃａｌａｂｌｅ Ｖｅｃｔｏｒ Ｇｒａｐｈｉｃｓ）を利用して、コンテンツを描画することも可能である。

【０１８４】

加えて、上記実施形態のプリンタは、インクジェットプリンタ、レーザープリンタ、昇華型プリンタ、ドットインパクトプリンタ等を利用することができる。これらには、プリンタ機能、スキャナ機能等を有する、単一機能ではない、いわゆる、複合機の場合もある。

【０１８５】

尚、以上の実施形態の機能は以下の構成によっても実現することができる。つまり、本実施形態の処理を行うためのプログラムコードをシステムあるいは装置に供給し、そのシステムあるいは装置のコンピュータ（またはＣＰＵやＭＰＵ）がプログラムコードを実行することによっても達成される。この場合、記憶媒体から読み出されたプログラムコード自体が上述した実施形態の機能を実現することとなり、またそのプログラムコードを記憶した記憶媒体も本実施形態の機能を実現することになる。

【０１８６】

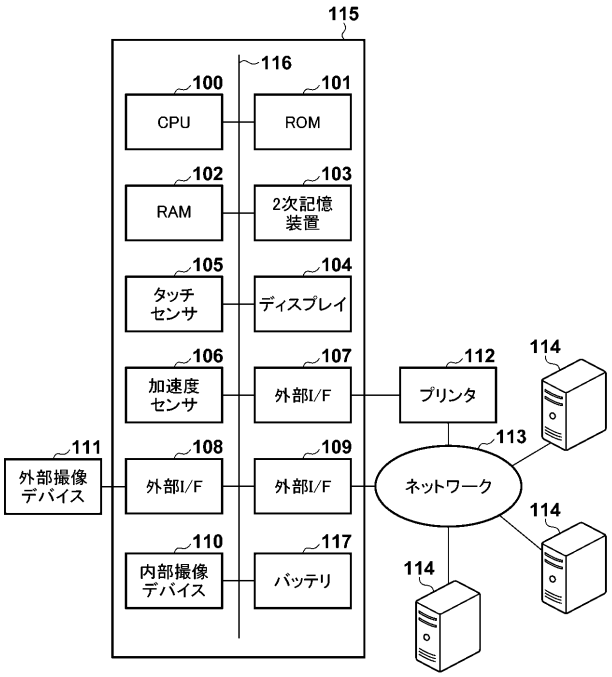
また、本実施形態の機能を実現するためのプログラムコードを、１つのコンピュータ（ＣＰＵ、ＭＰＵ）で実行する場合であってもよいし、複数のコンピュータが協働することによって実行する場合であってもよい。さらに、プログラムコードをコンピュータが実行する場合であってもよいし、プログラムコードの機能を実現するための回路等のハードウェアを設けてもよい。またはプログラムコードの一部をハードウェアで実現し、残りの部分をコンピュータが実行する場合であってもよい。

【符号の説明】

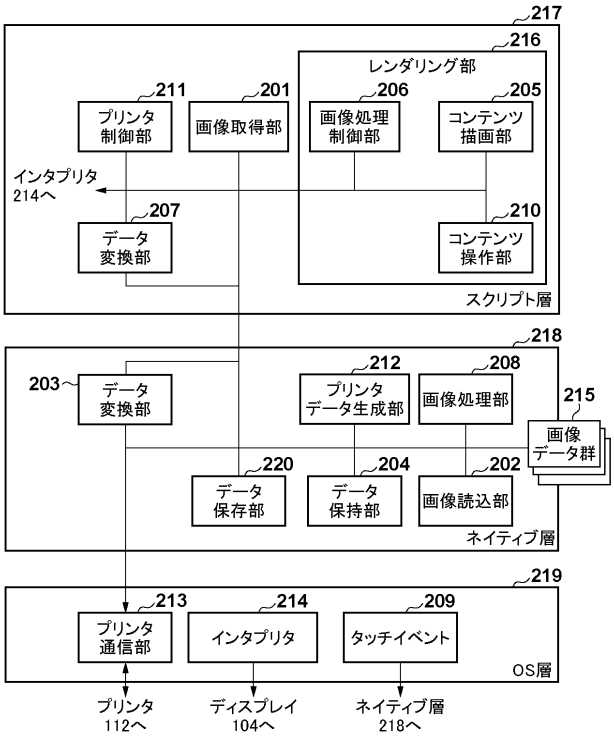
【０１８７】

２１７：スクリプト層、２１８：ネイティブ層、２１９：ＯＳ層、２１１：プリンタ制御部

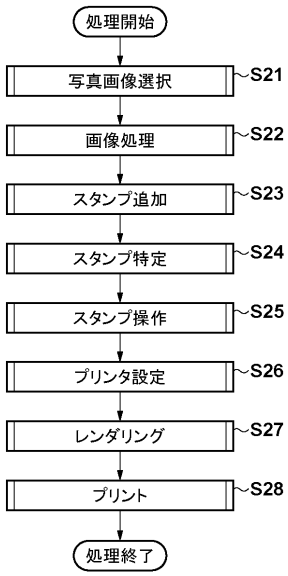
【図１】


115
116
100
CPU
101
ROM
102
RAM
103
2次記憶装置
105
タッチセンサ
104
ディスプレイ
106
加速度センサ
107
外部I/F
112
プリンタ
114
111
外部撮像デバイス
108
外部I/F
109
外部I/F
113
ネットワーク
114
110
内部撮像デバイス
117
バッテリ
114

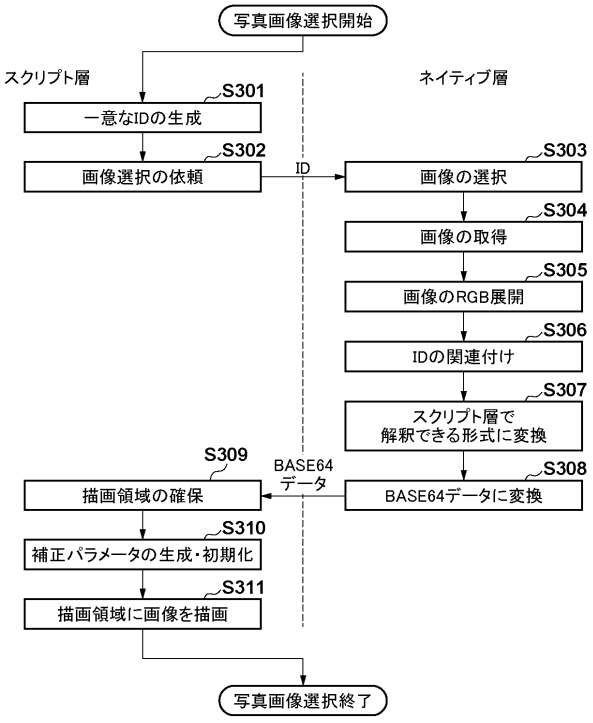
【図２】


217
216
レンダリング部
211
プリンタ制御部
201
画像取得部
206
画像処理制御部
205
コンテンツ描画部
インタプリタ214へ
207
データ変換部
210
コンテンツ操作部
スクリプト層
218
203
データ変換部
212
プリンタデータ生成部
208
画像処理部
215
画像データ群
220
データ保存部
204
データ保持部
202
画像読込部
ネイティブ層
219
213
プリンタ通信部
214
インタプリタ
209
タッチイベント
OS層
プリンタ112へ
ディスプレイ104へ
ネイティブ層218へ

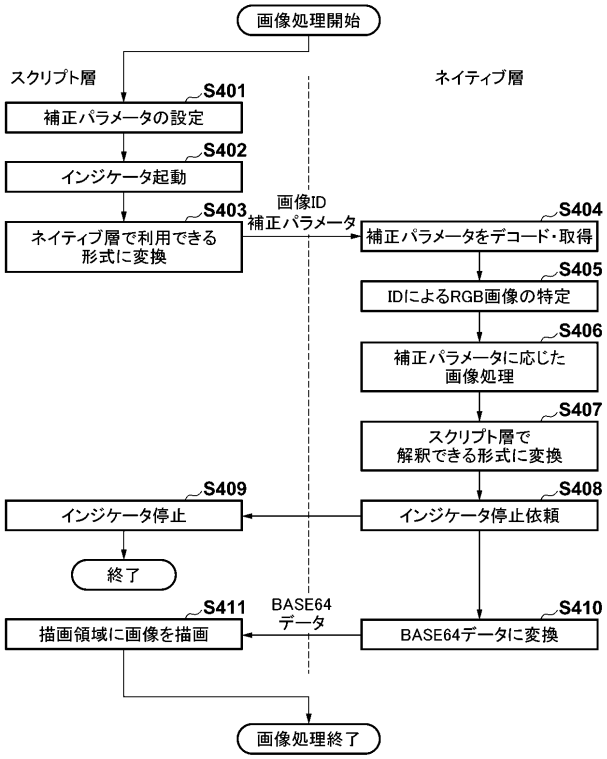
【図３】


処理開始
写真画像選択
S21
画像処理
S22
スタンプ追加
S23
スタンプ特定
S24
スタンプ操作
S25
プリンタ設定
S26
レンダリング
S27
プリント
S28
処理終了

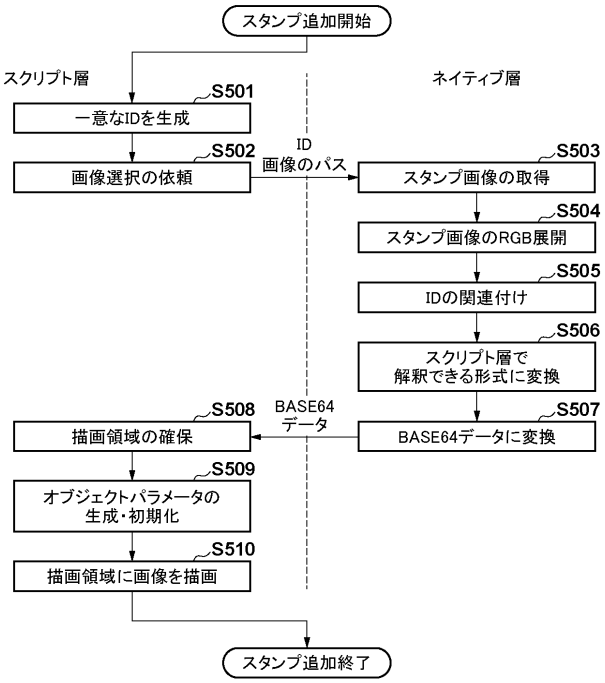
【図４】


写真画像選択開始
スクリプト層
ネイティブ層
S301
一意なIDの生成
S302
画像選択の依頼
ID
S303
画像の選択
S304
画像の取得
S305
画像のRGB展開
S306
IDの関連付け
S307
スクリプト層で解釈できる形式に変換
S309
BASE64データ
S308
BASE64データに変換
描画領域の確保
S310
補正パラメータの生成・初期化
S311
描画領域に画像を描画
写真画像選択終了

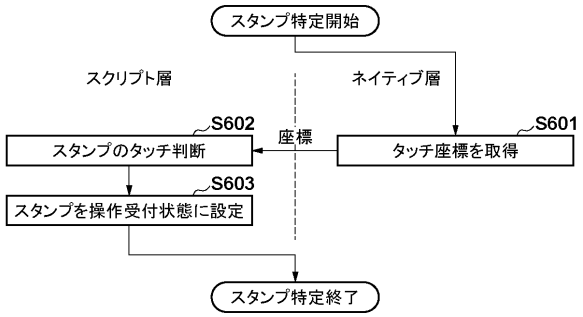
【図５】


画像処理開始
スクリプト層
ネイティブ層
S401
補正パラメータの設定
S402
インジケータ起動
S403
ネイティブ層で利用できる形式に変換
画像ID
補正パラメータ
S404
補正パラメータをデコード・取得
S405
IDによるRGB画像の特定
S406
補正パラメータに応じた画像処理
S407
スクリプト層で解釈できる形式に変換
S408
インジケータ停止依頼
S409
インジケータ停止
終了
S410
BASE64データに変換
BASE64データ
S411
描画領域に画像を描画
画像処理終了

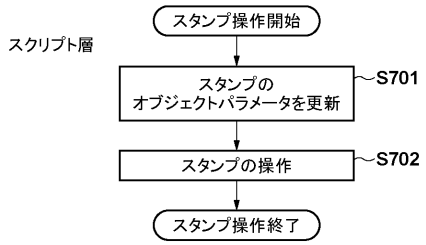
【図６】


スタンプ追加開始
スクリプト層
ネイティブ層
S501
一意なIDを生成
S502
画像選択の依頼
ID
画像のパス
S503
スタンプ画像の取得
S504
スタンプ画像のRGB展開
S505
IDの関連付け
S506
スクリプト層で解釈できる形式に変換
S507
BASE64データに変換
BASE64データ
S508
描画領域の確保
S509
オブジェクトパラメータの生成・初期化
S510
描画領域に画像を描画
スタンプ追加終了

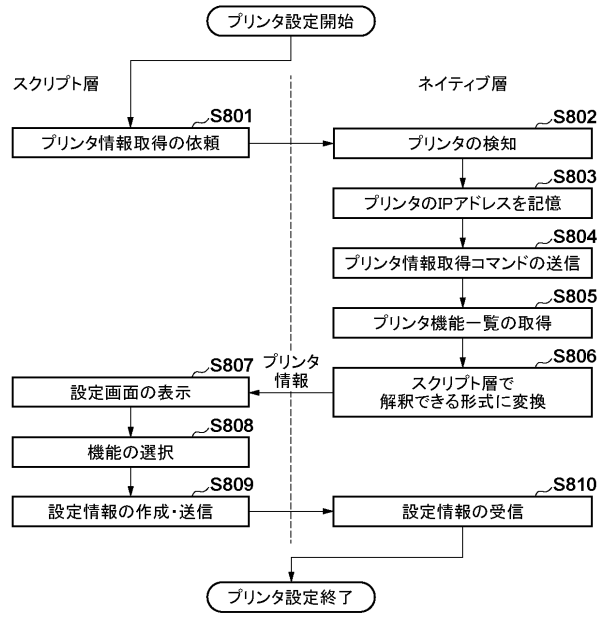
【図７】


スタンプ特定開始
スクリプト層
ネイティブ層
S602
スタンプのタッチ判断
座標
S601
タッチ座標を取得
S603
スタンプを操作受付状態に設定
スタンプ特定終了

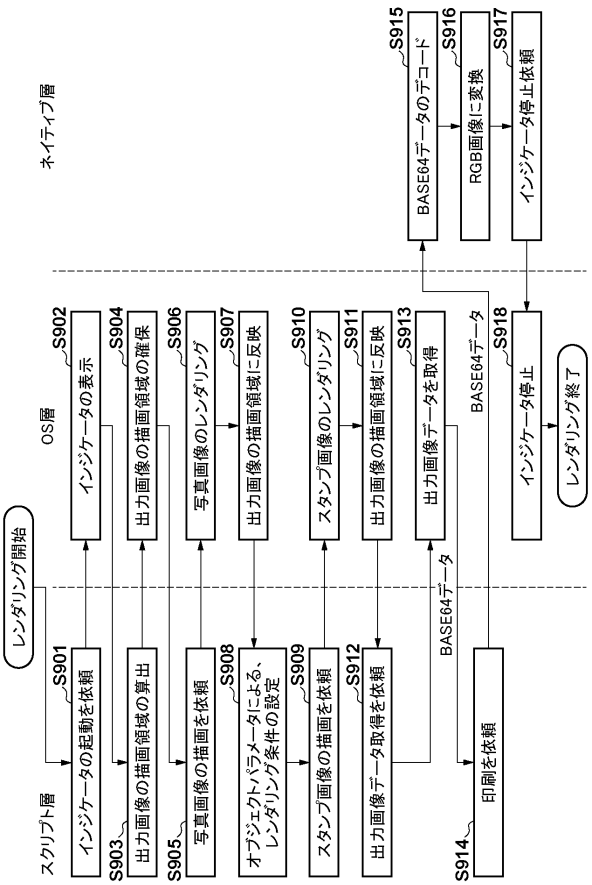
【図８】


スクリプト層
スタンプ操作開始
スタンプのオブジェクトパラメータを更新
S701
スタンプの操作
S702
スタンプ操作終了

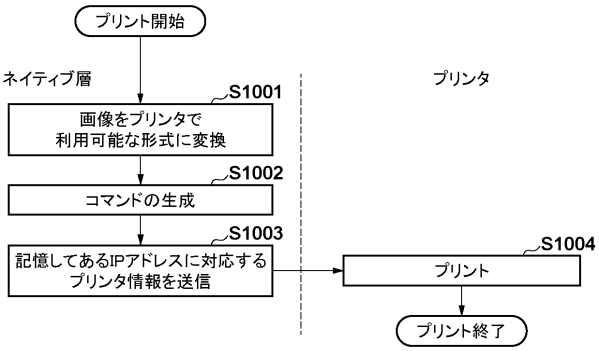
【図９】


プリンタ設定開始
スクリプト層
ネイティブ層
S801
プリンタ情報取得の依頼
S802
プリンタの検知
S803
プリンタのIPアドレスを記憶
S804
プリンタ情報取得コマンドの送信
S805
プリンタ機能一覧の取得
S806
スクリプト層で解釈できる形式に変換
プリンタ情報
S807
設定画面の表示
S808
機能の選択
S809
設定情報の作成・送信
S810
設定情報の受信
プリンタ設定終了

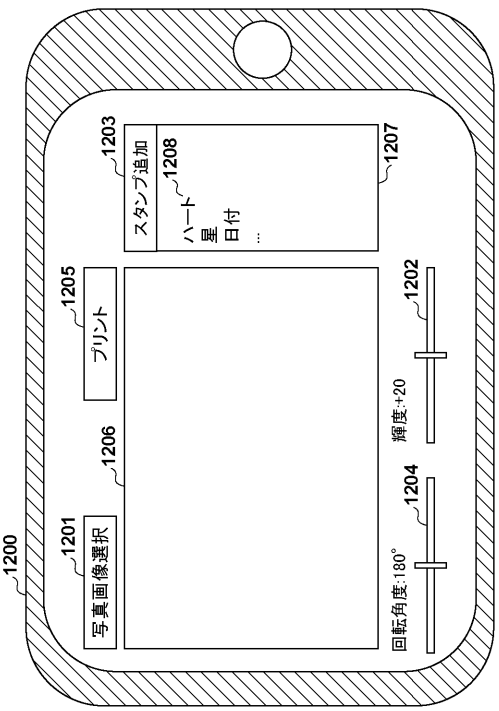
【図１０】



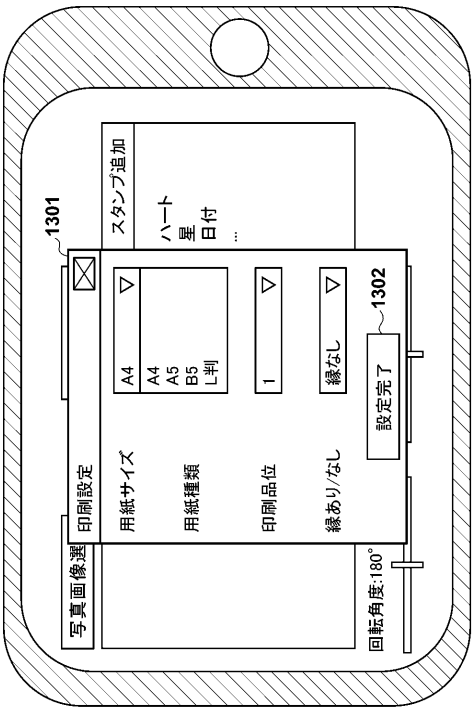
【図１１】



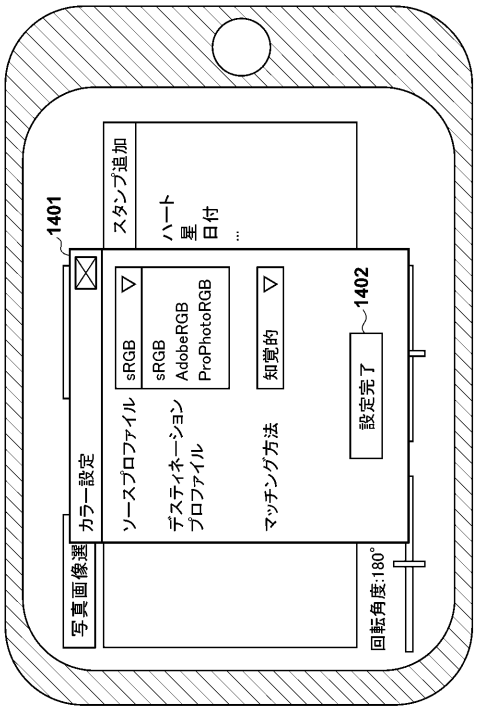
【図１２】



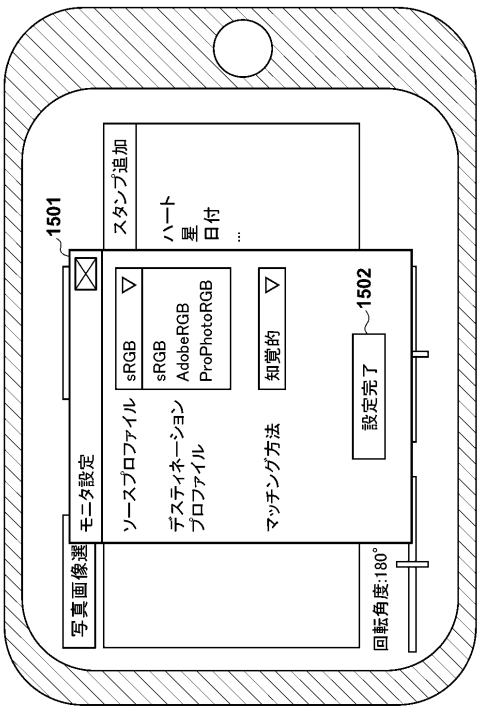
【図１３】



【図１４】


1401
写真画像選
カラー設定
スタンプ追加
ハート
星
日付
ソースプロファイル
sRGB
sRGB
AdobeRGB
ProPhotoRGB
デスティネーション
プロファイル
マッチング方法
知覚的
1402
設定完了
回転角度:180°

【図１５】


1501
写真画像選
モニタ設定
スタンプ追加
ハート
星
日付
ソースプロファイル
sRGB
sRGB
AdobeRGB
ProPhotoRGB
デスティネーション
プロファイル
マッチング方法
知覚的
1502
設定完了
回転角度:180°

フロントページの続き

(72)発明者　大林　正明
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内
(72)発明者　鷲見　尚紀
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内
(72)発明者　梅田　清
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内
(72)発明者　鈴木　智博
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

Ｆターム(参考) 5B057 CA01　CA08　CA12　CA16　CB01　CB08　CB12　CB16　CE17　CH01
CH07　CH11　DC25