(19)日本国特許庁(JP)　(12)特　許　公　報(B1)　(11)特許番号
特許第6913791号
(P6913791)
(45)発行日　令和3年8月4日(2021.8.4)　(24)登録日　令和3年7月14日(2021.7.14)

(51)Int.Cl.　　F I
A63F 13/85 (2014.01)　A63F 13/85
A63F 13/53 (2014.01)　A63F 13/53
A63F 13/70 (2014.01)　A63F 13/70

請求項の数 4　(全 22 頁)

(21)出願番号　特願2020-81051(P2020-81051)
(22)出願日　令和2年5月1日(2020.5.1)
審査請求日　令和2年8月19日(2020.8.19)

早期審査対象出願

前置審査

(73)特許権者　511249637
株式会社Ｃｙｇａｍｅｓ
東京都渋谷区南平台町１６番１７号
(74)代理人　100110928
弁理士　速水　進治
(74)代理人　100127236
弁理士　天城　聡
(72)発明者　花岡　洋輝
東京都渋谷区南平台町１６番１７号

審査官　宮本　昭彦

最終頁に続く

(54)【発明の名称】処理装置、処理方法及びプログラム

(57)【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アプリケーション内のコンテンツを実行した時の前記端末装置の状態を示す実行時状態情報、及び、前記コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、記憶手段に蓄積する情報蓄積部と、
前記記憶手段に蓄積された前記実行時状態情報及び前記バッテリ情報に基づき、前記端末装置の状態を示す情報から前記コンテンツを実行したときのバッテリ消費量を推定する推定モデルを生成する推定モデル生成部と、
所定の推定タイミングで前記端末装置の状態を示す推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定する推定部と、
前記推定部の推定結果を出力する出力部と、
を有し、
前記アプリケーションでは複数のコンテンツが実行可能であり、
前記情報蓄積部は、前記コンテンツ毎に、前記実行時状態情報及び前記バッテリ情報を前記記憶手段に蓄積し、
前記推定モデル生成部は、グループ毎に、少なくとも、前記グループに属する前記コンテンツの前記実行時状態情報及び前記バッテリ情報に基づき前記推定モデルを生成し、
前記推定部は、バッテリ消費量を推定する対象の前記コンテンツである対象コンテンツが属する前記グループを特定し、特定した前記グループに対応する前記推定モデルに基づ

き、前記対象コンテンツを実行したときのバッテリ消費量を推定する処理装置。
【請求項２】
前記端末装置が、前記コンテンツの実行を開始する入力を受付ける受付画面を表示する入力を受付けると、
前記推定部は、前記受付画面を表示する入力の受付に応じて前記端末装置の状態を示す前記推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定し、
前記出力部は、前記受付画面において、前記コンテンツに紐づけて前記推定結果を表示させる請求項１に記載の処理装置。
【請求項３】
コンピュータが、
端末装置にインストールされたアプリケーション内のコンテンツを実行した時の前記端末装置の状態を示す実行時状態情報、及び、前記コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、記憶手段に蓄積する情報蓄積工程と、
少なくとも、前記記憶手段に蓄積された前記実行時状態情報及び前記バッテリ情報に基づき、前記端末装置の状態を示す情報から前記コンテンツを実行したときのバッテリ消費量を推定する推定モデルを生成する推定モデル生成工程と、
所定の推定タイミングで前記端末装置の状態を示す推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定する推定工程と、
前記推定工程の推定結果を出力する出力工程と、
を実行し、
前記アプリケーションでは複数のコンテンツが実行可能であり、
前記コンピュータは、
前記情報蓄積工程では、前記コンテンツ毎に、前記実行時状態情報及び前記バッテリ情報を前記記憶手段に蓄積し、
前記推定モデル生成工程では、グループ毎に、少なくとも、前記グループに属する前記コンテンツの前記実行時状態情報及び前記バッテリ情報に基づき前記推定モデルを生成し、
前記推定工程では、バッテリ消費量を推定する対象の前記コンテンツである対象コンテンツが属する前記グループを特定し、特定した前記グループに対応する前記推定モデルに基づき、前記対象コンテンツを実行したときのバッテリ消費量を推定する処理方法。
【請求項４】
コンピュータを、
端末装置にインストールされたアプリケーション内のコンテンツを実行した時の前記端末装置の状態を示す実行時状態情報、及び、前記コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、記憶手段に蓄積する情報蓄積手段、
前記記憶手段に蓄積された前記実行時状態情報及び前記バッテリ情報に基づき、前記端末装置の状態を示す情報から前記コンテンツを実行したときのバッテリ消費量を推定する推定モデルを生成する推定モデル生成手段、
所定の推定タイミングで前記端末装置の状態を示す推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定する推定手段、
前記推定手段の推定結果を出力する出力手段、
として機能させ、
前記アプリケーションでは複数のコンテンツが実行可能であり、
前記情報蓄積手段は、前記コンテンツ毎に、前記実行時状態情報及び前記バッテリ情報を前記記憶手段に蓄積し、
前記推定モデル生成手段は、グループ毎に、少なくとも、前記グループに属する前記コ

ンテンツの前記実行時状態情報及び前記バッテリ情報に基づき前記推定モデルを生成し、
前記推定手段は、バッテリ消費量を推定する対象の前記コンテンツである対象コンテンツが属する前記グループを特定し、特定した前記グループに対応する前記推定モデルに基づき、前記対象コンテンツを実行したときのバッテリ消費量を推定するプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、処理装置、処理方法及びプログラムに関する。
【背景技術】
【０００２】
従来の研究や製品では、ＣＰＵやディスプレイといったハードウエアの部品を分析したり、ユーザの習慣的な行動によるアプリケーションの利用傾向を学習したりすることにより、アプリケーションによるバッテリ消費量や、今後のバッテリ消費量の推移を予測する。
【０００３】
非特許文献１は、利用者の習慣的な行動を反映して、数時間先までのバッテリ残量のグラフを予測して表示する機能を開示している。
【０００４】
非特許文献２に開示の技術では、スマートフォンの状態を５種に分類し、それぞれの状態が占める時間を以って、利用者を特徴付けている。具体的には、１）待ち受け、２）通話、３）第１の方式を利用したデータ通信、４）第２の方式を利用したデータ通信、５）その他、である。そして、ユーザのログを分析することにより、５種の状態に基づきユーザによる使用パターンの違いを分析する。
【０００５】
非特許文献３は、ＣＰＵやディスプレイといった電力消費量の大きなハードウエア部品の計測値をパラメータとする関数として、バッテリ消費量をモデル化する技術を開示している。
【０００６】
非特許文献４は、１日ないし数日間における長時間のプロファイルから、バッテリ消費量のモデルを学習することにより、"ｉｎ ｔｈｅ ｗｉｌｄ"のバッテリ消費量を指定する手法を開示している。当該手法は、ハードウエアの部品単位あるいはアプリケーション単位で、バッテリ消費量を推定することができる。
【０００７】
特許文献１は、バッテリ残量と所与の閾値を比較することにより、セーブを促す仕組みを開示している。
【０００８】
特許文献２は、電気自動車が走行を終了した際に、バッテリの残量と閾値との比較に基づいて充電を催促する仕組みを開示している。具体的には、車両の利用形態を学習し、学習結果に基づいて閾値設定するという思想が提案されている。
【０００９】
特許文献３は、充電場所、充電開始時刻及び充電時間の履歴から、日常の充電場所及び充電時間帯を学習し、利用者が充電を実行できそうな時だけ充電案内を行う仕組みを開示している。具体的には、バッテリの残電力量と、日常の走行にかかる消費電力量とを比較し、充電案内の実施可否を判定する思想が提案されている。
【００１０】
特許文献４は、バッテリ残量と所与の閾値を比較することにより、ゲームのパラメータを変更する仕組みを開示している。当該特許の実施形態の変形例によれば、バッテリが所定の残容量を下回った時に単位ゲームを選択できなくするという例が提示されている。すなわち、当該特許は、コンテンツ単位で、対応するバッテリ残量の閾値を、ゲーム開発者があらかじめ設計しておくものである。

【先行技術文献】
【特許文献】
【００１１】
【特許文献１】特許第４０３３４２７号
【特許文献２】特開２００３－２０９９０１号
【特許文献３】ＵＳ８９５４２２３Ｂ２
【特許文献４】ＵＳ１０４３４４１４Ｂ２
【非特許文献】
【００１２】
【非特許文献１】Michelle NYC, Smart battery on Pixel, https://support.google.com/pixelphone/forum/AAAAb4-OgUstyVoEaFNJ1k/?hl=by, retrieved November 25, 2019, Nov. 2017.
【非特許文献２】Joon-Myung Kang, Sin-Seok Seo, and James Won-Ki Hong, "Personalized Battery Lifetime Predic- tion for Mobile Devices based on Usage Patterns," Journal of Computing Science and Engineering, vol. 5, Dec. 2011, 338-345, doi: 10.5626/JCSE.2011.5.4.338.
【非特許文献３】Xia Zhao, Yao Guo, Qing Feng, and Xiangqun Chen, 2011, "A System Context-aware Approach for Battery Lifetime Prediction in Smart Phones," In Proceedings of the 2011 ACM Symposium on Applied Computing, SAC '11, ACM, TaiChung, Taiwan, 641-646, doi: 10.1145/1982185.1982327. http://doi.acm.org/10.1145/1982185.1982327.
【非特許文献４】Andres Munoz Medina, Ashish Sharma, Felix Yu, Paul Eastham, Sergei Vassilvitskii, and Umar Syed, 2016, "Learning mobile phone battery consumptions.", https://research.google/pubs/pub45901/.
【発明の概要】
【発明が解決しようとする課題】
【００１３】
近年、スマートフォン、タブレット端末、スマートウォッチ、携帯電話、携帯ゲーム機、ノートパソコン等のような可搬型の端末装置が広く普及している。このような可搬型の端末装置は、一般的には、間欠的に充電操作がなされる。例えば、外出中には充電操作は実行されず、自宅にいる間の任意のタイミングで充電操作が実行されたりする。このため、ユーザは、バッテリ残量を考慮し、次回の充電タイミングまでバッテリが残るように端末装置の使用を制限する等の対応が必要になる場合がある。
【００１４】
そこで、端末装置を用いた所定の処理（例えば、ゲームアプリケーションに含まれるコンテンツの実行）を実行した場合のバッテリ消費量を所定の処理を実行する前にユーザに通知する技術が望まれている。当該技術によれば、ユーザは、所定の処理を実行した場合のバッテリ消費量を考慮して、当該処理を実行するか否か等を判断することができる。しかし、アプリケーションの特定の機能を特定の端末で動作させた時の電力消費量を事前に推定することは難しい。例えばゲームのように複合的な機能を有するアプリケーションでは、サウンドやＧＰＵやネットワークを複合的に組み合わせるため、特定の機能を実行する時の電力消費量は、機能×デバイスの数だけ存在することになる。いずれの先行技術も、当該課題を開示していない。
【００１５】
なお、従来技術（例えば非特許文献４）では、ユーザの習慣的な利用傾向に基づきアプリケーション単位でバッテリ消費量を推定する技術を開示している。しかし、一般的に、アプリケーションは多機能であり、アプリケーション内のどの機能を利用するかに応じて、ユーザの利用時間やハードウエアの処理負担などは異なり得る。このため、ユーザの習慣的な利用傾向に基づくアプリケーション単位でのバッテリ消費量の推定は、１日単位や１カ月単位といった比較的大きな時間単位での推定に適しているが、より小さい時間単位

での推定には適していない。端末装置を用いた所定の処理を実行した場合のバッテリ消費量を所定の処理を実行する前にユーザに通知する技術においては、１日単位や１カ月単位といった比較的大きな時間単位でのバッテリ消費量の推定でなく、より小さい時間単位での推定が必要となる。

【００１６】

また、従来技術（例えば非特許文献３）では、ＣＰＵやディスプレイといった電力消費量の大きなハードウエア部品の計測値をパラメータとする関数として、バッテリ消費量をモデル化している。この技術の場合、所定の処理を開始した後に、所定の処理実行時のハードウエア部品の計測値に基づきバッテリ消費量を推定できる。しかし、所定の処理を開始する前に推定し、ユーザに通知することはできない。

【００１７】

本発明は、端末装置を用いた所定の処理を実行した場合のバッテリ消費量を所定の処理を実行する前にユーザに通知する技術を実現することを課題とする。

【課題を解決するための手段】

【００１８】

本発明によれば、

端末装置にインストールされたアプリケーション内のコンテンツを実行した時の前記端末装置の状態を示す実行時状態情報、及び、前記コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、記憶手段に蓄積する情報蓄積部と、

前記記憶手段に蓄積された前記実行時状態情報及び前記バッテリ情報に基づき、前記端末装置の状態を示す情報から前記コンテンツを実行したときのバッテリ消費量を推定する推定モデルを生成する推定モデル生成部と、

所定の推定タイミングで前記端末装置の状態を示す推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定する推定部と、

前記推定部の推定結果を出力する出力部と、

を有し、

前記アプリケーションでは複数のコンテンツが実行可能であり、

前記情報蓄積部は、前記コンテンツ毎に、前記実行時状態情報及び前記バッテリ情報を前記記憶手段に蓄積し、

前記推定モデル生成部は、前記コンテンツ毎に、前記コンテンツ各々を実行した時のバッテリ消費量を推定する前記推定モデルを生成し、

前記推定部は、前記コンテンツ各々を実行したときのバッテリ消費量を推定する記載の処理装置が提供される。

【００１９】

また、本発明によれば、

コンピュータが、

端末装置にインストールされたアプリケーション内のコンテンツを実行した時の前記端末装置の状態を示す実行時状態情報、及び、前記コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、記憶手段に蓄積し、

少なくとも、前記記憶手段に蓄積された前記実行時状態情報及び前記バッテリ情報に基づき、前記端末装置の状態を示す情報から前記コンテンツを実行したときのバッテリ消費量を推定する推定モデルを生成し、

所定の推定タイミングで前記端末装置の状態を示す推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定し、推定結果を出力し、

前記アプリケーションでは複数のコンテンツが実行可能であり、

前記コンピュータは、

前記コンテンツ毎に、前記実行時状態情報及び前記バッテリ情報を前記記憶手段に蓄積し、

前記コンテンツ毎に、前記コンテンツ各々を実行した時のバッテリ消費量を推定する前記推定モデルを生成し、
前記コンテンツ各々を実行したときのバッテリ消費量を推定する処理方法が提供される。
【００２０】
また、本発明によれば、
コンピュータを、
端末装置にインストールされたアプリケーション内のコンテンツを実行した時の前記端末装置の状態を示す実行時状態情報、及び、前記コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、記憶手段に蓄積する情報蓄積手段、
前記記憶手段に蓄積された前記実行時状態情報及び前記バッテリ情報に基づき、前記端末装置の状態を示す情報から前記コンテンツを実行したときのバッテリ消費量を推定する推定モデルを生成する推定モデル生成手段、
所定の推定タイミングで前記端末装置の状態を示す推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定する推定手段、
前記推定手段の推定結果を出力する出力手段、
として機能させ、
前記アプリケーションでは複数のコンテンツが実行可能であり、
前記情報蓄積手段は、前記コンテンツ毎に、前記実行時状態情報及び前記バッテリ情報を前記記憶手段に蓄積し、
前記推定モデル生成手段は、前記コンテンツ毎に、前記コンテンツ各々を実行した時のバッテリ消費量を推定する前記推定モデルを生成し、
前記推定手段は、前記コンテンツ各々を実行したときのバッテリ消費量を推定するプログラムが提供される。
【発明の効果】
【００２１】
本発明によれば、端末装置を用いた所定の処理を実行した場合のバッテリ消費量を所定の処理を実行する前にユーザに通知する技術が実現される。
【図面の簡単な説明】
【００２２】
【図１】本実施形態の処理装置のハードウエア構成の一例を示す図である。
【図２】本実施形態の処理装置の機能ブロックの一例を示す図である。
【図３】本実施形態の処理装置が処理する情報の一例を模式的に示す図である。
【図４】本実施形態の処理装置の処理の流れの一例を示すフローチャートである。
【図５】本実施形態の処理装置の処理の流れの一例を示すフローチャートである。
【図６】本実施形態の端末装置で表示される画面の一例を模式的に示す図である。
【図７】本実施形態の端末装置で表示される画面の一例を模式的に示す図である。
【図８】本実施形態の処理装置が処理する情報の一例を模式的に示す図である。
【図９】本実施形態の処理装置が処理する情報の一例を模式的に示す図である。
【発明を実施するための形態】
【００２３】
＜概要＞
本実施形態の処理装置は、端末装置の状態と、その状態でアプリケーション内の各コンテンツを実行した場合のバッテリ消費量との関係を示す推定モデルを生成する。そして、処理装置は、当該推定モデルに基づき、推定時の端末装置の状態で各コンテンツを実行した場合のバッテリ消費量を推定し、ユーザに通知する。
【００２４】
アプリケーションは、例えばゲームアプリケーションであるがこれに限定されない。アプリケーションは、実行可能な１つ又は複数のコンテンツを提供する。ユーザは、複数の

コンテンツの中から１つを選択して実行する。例えば、野球に関するゲームアプリケーションの場合、「試合（９イニング）」、「試合（７イニング）」、「練習（打撃）」、「練習（守備）」、「練習（投球）」、「トレード」、「ドラフト」等のコンテンツが提供され得る。

【００２５】

コンテンツは、アプリケーション内で実行可能な処理の１つである。このため、１つのコンテンツを実行している間におけるハードウエア処理負担の幅（最も処理負担が大きい時から最も処理負担が小さい時までの幅）は、複数のコンテンツを選択的に順次実行可能な１つのアプリケーションを実行している間におけるハードウエア処理負担の幅よりも小さい。また、１つのコンテンツの実行に要する時間（コンテンツを開始してから終了するまでの時間）の幅は、複数のコンテンツを選択的に順次実行可能な１つのアプリケーションの実行に要する時間の幅よりも小さい。

【００２６】

端末装置を用いた所定の処理を実行した場合のバッテリ消費量を所定の処理を実行する前にユーザに通知する技術において、このようなコンテンツ単位でバッテリ消費量を推定する技術を採用し、各コンテンツを実行する前に各コンテンツを実行した場合のバッテリ消費量を推定して通知するように構成することで、より精度が高く、有益な情報をユーザに提供することが可能となる。

【００２７】

＜ハードウエア構成＞

次に、処理装置のハードウエア構成を説明する。処理装置は、ユーザが操作する端末装置であってもよいし、当該端末装置と通信するサーバであってもよい。端末装置は、例えば、スマートフォン、タブレット端末、スマートウォッチ、携帯電話、携帯ゲーム機、ノートパソコン等のような可搬型の端末装置であるが、これに限定されない。

【００２８】

処理装置が備える各機能部は、任意のコンピュータのＣＰＵ（Central Processing Unit）、メモリ、メモリにロードされるプログラム、そのプログラムを格納するハードディスク等の記憶ユニット（あらかじめ装置を出荷する段階から格納されているプログラムのほか、ＣＤ（Compact Disc）等の記憶媒体やインターネット上のサーバ等からダウンロードされたプログラムをも格納できる）、ネットワーク接続用インターフェイスを中心にハードウエアとソフトウエアの任意の組合せによって実現される。そして、その実現方法、装置にはいろいろな変形例があることは、当業者には理解されるところである。

【００２９】

図１は、処理装置のハードウエア構成を例示するブロック図である。図１に示すように、処理装置は、プロセッサ１Ａ、メモリ２Ａ、入出力インターフェイス３Ａ、周辺回路４Ａ、バス５Ａを有する。周辺回路４Ａには、様々なモジュールが含まれる。なお、処理装置は周辺回路４Ａを有さなくてもよい。

【００３０】

処理装置は、物理的及び論理的に１つの装置で構成されてもよい。処理装置は、物理的及び／又は論理的に分かれた複数の装置で構成されてもよい。この場合、複数の装置各々が上記ハードウエア構成を備えることができる。

【００３１】

バス５Ａは、プロセッサ１Ａ、メモリ２Ａ、周辺回路４Ａ及び入出力インターフェイス３Ａが相互にデータを送受信するためのデータ伝送路である。プロセッサ１Ａは、例えばＣＰＵ、ＧＰＵ（Graphics Processing Unit）などの演算処理装置である。メモリ２Ａは、例えばＲＡＭ（Random Access Memory）やＲＯＭ（Read Only Memory）などのメモリである。入出力インターフェイス３Ａは、入力装置、外部装置、外部サーバ、外部センサ等から情報を取得するためのインターフェイスや、出力装置、外部装置、外部サーバ等に情報を出力するためのインターフェイスなどを含む。入力装置は、例えばキーボード、マウス、マイク等である。出力装置は、例えばディスプレイ、スピーカ、プリンター、メーラ

等である。プロセッサ１Ａは、各モジュールに指令を出し、それらの演算結果をもとに演算を行うことができる。

【００３２】

＜機能構成＞

次に、処理装置の機能構成を説明する。図２に、処理装置１０の機能ブロック図の一例を示す。図示するように、処理装置１０は、情報蓄積部１１と、推定モデル生成部１２と、推定部１３と、出力部１４と、記憶部１５とを有する。

【００３３】

情報蓄積部１１は、端末装置にインストールされたアプリケーション内のコンテンツを実行した時の端末装置の状態を示す実行時状態情報、及び、コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、履歴情報として記憶部１５に蓄積する。アプリケーションが複数のコンテンツを提供する場合、情報蓄積部１１は、コンテンツ毎に、実行時状態情報及びバッテリ情報を記憶部１５に蓄積する。

【００３４】

処理装置１０が端末装置である場合、情報蓄積部１１は、自装置が備えるセンサや機能を介して自装置に関する実行時状態情報及びバッテリ情報を取得する。一方、処理装置１０が端末装置と通信するサーバである場合、情報蓄積部１１は、端末装置が収集したその端末装置に関する実行時状態情報及びバッテリ情報を、インターネット等のネットワークを介して端末装置から受信する。

【００３５】

端末装置の状態を示す実行時状態情報は、取得可能な情報であって、バッテリ消費量に影響し得る任意の情報を含むことができる。例えば、実行時状態情報としては、端末装置にインストールされているＯＳ（operating system）の種類、通信ネットワークの接続方式、ディスプレイの輝度、バックグラウンドで起動中のアプリケーションの種類、ＣＰＵ使用率、ネットワーク通信量（byte/second）等が例示されるが、これらに限定されない。

【００３６】

コンテンツを実行した時のバッテリ消費量は、コンテンツ開始時のバッテリ残量とコンテンツ終了時のバッテリ残量とに基づき算出可能である。

【００３７】

図３に、実行時状態情報及びバッテリ情報を示す履歴情報の一例を模式的に示す。図示する例では、コンテンツＩＤと、コンテンツの実行を開始した日時であるコンテンツ開始日時（$Time_{start}$）と、コンテンツの実行を終了した日時であるコンテンツ終了日時（$Time_{end}$）と、コンテンツ開始日時におけるバッテリ残量（$Energy_{start}$）と、コンテンツ終了日時におけるバッテリ残量（$Energy_{end}$）と、コンテンツ実行時のＯＳの種類（K_1（ＯＳ））と、コンテンツ実行時の通信ネットワークの接続方式（K_2（Network））と、端末装置の種類（K_n（UA））とが互いに紐付けられている。

【００３８】

ここで、図４のフローチャートを用いて、情報蓄積部１１が実行する処理の流れの一例を説明する。ここでは、処理装置１０が端末装置であるものとする。

【００３９】

情報蓄積部１１は、コンテンツの実行開始を監視している（Ｓ１０）。情報蓄積部１１は、コンテンツの実行開始を検出すると（Ｓ１０のＹｅｓ）、その時点の端末装置の状態（実行時状態情報）やバッテリ残量を示す開始時状態情報を取得する（Ｓ１１）。そして、情報蓄積部１１は、取得した開始時状態情報を、実行開始されるコンテンツのコンテンツＩＤに紐付けて記憶部１５が記憶する履歴情報（図３参照）に登録する（Ｓ１２）。

【００４０】

その後、情報蓄積部１１は、そのコンテンツの実行終了を監視する（Ｓ１３）。情報蓄積部１１は、コンテンツの実行終了を検出すると（Ｓ１３のＹｅｓ）、その時点の端末装置のバッテリ残量を示す終了時状態情報を取得し（Ｓ１４）、記憶部１５が記憶する履歴

情報（図３参照）に登録する（Ｓ１５）。

【００４１】

情報蓄積部１１は当該処理を繰り返す。結果、図３に示すような履歴情報が蓄積されていく。なお、処理装置１０がサーバである場合、例えば、端末装置が図４のフローチャートで示す処理を実行し、端末装置内に履歴情報を蓄積する。そして、端末装置は、任意のタイミングで、自装置内に蓄積した履歴情報をサーバに送信する。その他、端末装置は図４のフローチャートで示す処理を実行し、Ｓ１１及びＳ１４で開始時状態情報及び終了時状態情報を取得すると、リアルタイム処理で取得した情報をサーバに送信してもよい。そして、サーバが、受信した開始時状態情報及び終了時状態情報を自装置内に登録してもよい。

【００４２】

図２に戻り、推定モデル生成部１２は、記憶部１５に蓄積された履歴情報（実行時状態情報及びバッテリ情報）に基づき、「端末装置の状態を示す情報」から「コンテンツを実行したときのバッテリ消費量」を推定する推定モデルを生成する。

【００４３】

推定モデルは、例えば、端末装置の状態を示す情報と、当該状態でコンテンツを実行したときのバッテリ消費量（＝（$Energy_{start}$）－（$Energy_{end}$））とを紐付けた教師データに基づく任意の機械学習で生成されたモデルであってもよい。この場合、推定モデルの出力は、コンテンツを実行したときのバッテリ消費量となる。その他、推定モデルは、端末装置の状態を示す情報と、コンテンツを実行する前のバッテリ残量（$Energy_{start}$）と、コンテンツを実行した後のバッテリ残量（$Energy_{end}$）とを紐付けた教師データに基づく任意の機械学習で生成されたモデルであってもよい。この場合、推定モデルの出力は、コンテンツを実行した後のバッテリ残量となる。なお、いずれの場合も、バッテリ消費量を推定している。

【００４４】

その他、推定モデルは、端末装置の状態を示す情報と、当該状態でコンテンツを実行した場合のバッテリ消費量との関係を示すテーブルをパターンマッチング等で照合し、キー（端末装置の状態を示す情報）に紐付くバッテリ消費量を特定するモデルであってもよい。その他、推定モデルは、端末装置の状態を示す情報と、コンテンツを実行する前のバッテリ残量（$Energy_{start}$）と、コンテンツを実行した後のバッテリ残量（$Energy_{end}$）との関係を示すテーブルをパターンマッチング等で照合し、キー（端末装置の状態を示す情報及びコンテンツを実行する前のバッテリ残量（$Energy_{start}$））に紐付くバッテリ消費量を特定するモデルであってもよい。

【００４５】

その他、推定モデルは、相関量を計量するためのベクトルに写像する等の手法を用いるモデルであってもよい。

【００４６】

なお、アプリケーションが複数のコンテンツを提供する場合、推定モデル生成部１２は、コンテンツ毎に推定モデルを生成することができる。

【００４７】

推定部１３は、所定の推定タイミングで端末装置の状態を示す推定時状態情報を取得し、推定時状態情報と推定モデルとに基づき、推定時状態情報で示される状態の端末装置でコンテンツを実行したときのバッテリ消費量を推定する。

【００４８】

推定時状態情報は、実行時状態情報と同種の情報を含む。推定時状態情報としては、例えば、端末装置にインストールされているＯＳの種類、通信ネットワークの接続方式、ディスプレイの輝度、バックグラウンドで起動中のアプリケーションの種類等が例示されるが、これらに限定されない。また、推定モデルの内容によっては、推定時のバッテリ残量が推定時状態情報に含まれる。

【００４９】

処理装置１０が端末装置である場合、推定部１３は、自装置が備えるセンサや機能を介して自装置に関する推定時状態情報を取得する。一方、処理装置１０が端末装置と通信するサーバである場合、推定部１３は、端末装置が収集したその端末装置に関する推定時状態情報を、インターネット等のネットワークを介して端末装置から受信する。

【００５０】

出力部１４は、推定部１３の推定結果を出力する。処理装置１０が端末装置である場合、出力部１４は、端末装置が備えるディスプレイやスピーカ等の出力装置を介して、推定結果を出力する。一方、処理装置１０が端末装置と通信するサーバである場合、出力部１４は、推定結果を端末装置に送信する。そして、端末装置は、自装置が備えるディスプレイやスピーカ等の出力装置を介して、受信した推定結果を出力する。

【００５１】

ここで、図５のフローチャートを用いて、推定部１３及び出力部１４による処理の流れの一例を説明する。

【００５２】

まず、推定部１３は、バッテリ消費量を推定する推定タイミングの到来を監視している（Ｓ２０）。推定タイミングの具体例は後述する。

【００５３】

そして、推定タイミングの到来を検出すると（Ｓ２０のＹｅｓ）、推定部１３は、推定対象のコンテンツのコンテンツＩＤを取得する（Ｓ２１）。推定対象のコンテンツは１つであってもよいし、複数であってもよい。推定対象のコンテンツの具体例は後述する。

【００５４】

また、推定部１３は、その時点における端末装置の状態を示す推定時状態情報を取得する（Ｓ２２）。推定部１３は、推定時のバッテリ残量を推定時状態情報として取得してもよい。なお、Ｓ２１及びＳ２２の処理順はこれに限定されない。

【００５５】

次いで、推定部１３は、Ｓ２１で取得したコンテンツＩＤに対応する推定モデルと、Ｓ２２で取得した推定時状態情報とに基づき、当該推定時状態情報で示される状態の端末装置で、当該コンテンツＩＤで示されるコンテンツを実行した場合のバッテリ消費量を推定する（Ｓ２３）。

【００５６】

次いで、出力部１４は、Ｓ２３の推定処理で得られた推定結果を出力する（Ｓ２５）。

【００５７】

ここで、推定タイミング、推定対象のコンテンツ及び推定結果の出力画面の一例を説明する。

【００５８】

「例１」

当該例では、推定タイミングは、コンテンツの実行を開始する入力を受付ける受付画面を表示する入力（ユーザ入力）を受付けたタイミングである。そして、当該例では、推定対象のコンテンツは、当該受付画面で実行を開始する入力を受付可能なコンテンツである。

【００５９】

推定部１３は、当該受付画面を表示する入力の受付に応じて推定時状態情報を取得し、推定時状態情報と推定モデルとに基づき、その推定時状態情報で示される状態の端末装置で上記推定対象のコンテンツ各々を実行したときのバッテリ消費量を推定する。

【００６０】

そして、出力部１４は、当該受付画面において、各コンテンツに紐づけて推定結果を表示させる。図６に、当該例で端末装置に表示される当該受付画面の一例を模式的に示す。当該例では、アプリケーションは、野球に関するゲームアプリケーションである。そして、当該受付画面は、「試合（９イニング）」、「試合（７イニング）」、「練習（打撃）」、「練習（守備）」、「練習（投球）」等のコンテンツの実行を開始する入力を受付け

可能になっている。そして、各コンテンツに紐付けて、その時点で各コンテンツを実行した場合のバッテリ消費量の予測結果が示されている。

【００６１】

「例２」

当該例では、推定タイミングは、アプリケーションが提供する複数のコンテンツ各々のバッテリ消費量の一覧を表示する入力（ユーザ入力）を受付けたタイミングである。そして、当該例では、推定対象のコンテンツは、アプリケーションが提供するすべてのコンテンツである。

【００６２】

推定部１３は、当該一覧を表示する入力の受付に応じて推定時状態情報を取得し、推定時状態情報と推定モデルとに基づき、その推定時状態情報で示される状態の端末装置で上記推定対象のコンテンツ各々を実行したときのバッテリ消費量を推定する。

【００６３】

そして、出力部１４は、当該一覧を表示する画面において、各コンテンツに紐づけて推定結果を表示させる。図７に、当該例で端末装置に表示される画面の一例を模式的に示す。当該画面では、アプリケーションが提供するすべてのコンテンツ各々に紐付けて、その時点で各コンテンツを実行した場合のバッテリ消費量の予測結果が示されている。

【００６４】

なお、図６及び図７では、満充電電力量に対する消費電力量の割合でバッテリ消費量を示しているが、その他の手法でバッテリ消費量を示してもよい。しかし、一般的な端末装置では、満充電電力量に対する残電力量の割合でバッテリ残量を示す。このため、図示するような手法でバッテリ消費量を示すと、ユーザがバッテリ残量とバッテリ消費量とを対比しやすくなり好ましい。

【００６５】

また、図６及び図７では、出力部１４は、推定結果としてコンテンツ実行によるバッテリ消費量を示しているが、その他、コンテンツ実行後のバッテリ残量の予測値を推定結果として出力してもよい。

【００６６】

＜変形例１＞

当該変形例では、アプリケーションが提供する複数のコンテンツは、バッテリの消費の仕方が類似するもの同士でグループ化される。そして、図８に示すような、各コンテンツがどのグループに属するかを示すグループ情報が処理装置１０内に予め記憶されている。

【００６７】

推定モデル生成部１２は、グループ毎に、グループに属するコンテンツの履歴情報（実行時状態情報及びバッテリ情報）に基づき上述した推定モデルを生成する。すなわち、変形例では、コンテンツ毎でなく、グループ毎に推定モデルが生成される。

【００６８】

そして、推定部１３は、バッテリ消費量を推定する対象のコンテンツが属するグループを特定し、特定したグループに対応する推定モデルに基づき、その時点でそのコンテンツを実行したときのバッテリ消費量を推定する。

【００６９】

＜変形例２＞

処理装置１０がサーバである場合、処理装置１０は、複数のユーザの端末装置から履歴情報（実行時状態情報及びバッテリ情報）を取得することができる。処理装置１０は、ユーザ毎に各ユーザの履歴情報を処理して推定モデルを生成してもよいし、複数のユーザの履歴情報をまとめて処理して複数のユーザ毎に推定モデルを生成してもよい。また、複数のユーザの属性情報（性別、年令、国籍、アプリケーションの利用歴（年数等）、端末装置の情報（メーカ、型番、ＯＳ、通信サービスを提供するキャリア等）等）が処理装置１０内に登録されている場合、処理装置１０は、属性情報が一致又は類似するユーザ同士でグループ化し、グループ毎に各グループに属するユーザの履歴情報をまとめて処理してグ

ループ毎に推定モデルを生成してもよい。
【００７０】
＜変形例３＞
処理装置１０は、少なくとも実行時状態情報及びバッテリ情報に基づき推定モデルを生成し、少なくとも実行時状態情報及びバッテリ情報に基づきバッテリ消費量を推定する。処理装置は、実行時状態情報及びバッテリ情報以外の情報であって、バッテリ消費量に影響し得るその他の情報を、例えば外部装置（端末装置以外の装置）から取得し、当該情報をさらに利用して推定モデルの生成、及び、バッテリ消費量の推定を行ってもよい。例えば、使用している部品（ハードウエア）ごとに推定モデルを構築しておき、当該推定モデルと、実行時状態情報に基づく推定モデルとを組み合わせてバッテリ消費量を推定してもよい。
【００７１】
＜実施例＞
次に、本実施形態の処理装置１０をより具体化した実施例を説明する。
【００７２】
本実施例は、ユーザがこれから実行するコンテンツを選択するための画面に遷移した時に、その画面に選択可能に表示されるコンテンツを実行した場合のバッテリ消費量を予測し、予測結果をその画面に表示する方式である。例えば、ユーザが２つのコンテンツ（００１－ＮＯＲＭＡＬと００１－ＨＡＲＤ）を選択可能な画面に遷移した時、本実施例の処理装置１０は、「００１－ＮＯＲＭＡＬはバッテリの０．５％を消費」、「００１－ＨＡＲＤはバッテリの０．８％を消費」のように、コンテンツごとにその時に実行した場合のバッテリ消費量を予測し、予測した結果を、コンテンツを開始するためのボタンと合わせて表示する。
【００７３】
本実施例の処理装置１０のシステムアーキテクチャは、コンテンツごとのバッテリ消費量を予測するための推定フェーズと、予測のための推定モデルを生成するための学習フェーズとに分けられる。ここでは、本実施例の処理装置１０を実現するための３つのモジュールについて説明した後、学習フェーズと推定フェーズについて詳述する。
【００７４】
「モジュール」
本実施例の処理装置１０は、［Ｍ１］Learning Moduleと、［Ｍ２］Model Dataと、［Ｍ３］Inference Moduleとの３モジュールを含んで構成される。
【００７５】
［Ｍ１］Learning Moduleは、開発者やユーザがコンテンツを実行した時の履歴情報（実行時状態情報及びバッテリ情報）を入力として、コンテンツごとのバッテリ消費量を予測するためのモデルをＭ２として出力する関数である。Ｍ１の入力となる履歴情報は、以下の式（１）及び式（２）に示すような時系列データとして定義できる。
【００７６】
【数１】

$$History = Item_{t1}, \ldots, Item_{t_k} \quad \cdots \text{式}(1)$$

【００７７】
【数２】

$$Item_{time-index} = (ContentID, Time_{start}, Time_{end}, Energy_{start}, Energy_{end}, [(K_1, V_1), \ldots (K_n, V_n)]) \quad \cdots \text{式}(2)$$

【００７８】
ここで、$Item_t$は、コンテンツごとの履歴情報の項目である。ContentIDは、コンテンツごと、もしくは、コンテンツのグループごとにあらかじめ設定されたユニークな識別子である。$Time_{start}$は、当該コンテンツを開始した時刻である。$Time_{end}$は、当該コンテンツ

を終了した時刻である。$Energy_{start}$は、当該コンテンツを開始した時刻におけるバッテリ残量である。$Energy_{end}$は、当該コンテンツを終了した時刻におけるバッテリ残量である。$Energy_{start}$及び$Energy_{end}$の値は、［０，１］の範囲の浮動小数点数とする。$Energy_{start}$及び$Energy_{end}$の値が０のとき、バッテリが完全に放電した状態（残量０）であることを示す。そして、$Energy_{start}$及び$Energy_{end}$の値が１のとき、バッテリが完全に充電された状態（満充電状態）であることを示す。

【００７９】

(K_i,V_i)は、ハードウエアの構成要素やセンサ情報の種類を示すKeyと、その値を示すValueのペアである。このKey-Valueペアは、例えば、ＯＳの種類のような静的な情報や、ネットワーク接続方式のような動的に変化する情報を含む。この履歴情報の実装方法として、開発者が、リリース前に、デバッグやテストのために蓄積したものであっても良いし、ユーザが、以前に同じ端末で実行した時に蓄積したものであっても良い。履歴情報の実装の例は、図３で示される。

【００８０】

［Ｍ２］Model Dataは、Ｍ１によって出力され、Ｍ３の入力となる推定モデルデータである。本モジュールの具体的な実装は、Ｍ１の実装に応じて設定されるが、例を図９に示す。図９の詳細は、後述する。

【００８１】

［Ｍ３］Inference Moduleは、ユーザがこれから実行するコンテンツを選択するための画面に遷移した時に、その画面に表示される予定のコンテンツのリスト（ContentIDのリスト）と、その時刻におけるスマートフォンのバッテリ残量（$Energy_{start}$）と、その時刻におけるハードウエアの構成要素やセンサ情報（［(K_1,V_1)，・・・・(K_m,V_m)］）を入力として受け取り、受け取った各ContentIDを対象に、入力の時点からユーザが当該コンテンツを実行したと仮定して、実行し終わった時のバッテリ残量（$Energy_{end}$）を予測して出力する関数である。当該関数は以下の式（３）及び式（４）のように示される。

【００８２】

【数３】

$$Input_{inference} = array\ of\ (ContentID, Energy_{start}, [(K_1,V_1),\ldots,(K_m,V_m)]) \quad \cdots 式(3)$$

【００８３】

【数４】

$$Output_{inference} = array\ of\ (ContentID, Energy_{end}) \quad \cdots 式(4)$$

【００８４】

本モジュールの具体的な実装は、Ｍ１の実装に応じて設定されるが、本実施例では、最も単純な実施例として、テーブルを用いたパターンマッチングによる推定を行う方式を採用する。他の実装例として、特徴量を抽出して何らかの機械学習を行っても良いし、相関量を計量するためのベクトルに写像しても良い。

【００８５】

「学習フェーズ」

次に、本実施例の学習フェーズを説明する。本実施例の学習フェーズでは、開発者やユーザがゲームのコンテンツを実行した時の履歴情報（実行時状態情報及びバッテリ情報）を用いて、パターンマッチングによる推定を行うためのテーブルを生成する。例えば、本実施例の処理装置１０は、図３に示すような履歴情報に基づき、以下の式（５）のように定義されるテーブルを生成する。

【００８６】

【数５】

$$PatternMatchTable = array\ of\ (ContentID, Energy_{start}, Energy_{end}, [(K_1,V_1),\ldots,(K_m,V_m)]) \quad \cdots 式(5)$$

【００８７】
式（５）で定義されるテーブルの一例は、図９に示される。図９に示すテーブルは、ContentIDごとに$Energy_{start}$を０．０５刻みで区切り、(K_i,V_i)と$Energy_{end}$を対応付けたものである。ここで、(K_i,V_i)は、収集されたデータのうち、推定のために必要なものだけ使用し、その他は使用しないといった選別を行っても良い。
【００８８】
次に、学習フェーズにおける処理の流れを説明する。
【００８９】
「Ｓｔｅｐ１」：開発者やユーザがアプリケーションを開始した時、処理装置１０は、履歴情報を格納するためのテーブルHistoryを作成する。処理装置１０は、アプリケーションの異常終了に備えて、永続化できる媒体にHistoryを作成することが好ましいが、実装の要件に応じてメモリ上に作成しても良い。
【００９０】
処理装置１０は、過去に作成したHistoryが残っていた時、後述するＳｔｅｐ４の学習モデルの更新の処理を実行し、Historyを空にする。
【００９１】
「Ｓｔｅｐ２」:開発者やユーザが、開発者によって識別子を定義されているコンテンツの実行を開始した時、処理装置１０は、Ｓｔｅｐ１で作成したHistoryに新たなエントリ$Item_t$を追加し、次の４種の情報を記録する。
【００９２】
・開始されたコンテンツのContentID
・当該コンテンツを開始した時刻$Time_{start}$
・当該コンテンツを開始した時刻におけるバッテリ残量$Energy_{start}$
・当該コンテンツを開始した時刻におけるハードウエアの構成要素やセンサ情報［(K_1,V_1)・・・・(K_n,V_n)］
【００９３】
「Ｓｔｅｐ３」：開発者やユーザが、開発者によって識別子を定義されているコンテンツの実行を終了した時、処理装置１０は、Ｓｔｅｐ１で作成し、Ｓｔｅｐ２で更新したHistoryに、次の２種の情報を記録する。
【００９４】
・当該コンテンツを終了した時刻$Time_{end}$
・当該コンテンツを終了した時刻におけるバッテリ残量$Energy_{end}$
【００９５】
「Ｓｔｅｐ４」:開発者やユーザが、ゲームの実行を中断あるいは終了した時、処理装置１０は、Historyに記録した$Item_{t1}$・・・$Item_{tk}$を使って推定モデルを更新し、更新が完了したらHistoryを空にする。
【００９６】
本実施例では、処理装置１０は、Historyに記録した$Item_{t1}$・・・$Item_{tk}$を使って図９に示すテーブルPatternMatchTableを更新する。具体的には、処理装置１０は、Historyを走査し、各要素$Item_t$について、以下に示すＳｔｅｐ４－１、Ｓｔｅｐ４－２を実行する。
【００９７】
「Ｓｔｅｐ４－１」:まず、処理装置１０は、コンテンツの開始時刻におけるバッテリ残量$Energy_{start}$について、［０、０．０５、０．１０、・・・０．９５、１．００］のように、複数のインデックス値$Energy_{startindex}$を設定している。そして、処理装置１０は、Historyに記録された複数の$Item_t$の中の処理対象の$Item_t$の$Energy_{start}$に最も近い$Energy_{startindex}$を特定する。例えば上述のように０．０５刻みで$Energy_{startindex}$を設定した場合において、処理対象の$Item_t$の$Energy_{start}$が０．７６である場合、０．７５が最も近い$Energy_{startindex}$として特定される。
【００９８】

さらに、処理装置１０は、処理対象の$Item_t$のハードウエアの構成要素やセンサ情報（［(K_1,V_1)，・・・・(K_m,V_m)］）の中から、推定時に必要となる要素を取り出す。最も単純には同じ配列を使っても良い。

【００９９】

「Ｓｔｅｐ４－２」：次に、処理装置１０は、図９に示すPatternMatchTableにおいて、ContentID及び［(K_1,V_1)，・・・・(K_m,V_m)］の列の値が処理対象の$Item_t$に含まれる値と一致し、かつ、$Energy_{start}$の列の値がＳｔｅｐ４－１で特定した$Energy_{startindex}$と一致する行を探す。

【０１００】

当該行が存在しない場合、処理装置１０は、処理対象の$Item_t$に含まれるContentID、$Energy_{end}$、［(K_1,V_1)，・・・・(K_m,V_m)］、及び、Ｓｔｅｐ４－１で特定した$Energy_{startindex}$を新たな行として図９に示すPatternMatchTableに追加する。

【０１０１】

一方、当該行が存在する場合、処理装置１０は、その行の$Energy_{end}$の値を、処理対象の$Item_t$に含まれる$Energy_{end}$の値に基づき更新する。例えば、バッテリの劣化を考慮し、その行の$Energy_{end}$の値を、最新の値（処理対象の$Item_t$に含まれる$Energy_{end}$の値）に更新してもよい。

【０１０２】

その他、その行の$Energy_{end}$の値を、その行の$Energy_{end}$の値と処理対象の$Item_t$に含まれる$Energy_{end}$の値の統計値（最大値、最小値、平均値等）に更新してもよい。その他、その行の$Energy_{end}$の値を、その行の$Energy_{end}$の値及び処理対象の$Item_t$に含まれる$Energy_{end}$の値各々に各々の重み係数を掛けた値の合計値に更新してもよい。

【０１０３】

「推定フェーズ」

次に、本実施例の推定フェーズを説明する。本実施例は、ユーザがこれから実行するコンテンツを選択するための画面に遷移した時に、その画面に表示される予定のコンテンツごとに、パターンマッチングによってバッテリ消費量を予測する。具体的には、パターンマッチングのためのテーブル（図９）において、$Energy_{end}$のカラムを除いて、推定時の時刻における端末装置のバッテリ残量と、当該時刻におけるハードウエアの構成要素やセンサ情報がマッチする行を探す。

【０１０４】

例えば、ＯＳ００１の端末装置を利用するユーザが、ＮＷ００１の接続方式でネットワークに接続している時に、２つのコンテンツ（００１－ＮＯＲＭＡＬと００１－ＨＡＲＤ）が表示される画面に遷移したとする。この時の端末装置のバッテリ残量が０．７６であったとすると、処理装置１０は、図９に示すPatternMatchTableの中から、ContentIDが「００１－ＮＯＲＭＡＬ」又は「００１－ＨＡＲＤ」にマッチし、K_1（OS）が，「ＯＳ００１」にマッチし、K_2（Network）が「ＮＷ００１」にマッチし、かつ、$Energy_{start}$の値が「０．７６」に最も近い行を探す。

【０１０５】

結果、ContentID「００１－ＮＯＲＭＡＬ」に対応して上から２番目の行が特定され、ContentID「００１－ＨＡＲＤ」に対応して上から５番目の行が特定される。結果、処理装置１０は、「００１－ＮＯＲＭＡＬを実行した場合、バッテリ残量が０．７３になる」と推定し、「００１－ＨＡＲＤをプレイした場合、バッテリ残量が０．６９になる」と推定する。処理装置１０は、この結果を画面に反映する際に、「００１－ＮＯＲＭＡＬを実行した場合、バッテリ残量が７３％になる」、「００１－ＨＡＲＤをプレイした場合、バッテリ残量が６９％になる」のように、ユーザが理解しやすい形式に変換して表示しても良い。

【０１０６】

次に、推論フェーズにおける処理の流れを説明する。

【０１０７】

「Ｓｔｅｐ１」：ユーザが、コンテンツを選択する画面に遷移する入力を行うと、処理装置１０は、当該画面に表示される予定のコンテンツの一覧を、ContentIDの配列として取得する。また、処理装置１０は、端末装置のＯＳから、当該時刻におけるバッテリ残量$Energy_{start}$と、当該時刻におけるハードウエアの構成要素やセンサ情報［(K_1,V_1)，・・・・(K_m,V_m)］を受け取る。

【０１０８】

「Ｓｔｅｐ２」：処理装置１０は、受け取ったContentIDの配列を走査し、ContentIDごとに、各コンテンツを実行した後のバッテリ残量を予測する。具体的には、処理装置１０は、図９に示すPatternMatchTableを用いて、マッチする行を探す。最も単純には、本システムは、Exact-Matchする行がある場合だけ、プレイ後のバッテリ残量を当該行に含まれる$Energy_{end}$であると予測する。そして、Exact-Matchする行がない場合、処理装置１０は、「バッテリ消費量が不明」という結果を出力しても良い。

【０１０９】

例えば、処理装置１０は、ContentIDの配列に含まれるContentIDごとに、以下に示すＳｔｅｐ２－１乃至２－３を実行してもよい。

【０１１０】

「Ｓｔｅｐ２－１」：処理装置１０は、図９に示すPatternMatchTableにおいて、処理対象のContentIDと一致する全ての行を特定する。当該行が１つも存在しない場合、処理装置１０は、処理対象のContentIDで識別されるコンテンツを実行した場合のバッテリ消費量は「不明」という結果を出力する。当該行が１でも存在する場合、処理装置１０は、次のステップの処理を実行する。

【０１１１】

「Ｓｔｅｐ２－２」：処理装置１０は、Ｓｔｅｐ２－１で特定した行の中から、Ｓｔｅｐ１で取得したバッテリ残量$Energy_{start}$が最も近い行を特定する。具体的には、各行が示す$Energy_{start}$とＳｔｅｐ１で取得したバッテリ残量$Energy_{start}$との差の絶対値が最も小さい行を特定する。特定した行が１つである場合、処理装置１０は、処理対象のContentIDで識別されるコンテンツを実行した場合のバッテリ消費量はその特定した行が示す$Energy_{end}$という結果を出力する。一方、特定した行が複数ある場合、処理装置１０は、Ｓｔｅｐ２－３を実行する。

【０１１２】

「Ｓｔｅｐ２－３」：処理装置１０は、Ｓｔｅｐ２－２で特定した行の中から、Ｓｔｅｐ１で取得したハードウエアの構成要素やセンサ情報［(K_1,V_1)，・・・・(K_m,V_m)］が最も近い行を特定する。なお、このｍ次元の情報の近さ（類似度）を算出する手法は特段制限されず、あらゆる手法を採用できる。特定した行が１つである場合、処理装置１０は、処理対象のContentIDで識別されるコンテンツを実行した場合のバッテリ消費量はその特定した行が示す$Energy_{end}$という結果を出力する。一方、特定した行が複数ある場合、処理装置１０は、任意の手法で特定した複数の行の中から１つを特定する。そして、処理装置１０は、処理対象のContentIDで識別されるコンテンツを実行した場合のバッテリ消費量はその特定した行が示す$Energy_{end}$という結果を出力する。

【０１１３】

「Ｓｔｅｐ３」：処理装置１０は、ContentIDごとのバッテリ消費量の推定結果をアプリシステムに渡す。アプリシステムは、受け取った推定結果を反映した画面を描画し、ディスプレイに表示する。この時、アプリシステムは、アプリ実行後のバッテリ残量を見積もった結果をそのまま表示しても良いし、アプリ実行前のバッテリ残量からアプリ実行後のバッテリ残量を引くことで算出したバッテリ消費量を表示してもよい。

【０１１４】

＜作用効果＞

処理装置１０は、端末装置を用いた所定の処理を実行した場合のバッテリ消費量を所定の処理を実行する前にユーザに通知することができる。当該技術によれば、ユーザは、所定の処理を実行した場合のバッテリ消費量を考慮して、当該処理を今実行するか否か等を

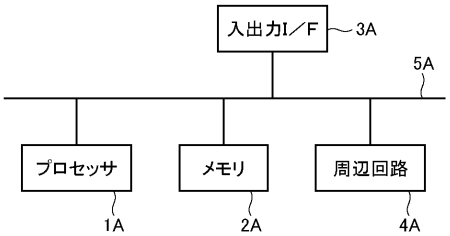
判断することができる。
【０１１５】
　また、処理装置１０は、コンテンツ単位でバッテリ消費量を推定し、その結果をユーザに通知する。コンテンツは、アプリケーション内で実行可能な処理の１つである。このため、１つのコンテンツを実行している間におけるハードウエア処理負担の幅（最も処理負担が大きい時から最も処理負担が小さい時までの幅）は、複数のコンテンツを選択的に順次実行可能な１つのアプリケーションを実行している間におけるハードウエア処理負担の幅よりも小さい。また、１つのコンテンツの実行に要する時間（コンテンツを開始してから終了するまでの時間）の幅は、複数のコンテンツを選択的に順次実行可能な１つのアプリケーションの実行に要する時間の幅よりも小さい。よって、コンテンツ単位でバッテリ消費量を推定する場合、アプリケーション単位でバッテリ消費量を推定するよりも、推定精度が高くなる。結果、推定精度の高い有益な情報をユーザに提供することができる。
【０１１６】
　また、処理装置１０は、例えば何度も繰り返し実行することを前提としたアプリケーションの周回コンテンツを対象に、コンテンツ毎にバッテリ消費量の傾向を学習した推定モデルを用いてバッテリ消費量を推定し、ユーザがコンテンツを選択する時にその推定結果を通知することができる。これにより、ユーザの主体的なバッテリ制御が可能となる。
【０１１７】
　また、処理装置１０は、ハードウエア/ソフトウエアの状況という形でデータを取得することにより、自然に実行環境を考慮することができるため、多様な携帯端末に適用することが可能である。
【０１１８】
　また、処理装置１０は、コンテンツＩＤのみを考慮し、具体的なコンテンツの内容を全く考慮しないので、アプリケーションの開発者は、本実施形態の処理を実現するための特別な設定を追加する必要がない。このため、既存のコンテンツをそのまま利用することができ、かつ、新規コンテンツを従来通りに追加すると、自動的に、処理装置１０にバッテリ消費量を推定させることができる。このように、既存のゲーム開発パイプラインを変更せずに適用可能である。
【０１１９】
　また、処理装置１０は、端末装置側で実現されてもよいし、サーバ側で実現されてもよい。すなわち、ユーザの端末装置で集めたデータを、そのままエッジサイドでの学習に利用しても良いし、サーバに送信してクラウドサイドでの学習に利用しても良い。これにより、開発者は、アプリケーションの特性に応じて、学習のリアルタイム性やユーザプライバシーの保全を重視するならエッジサイドの学習を、端末の負荷低減を重視するならクラウドサイドの学習を、それぞれ選択することができる。このように、処理装置１０は、汎用性が高い。
【０１２０】
　以下、参考形態の例を付記する。
１．　端末装置にインストールされたアプリケーション内のコンテンツを実行した時の前記端末装置の状態を示す実行時状態情報、及び、前記コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、記憶手段に蓄積する情報蓄積部と、
　前記記憶手段に蓄積された前記実行時状態情報及び前記バッテリ情報に基づき、前記端末装置の状態を示す情報から前記コンテンツを実行したときのバッテリ消費量を推定する推定モデルを生成する推定モデル生成部と、
　所定の推定タイミングで前記端末装置の状態を示す推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定する推定部と、
　前記推定部の推定結果を出力する出力部と、
を有する処理装置。
２．　前記アプリケーションでは複数のコンテンツが実行可能であり、

前記情報蓄積部は、前記コンテンツ毎に、前記実行時状態情報及び前記バッテリ情報を前記記憶手段に蓄積し、
前記推定モデル生成部は、前記コンテンツ毎に、前記コンテンツ各々を実行した時のバッテリ消費量を推定する前記推定モデルを生成し、
前記推定部は、前記コンテンツ各々を実行したときのバッテリ消費量を推定する１に記載の処理装置。
３．　前記アプリケーションでは複数のコンテンツが実行可能であり、
前記情報蓄積部は、前記コンテンツ毎に、前記実行時状態情報及び前記バッテリ情報を前記記憶手段に蓄積し、
前記推定モデル生成部は、グループ毎に、少なくとも、前記グループに属する前記コンテンツの前記実行時状態情報及び前記バッテリ情報に基づき前記推定モデルを生成し、
前記推定部は、バッテリ消費量を推定する対象の前記コンテンツである対象コンテンツが属する前記グループを特定し、特定した前記グループに対応する前記推定モデルに基づき、前記対象コンテンツを実行したときのバッテリ消費量を推定する１に記載の処理装置。
４．　前記端末装置が、前記コンテンツの実行を開始する入力を受付ける受付画面を表示する入力を受付けると、
前記推定部は、前記受付画面を表示する入力の受付に応じて前記端末装置の状態を示す前記推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定し、
前記出力部は、前記受付画面において、前記コンテンツに紐づけて前記推定結果を表示させる１から３のいずれかに記載の処理装置。
５．　コンピュータが、
　端末装置にインストールされたアプリケーション内のコンテンツを実行した時の前記端末装置の状態を示す実行時状態情報、及び、前記コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、記憶手段に蓄積し、
少なくとも、前記記憶手段に蓄積された前記実行時状態情報及び前記バッテリ情報に基づき、前記端末装置の状態を示す情報から前記コンテンツを実行したときのバッテリ消費量を推定する推定モデルを生成し、
所定の推定タイミングで前記端末装置の状態を示す推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定し、推定結果を出力する処理方法。
６．　コンピュータを、
　端末装置にインストールされたアプリケーション内のコンテンツを実行した時の前記端末装置の状態を示す実行時状態情報、及び、前記コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、記憶手段に蓄積する情報蓄積手段、
前記記憶手段に蓄積された前記実行時状態情報及び前記バッテリ情報に基づき、前記端末装置の状態を示す情報から前記コンテンツを実行したときのバッテリ消費量を推定する推定モデルを生成する推定モデル生成手段、
所定の推定タイミングで前記端末装置の状態を示す推定時状態情報を取得し、前記推定時状態情報と前記推定モデルとに基づき、前記推定時状態情報で示される状態の前記端末装置で前記コンテンツを実行したときのバッテリ消費量を推定する推定手段、
前記推定手段の推定結果を出力する出力手段、
として機能させるプログラム。
【符号の説明】
【０１２１】
１Ａ　　プロセッサ
２Ａ　　メモリ

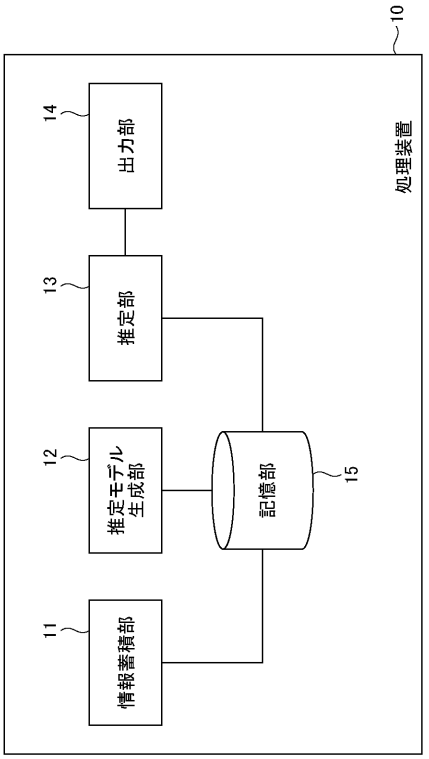
３Ａ　入出力Ｉ／Ｆ
４Ａ　周辺回路
５Ａ　バス
１０　処理装置
１１　情報蓄積部
１２　推定モデル生成部
１３　推定部
１４　出力部
１５　記憶部

【要約】
【課題】端末装置を用いた所定の処理を実行した場合のバッテリ消費量を所定の処理を実行する前にユーザに通知する。
【解決手段】本発明は、端末装置にインストールされたアプリケーション内のコンテンツを実行した時の端末装置の状態を示す実行時状態情報、及び、コンテンツを実行したときのバッテリ消費量を示すバッテリ情報を取得し、記憶部１５に蓄積する情報蓄積部１１と、記憶部１５に蓄積された実行時状態情報及びバッテリ情報に基づき、端末装置の状態を示す情報からコンテンツを実行したときのバッテリ消費量を推定する推定モデルを生成する推定モデル生成部１２と、所定の推定タイミングで端末装置の状態を示す推定時状態情報を取得し、推定時状態情報と推定モデルとに基づき、推定時状態情報で示される状態の端末装置でコンテンツを実行したときのバッテリ消費量を推定する推定部１３と、推定部１３の推定結果を出力する出力部１４と、を有する処理装置１０を提供する。
【選択図】図２

【図１】



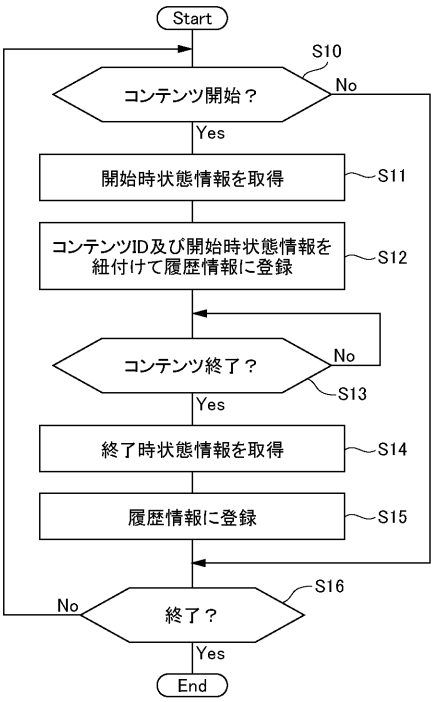
【図２】



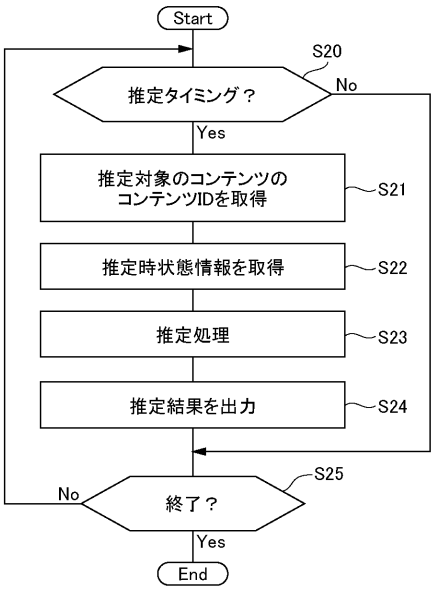
【図３】

ContentID	$Time_{start}$	$Time_{end}$	$Energy_{start}$	$Energy_{end}$	$K_1(OS)$	$K_2(Network)$	...	$K_n(UA)$
001-NORMAL	2020/02/01 12:14:01	2020/02/01 12:21:30	0.79	0.74	OS001	NW001	...	Mobile
001-HARD	2020/02/01 12:23:21	2020/02/01 12:40:18	0.73	0.66	OS001	NW001	...	Mobile
001-HARD	2020/02/01 12:59:15	2020/02/01 13:29:42	0.63	0.55	OS001	NW002	...	Mobile
...	...	...	...	...	...	...	...	...

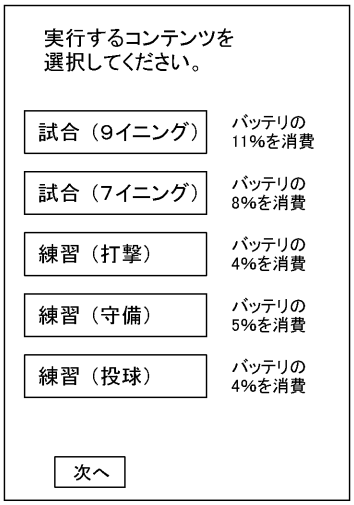
【図４】



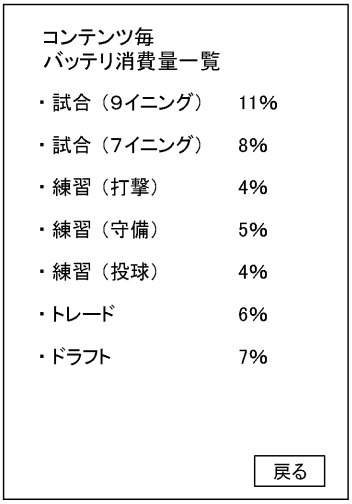
【図５】



【図６】



【図７】



【図８】

Content ID	Group ID
001-NORMAL	G001
001-HARD	G013
・ ・ ・ ・	・ ・ ・ ・

【図９】

ContentID	$Energy_{start}$	$Energy_{end}$	$K_1(OS)$	$K_2(Network)$	…	$K_m(BLE)$
001-NORMAL	0.80	0.75	OS001	NW001	…	OFF
001-NORMAL	0.75	0.73	OS001	NW001	…	OFF
001-NORMAL	0.70	0.67	OS002	NW001	…	ON
001-HARD	0.80	0.72	OS002	NW001	…	ON
001-HARD	0.75	0.69	OS001	NW001	…	OFF
001-HARD	0.70	0.65	OS001	NW001	…	ON
001-HARD	0.80	0.71	OS003	NW002	…	OFF
001-HARD	0.75	0.63	OS001	NW002	…	OFF
001-HARD	0.70	0.62	OS002	NW002	…	ON
…	…	…	…	…	…	…

フロントページの続き

(56)参考文献 特開２０１３－２２８９０２（ＪＰ，Ａ）
特開２０２０－０２８１９６（ＪＰ，Ａ）
特開２０１２－０６８８０３（ＪＰ，Ａ）
米国特許出願公開第２０１７／０２７９６９７（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ａ６３Ｆ　１３／００ － １３／９８
Ｇ０６Ｆ　１／２５ － １／３２９６