

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第4613817号

(P4613817)

(45) 発行日 平成23年1月19日 (2011. 1. 19)

(24) 登録日 平成22年10月29日 (2010. 10. 29)

(51) Int. Cl.

F I

G 1 O G 1/02 (2006. 01)

G 1 O G 1/02

G O 9 B 15/00 (2006. 01)

G O 9 B 15/00 B

G 1 O H 1/00 (2006. 01)

G 1 O H 1/00 Z

G 1 O H 1/00 1 O 2 Z

請求項の数 3 (全 21 頁)

(21) 出願番号 特願2005-376694 (P2005-376694)
 (22) 出願日 平成17年12月27日 (2005. 12. 27)
 (65) 公開番号 特開2007-178695 (P2007-178695A)
 (43) 公開日 平成19年7月12日 (2007. 7. 12)
 審査請求日 平成20年5月23日 (2008. 5. 23)

(73) 特許権者 000001443
 カシオ計算機株式会社
 東京都渋谷区本町 1 丁目 6 番 2 号
 (74) 代理人 100074099
 弁理士 大菅 義之
 (74) 代理人 100133570
 弁理士 ▲徳▼永 民雄
 (72) 発明者 副島 淳一郎
 東京都羽村市栄町 3 丁目 2 番 1 号 カシオ
 計算機株式会社羽村技術センター内

審査官 鈴木 聡一郎

最終頁に続く

(54) 【発明の名称】 運指表示装置、及びプログラム

(57) 【特許請求の範囲】

【請求項 1】

楽器に設けられた演奏操作子群のなかで操作していきべき演奏操作子の操作に用いるべき指を示す指情報を表示させることにより、楽曲の演奏のための指の運びである運指を教示できる運指表示装置において、

前記楽曲の進行に応じて順次操作していきべき演奏操作子、及び該演奏操作子の操作に用いるべき指を示す運指データを取得するデータ取得手段と、

前記データ取得手段が取得した運指データの運指評価を行い難易度を算出する難易度算出手段と、

該難易度算出手段により算出された難易度が、ユーザが設定した所定内である運指データを抽出して該運指データによる指情報を表示させる表示制御手段と、

を具備することを特徴とする運指表示装置。

【請求項 2】

前記表示制御手段は、前記運指データから前記演奏操作子への操作によって発音される楽音を示す音符を表示させ、前記指情報は、対応する音符の近傍に表示させる、ことを特徴とする請求項 1 記載の運指表示装置。

【請求項 3】

楽器に設けられた演奏操作子群のなかで操作していきべき演奏操作子の操作に用いるべき指を示す指情報を表示させることにより、楽曲の演奏のための指の運びである運指を教示できる運指表示装置として用いられるコンピュータに、

10

20

前記楽曲の進行に応じて操作していくべき演奏操作子毎に、該演奏操作子、及び該演奏操作子の操作に用いるべき指を示す運指データを取得する機能と、

前記取得する機能により取得した運指データの運指評価を行い難易度を算出する機能と

、該難易度を算出する機能により算出された難易度が、ユーザが設定した所定内である運指データを抽出して該運指データによる指情報を表示装置に表示させる機能と、

を実現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、楽器に設けられた演奏操作子群のなかで操作していくべき演奏操作子の操作に用いるべき指を示す指情報を表示させることにより、楽曲の演奏のための指の運びである運指を教示するための技術に関する。

【背景技術】

【0002】

楽器による楽曲の演奏は、楽器に設けられた演奏操作子群、例えば鍵盤を操作していくことで行われる。しかし、演奏技術の未熟な演奏者（ユーザー）の多くは、演奏の進行に応じて操作（押鍵）していくべき鍵をどの指で操作すべきかが判らないのが実情である。このことから、鍵の押鍵に用いるべき指を示す指情報を表示できる運指表示装置が製品化されている。従来の運指表示装置としては、例えば特許文献1に記載されたものがある。

【0003】

特許文献1に記載された運指表示装置を含め、従来の運指表示装置では、鍵（楽音）毎に指情報を表示して、その鍵の押鍵に用いるべき指をユーザーに通知するようになっていた。その指情報の表示は、例えば特許文献2に記載された技術を用いて行うこともできる。しかし、そのように鍵毎に指情報を表示する場合、ユーザーの演奏技術、或いは演奏する楽曲の内容などによっては、指情報の確認に要するユーザーの負担が大きくなりすぎると考えられる。その確認が負担なユーザーにとっては、指情報の確認のために、演奏への集中が阻害されるという面が無視できなくなると言える。このようなことから、ユーザーにとって重要性が高いと思われる指情報（運指データ）を選択的に表示させることも重要と考えられる。

【特許文献1】特開2001-282094号公報

【特許文献2】特開2002-318579号公報

【発明の開示】

【発明が解決しようとする課題】

【0004】

本発明の課題は、ユーザーにとって重要性が高い（有益）と思われる指情報（運指データ）を選択的に表示させるための技術を提供することにある。

【課題を解決するための手段】

【0005】

本発明の運指表示装置は、楽器に設けられた演奏操作子群のなかで操作していくべき演奏操作子の操作に用いるべき指を示す指情報を表示させることにより、楽曲の演奏のための指の運びである運指を教示できる運指表示装置において、

前記楽曲の進行に応じて順次操作していくべき演奏操作子、及び該演奏操作子の操作に用いるべき指を示す運指データを取得するデータ取得手段と、

前記データ取得手段が取得した運指データの運指評価を行い難易度を算出する難易度算出手段と、

該難易度算出手段により算出された難易度が、ユーザが設定した所定内である運指データを抽出して該運指データによる指情報を表示させる表示制御手段と、
を具備することを特徴とする。

【0007】

本発明のプログラムは、楽器に設けられた演奏操作子群のなかで操作していくべき演奏操作子の操作に用いるべき指を示す指情報を表示させることにより、楽曲の演奏のための指の運びである運指を教示できる運指表示装置を実現可能なデータ処理装置に実行させることを前提とし、上記運指表示装置が具備する手段を実現させるための機能を搭載している。

【発明の効果】

【0008】

本発明は、楽曲の進行に応じて操作していくべき演奏操作子毎に、その演奏操作子、及びその演奏操作子の操作に用いるべき指を示す運指データを取得し、取得した運指データのなかから予め設定された条件を満たす運指データを抽出して、その運指データが操作に用いるべきとする指を示す指情報を選択的に表示させる。

10

【0009】

上記条件によって運指データを選択・抽出することにより、ユーザーにとって、或いは演奏上、重要性の高い運指、困難と思われる運指をユーザーは確認できるようになる。指情報は選択的に表示されるために、その確認に要する負担は全ての指情報を表示させる場合と比較して軽減され、負担の軽減によって、表示された指情報に従った運指はより容易となる。指情報が表示されない演奏期間であっても、直前に表示された指情報によって適切な手指の状態から演奏を行えるために、適切な運指を行える可能性は高くなる。これらのことから、指情報を選択的に表示させたとしても、ユーザーは適切な運指による演奏を高い確率で行えることとなる。

20

【発明を実施するための最良の形態】

【0010】

以下、本発明の実施の形態について、図面を参照しながら詳細に説明する。

図1は、本実施の形態による運指表示装置の構成を説明する図である。

その運指表示装置10は、図1に示すように、装置10全体の制御を行うCPU11と、そのCPU11がワークに用いるRAM12と、CPU11が実行するプログラムや各種制御用データ等を格納したROM103と、例えばキーボードやポインティングデバイス（マウス等）等の入力装置用のインターフェース、CD-ROMやDVD等の記録媒体にアクセスする媒体駆動装置、及びそのインターフェース等からなる入力部14と、及び表示装置に画像を表示させる表示部15と、を備えた構成となっている。

30

【0011】

その運指表示装置10は、例えばパーソナルコンピュータ（PC）に、運指表示装置10として動作させるためのアプリケーション・プログラム（以下「運指表示アプリケーション」）を搭載させることにより実現させている。そのアプリケーションは、例えば入力部14を構成する媒体駆動装置がアクセス可能な記録媒体に記録して提供されるものである。そのアプリケーションは、インターネット等のネットワークを介して配信するようにしても良い。アプリケーションを搭載（インストール）できるように、ROM13としては書き込み可能なメモリ（例えばフラッシュメモリ）が採用されている。運指表示装置10を実現させるPCは、ROM13の他に、或いはその代わりにハードディスク装置等の補助記憶装置を搭載したものであっても良い。表示部15が画像を表示させる表示装置は、外付けのもの、搭載されたものの何れであっても良い。ここでは便宜的に、表示装置は搭載されたもの、つまり表示部15を構成しているものと想定する。

40

【0012】

本実施の形態では、楽曲の演奏で発音させるべき楽音毎に、その楽音を発音させるべきタイミング、及びその楽音の音高を示す演奏データを取得し、その演奏データから楽譜（図14～図17）を表示部15に表示させるようにしている。その楽音を発音させるために用いるべき指を示す運指データは、必要に応じて演奏データから生成し、その演奏データに追加するようにしている。それにより、最終的に図2に示すような演奏データを取得して、運指データを表示可能とさせている。図14～図17において、楽譜の近傍に表記した「1」～「5」の数字は、運指データが示す指に割り当てた指番号（指情報）である

50

。その指番号を対応する音符の近傍に配置して表示することにより、楽曲の進行により発音させるべき楽音と併せて運指をユーザーに教示するようにしている。図2に示す演奏データはRAM12に書き込まれた場合のものであり、その格納場所は、[]で示す括弧内の数値（インデクス値）で指定される。

【0013】

その演奏データは、図2に示すように、データlTime、lGate、lTmsec、Pitch、iPosX、DrawPos、cfg、cDisp、iHand、icost、icostN、cIsHarm、iArpS、iArpE、pHTop、pHTail、cTendency、cTendSteps、cIsSO、pPFTop、pPHTail、pNHTop、pNHTail、iNP、prev、及びnextを有している。各データは以下のようなものである。

【0014】

データlTimeは、発音開始時刻を示すものである。その発音開始時刻は例えば演奏開始からのものをMIDIで扱う単位時間（ステップ）数で表されている。データlGateは発音時間を示すものである。データlTmsecはデータlTimeが示す発音開始時刻を絶対時間（単位はmsec）で示すものである。データPitchは楽音のピッチを示すもの（ピッチ番号、或いはノート番号と呼ぶ）である。データiPosXはそのピッチが割り当てられた鍵の鍵盤上の座標（X座標）を示すものである。データDrawPosは、その鍵の押鍵に用いるべき指の指番号を表示部15に表示させる際の座標を示すデータである。データcfgは、その鍵の押鍵に割り当てられた指を示す指番号である。値の0は親指を示し、同様に、1は人差し指、2は中指、3は薬指、4は小指をそれぞれ示している。データcDispは、指番号の表示・非表示を示すフラグである。値の0は非表示、1は表示を示している。データiHandは鍵を押鍵して発音される楽音が属するパートを示すものである。そのパートは押鍵に使う指がある手で区別されるものであり、その手が右手のパートでは1、左手では2がデータiHandとして格納される。データicostは、割り当てられた指で鍵を押鍵する難易度の評価値（難易度データ）である。データicostNは、難易度を正規化した評価値である。データcIsHarmは楽音が和音の構成音か否かを示すものである。その値は構成音でなければ0、構成音であれば-1となる。

【0015】

データiArpSは、アルペジオを構成する先頭の楽音か否かを示す先頭フラグである。データiArpEは、その末端（最後）の楽音か否かを示す末端フラグである。それらの値は、該当するか否かによって、予め定めた2種類の値のうちの何れかとなる。データpHTopは、和音の先頭音（一番下の音（最低音））の演奏データの格納場所を示すポインタ（インデクス値）である。データpHTailは、和音の終端音（一番上の音（最高音））の演奏データの格納場所を示すポインタ（インデクス値）である。それらのインデクス値は、楽音が和音の構成音であった場合に格納される。そうでなければ予め定めた固定値が格納される。

【0016】

データcTendencyは、楽音近傍の楽音の並び（音列）の高低傾向を示す値（0は極値、正の値は上昇傾向、負の値は下降傾向、1は変化しない踊り場、2はそれ以外（の変化あり））である。データcTendStepsは極値となるまでの段階数である。データcIsSOは、ポリフォニーとなっているか否かを示すフラグである。その値の絶対値は音数を示している。正の値は前に同時発音中の楽音の存在を示し、負の値は後に同時発音中の楽音の存在を示している。データpPHTopは前の音列を構成する和音の先頭音の演奏データのポインタである。データpPHTailはその和音の終端音の演奏データのポインタである。データpNHTop、pNHTailはそれぞれ、次の音列を構成する和音の先頭音、終端音の演奏データのポインタである。データiNPは割り当てた指を押鍵に使うべきでない禁止条件に該当するか否かを示す値である（0は該当しないことを示す）。データprevは前の演奏データのポインタである。データnextは次の

10

20

30

40

50

演奏データのポインタである。

【0017】

上記データ *lTime*、*lGate*、*lTmsec*、及び *Pitch* は発音させるべき楽音、及びそれを発音させるべき期間（タイミング）を示すデータであり、本来の演奏データ（以下、混乱を避けるために「実演奏データ」と呼ぶ）を構成する。それら、並びにデータ *cf ig*、及び *iHand* は運指データを構成するものである。それらのなかで、データ *cf ig*、及び *iHand* は最小限の運指データ（以下、混乱を避けるために「最小運指データ」と呼ぶ）を構成する。それら、並びにデータ *prev*、及び *next* を除く他のデータは、その最小運指データ生成用のものである。

【0018】

図2に示すような構成の演奏データ、実演奏データ、或いは運指データは例えば、入力部14を構成する媒体駆動装置がアクセス可能な記録媒体により、或いは不図示の通信ネットワークを介して運指表示装置10に入力される。ここでは記録媒体により入力されるものと想定する。

【0019】

本実施の形態では、上記演奏データが実演奏データ分のみで構成されている場合に対応するために、運指データを生成する機能を搭載させている。それにより、実演奏データ分のみで構成されている演奏データを入力したとしても、演奏データ毎に運指データを生成してそれを表示できるようにしている。

【0020】

図14～図17に示す楽譜は、演奏（実演奏）データ毎に、それが表す音符の種類、それを配置すべき位置を決定し、五線譜上に配置して表示させるものである。音符の種類はデータ *lGate* から特定でき、Y座標はデータ *Pitch* から特定でき、X座標はデータ *lTime* から特定できる。休符は、音符（楽音）の発音終了時刻（データ *lTime*、*lGate* によって示される）から次の音符の発音開始時刻（データ *lTime*）の時間間隔（無音期間）の長さから特定することができる。

【0021】

上述したように、図14～図17に示す楽譜の近傍に表記した「1」～「5」の数字は、運指データが示す指に割り当てた指番号（指情報）である。上記データ *cDisp* は、その指番号を表示させるか否か管理するために用意したデータであり、その値として1が格納された演奏データのみを対象に、それを構成するデータ *cf ig* を表示させるようにしている。図14～図17にそれぞれ示した運指表示例は、楽譜上、休符の直後に位置する音符の発音に用いるべき指の指番号を表示させた場合、指越え・指くぐりなどの特殊な指使いをして音符を発音させるべき指の指番号を表示させた場合、音符を発音させる難易度によりその発音に用いるべき指の指番号を表示させた場合、及び正規化した難易度により音符の発音に用いるべき指の指番号を表示させた場合のものである。

【0022】

このようにして本実施の形態では、休符の直後に位置する音符、特殊な指使いをして発音させるべき音符、或いは発音させる難易度が予め設定された範囲内の音符の少なくとも一方に相当する音符の発音に用いるべき指の指番号を選択的に表示させるようにしている。それにより、指番号の確認をより容易に行えるようにして、その確認のために要する負担を軽減させ、演奏により集中できるようにさせている。

【0023】

表示させる指番号の選択に休符、特殊な指使い、及び難易度を考慮したのは以下の理由からである。

休符は、演奏操作の切れ目に相当する。その休符箇所では、手を移動させる、或いは各指を余裕のある状態に戻す、といった演奏を続けるうえで動作のポイントとなる場合が多い。休符の直後の押鍵に用いるべき指を知らせると、ユーザーはその指でその押鍵を行えるように手、或いは指を動かすようになる。そのような動作を行うことにより、手、或いは指は演奏を行ううえでより理想的な状態に近づく結果となる。このようなことから、休

10

20

30

40

50

符後の押鍵に用いるべき指を教示することはユーザーにとって非常に有益と考えられる。このため、休符に着目している。

【0024】

特殊な指使いは、難易度が高いだけでなく、それを行ううえでの制約もある。それにより、休符と同様に、演奏を続けるうえで動作のポイントとなる場合が多い。このことから、特殊な指使いによる押鍵に用いるべき指を教示することはユーザーにとって非常に有益と考えられる。このため、特殊な指使いに着目している。

【0025】

一般的に、難易度（データ *i c o s t*）が高い押鍵ほど、その押鍵を適切に行うのは困難となる。このことから、困難と思われる押鍵、及びその押鍵に用いるべき指といったものをユーザーが確認できるようにさせている。ユーザーによって困難、或いは押鍵に用いるべき指を確認したいと感じる難易度は異なることから、指番号を表示させる難易度の範囲はユーザーが任意に指定できるようにさせている（図16）。

10

【0026】

演奏する楽曲によって平均的な難易度が異なる。ユーザーにとって高いと思われる難易度であっても、楽曲全体では低いほうに位置していることもある。このことから本実施の形態では、正規化した難易度により指番号を表示できるようにしている。それにより、演奏する楽曲での相対的な評価による難易度（データ *i c o s t N*）で指番号を表示させることができるようにさせている。

【0027】

20

上述したようにして選択・表示される指番号は、演奏を行ううえで非常に重要なポイントとなる押鍵、或いはユーザーにとって確認を望む押鍵を適切に行えるように支援する情報となる。このため、そのような指番号のみを選択して表示させたとしても、ユーザーは表示された指番号が示す指で押鍵すべき鍵を適切に押鍵することにより、楽曲全体の演奏も高い確率で適切な運指を行えるようになる。従って、表示させない指番号が存在することの影響は実質的でないか、或いは非常に小さくなる。

【0028】

上記運指（最小運指）データは、運指がより容易となるように生成される。そのために、運指（押鍵すべき鍵の押鍵に使う指をその鍵に運ぶ動作）の難易度を計算している。本実施の形態では、そのこともあって、難易度を考慮した指番号の選択を可能とさせている。最終的に求められた難易度を示す値は上記データ *i c o s t* として格納され、正規化された難易度を示す値は上記データ *i c o s t N* として格納される。それらの値は以降、コスト値とも呼ぶことにする。

30

【0029】

以降は図3～図13に示す各種フローチャートを参照して、上述したように指番号を表示させる対象とする演奏データを選択して指番号を図14～図17に示すように表示させる運指表示装置10の動作について詳細に説明する。その動作は、CPU13が、上記運指表示アプリケーションをROM13からRAM12に読み出して実行することで実現される。

【0030】

40

図3は、全体処理のフローチャートである。これは、上記運指表示アプリケーションを起動させてから終了させるまでの間に実行される処理を抜粋してその流れを示したものである。始めに図3を参照して、その全体処理について詳細に説明する。

【0031】

まず、ステップ101では、各種変数に初期値を代入するといった初期化を行う。次のステップ102では、入力部14を構成する媒体駆動装置に、それに装着された記録媒体にアクセスさせ、その記録媒体に格納された演奏データ（図2）の読み込みを行う。特に詳細な説明は省略するが、その読み込みは、入力部14を構成する入力装置、或いはそれと接続された入力装置への操作によってユーザーが指定した演奏データを対象に行う。その読み込みによって演奏データをRAM12に格納した後はステップ103に移行する

50

。

【0032】

ステップ103では、図6に示すコスト値算出用の運指コストテーブルをRAM12に展開し、準備する。続くステップ104では、演奏データに格納された運指データを評価する運指評価処理を実行する。その評価処理の実行により、演奏データが実演奏データ分のみで構成されている場合には、運指データが生成される。それにより、実行後の演奏データは図2に示すような構成のものとなる。

【0033】

ステップ104に続くステップ105では、演奏データを対象に、指番号の表示・非表示を設定する表示・非表示設定処理を実行する。次のステップ106では、その設定結果に従って、楽譜、及び指番号を表示させる描画処理を実行する。その描画処理の実行により、図14～図17に示すように指番号を楽譜と共に表示部15に表示させた後、一連の処理を終了する。

10

【0034】

以降は、上記全体処理内で実行されるサブルーチン処理について詳細に説明する。

図4は、上記ステップ104として実行される運指評価処理のフローチャートである。次に図4を参照して、その評価処理について詳細に説明する。

【0035】

まず、ステップ201では、運指評価に用いる各種変数の初期化等を行う。続くステップ202では、運指データを演奏データ中に有する楽曲データを取得するためのデータ取得処理を実行する。入力部14を介して入力した楽曲データを構成する演奏データに運指データが含まれていない場合、その運指データを演奏データ毎に生成する処理を行う。その生成は周知の技術を用いて行う。

20

【0036】

ステップ202に続くステップ203では、変数meに、楽曲データ中で先頭に位置する演奏データを指定するインデクス値（ポインタ）を代入する。次のステップ204では、運指評価が終了したか否か判定する。その評価は、変数meの値で指定される演奏データを先頭のものから最後のものまで順次、変更しながら行われる。このため、変数meの値で指定される演奏データが存在しない場合、判定はYESとなり、ここで一連の処理を終了する。そうでない場合には、判定はNOとなってステップ205に移行する。

30

【0037】

ステップ205では、変数meの値で指定される演奏データが示す楽音（指定音）とその次に発音すべき楽音間のコストを算出する2音間コスト算出処理を実行する。次に移行するステップ206では、現在、注目している運指（指定音を発音させるための運指）が禁止条件とする事項の何れかに該当しているか否か確認するための禁止事項検索処理を実行する。その後はステップ207に移行する。

【0038】

ステップ207では、変数meの値で指定される演奏データ中のデータiNP（図中「me．iNP」と表記。以降、他のデータも同様にこの表記法を用いる）の値が0でないか否か判定する。上記禁止事項検索処理では、該当する事項（禁止条件）に該当していることが判明した場合、その値として0以外の値を格納する。このことから、その場合、判定はYESとなり、ステップ212でme．icostとして、想定範囲内におけるコストの最大値を格納した後、ステップ209に移行する。そうでない場合には、判定はNOとなってステップ208に移行し、運指に従って指定音を発音させるためのコストをより適切なものに更新するためのコスト補正処理を実行する。ステップ209にはその後に移行する。

40

【0039】

ステップ209では、me．icostの値、つまり指定音のコストの値が、閾値として定めた定数THRCOST（<コストの最大値）より大きいと否か判定する。その値が定数THRCOSTより大きい場合、判定はYESとなり、運指を修正するための運指修

50

正処理を実行し（ステップ210）、その実行後に変数`me`に`me. next`の値（次の演奏データを指定するインデックス値）を代入してから（ステップ211）、上記ステップ204に戻る。一方、そうでない場合には、判定はYESとなり、次にそのステップ211に移行する。

【0040】

上記運指修正処理での運指の修正は、特に詳細な説明は省略するが、例えば指定音の前後の部分を考慮して処理対象とする区間を決定し、決定した区間内の演奏を行ううえで可能性のある全ての運指の組み合わせのなかからコストが最小となる組み合わせを抽出することで行っている。それにより、指定音に対して最適な運指を修正時には確実に設定させている。この結果、評価に用いる運指データは全て、それが示す運指が可能、且つそのコストが異常に大きくないものとなる。

10

【0041】

ここで、上記ステップ205、206、及び208で実行されるサブルーチン処理について、図5～図8を参照して詳細に説明する。

図5は、上記ステップ205として実行される2音間コスト算出処理のフローチャートである。始めに図5を参照して、そのコスト算出処理について詳細に説明する。

【0042】

このコスト算出処理では、連続する2音にのみ着目する形でその2音を発音させるための運指のコストを算出するようになっている。そのコストの算出は、図6に示す運指コストテーブルを参照し、対応する値を抽出することで行っている。コスト算出処理について説明する前に、そのコストテーブルについて具体的に説明する。

20

【0043】

図6中、「`bw`」は鍵盤種類（0は白鍵、1は黒鍵）、「`pm`」はピッチ変化方向（0は増加方向、1は減少方向）、「`f1`」は変数`ev1`の値で指定される音符の指番号（その値で指定される演奏データ中のデータ`cfi g`の値）、「`f2`」は変数`ev2`の値で指定される音符の指番号、「`int v`」はそれら2つの音符間のピッチ差、をそれぞれ表している。それにより、テーブルは、それらのデータの各値の組み合わせで示される状況別に、その状況下での運指のコスト（の値）を格納したのとなっている。このことから、2音間コスト算出処理では、それらのデータの各値を求め、各値の組み合わせで指定される欄に格納されたコスト（値）を抽出し、それを指定音の運指のコストとして更新するようになっている。運指コストテーブルは、5個の値によって示される状況別にコストの値を格納したものであることから、図5中では5次元の配列変数`CT`として「`CT [bw] [pm] [f1] [f2] [int v]`」と表記している。

30

【0044】

まず、ステップ301では、変数`ev1`、及び`ev2`にそれぞれ、変数`me`の値、及び`me. next`の値を代入する。続くステップ302では、変数`f1`に`ev1. cfi g`の値、変数`p1`に`ev1. iPos X`の値、変数`bw`に変数`p1`の値を2で割った余り（図中「`p1 % 2`」と表記）、変数`f2`に`ev2. cfi g`の値、変数`p2`に`ev2. iPos X`の値、をそれぞれ代入する。その代入後に移行するステップ303では、変数`p1`の値が変数`p2`の値以下か否か判定する。後者が前者より大きい場合、判定はYESとなり、ステップ304で0を変数`pm`に、変数`p2`の値から変数`p1`の値を引いた値を変数`int v`にそれぞれ代入してからステップ306に移行する。そうでない場合には、判定はNOとなり、ステップ305で1を変数`pm`に、変数`p1`の値から変数`p2`の値を引いた値を変数`int v`にそれぞれ代入してからステップ306に移行する。

40

【0045】

ステップ306では、変数`int v`の値が定数`MAXINTV`以下か否か判定する。その定数`MAXINTV`は、運指により押鍵を行う難易度が極めて高いと考えられる鍵間（ピッチ差）を判定するために用意したものである。このことから、着目する2つの音符の間にそのようなピッチ差が存在していない場合、判定はYESとなり、ステップ307で`ev1. icost`として、変数`bw`、`pm`、`f1`、`f2`、及び`int v`の各値によって

50

運指コストテーブル（図6）から抽出される値（図中「CT [bw] [pm] [f1] [f2] [intv]」）と表記）を代入した後、一連の処理を終了する。そうでない場合には、判定はNOとなり、ステップ308でevl. icostとして定数MAXCOSTを代入した後、一連の処理を終了する。

【0046】

上記定数MAXCOSTは、例えばコストとして想定する範囲内で最大値として定めたものである。図4中のステップ212では最大値としてこの定数MAXCOSTにme. icostを更新するようになっている。

【0047】

上述したような構成の運指コストテーブルでは、2音間の発音開始時刻の時間間隔が考慮されていない。その時間間隔が短くなるほど、運指は困難となるのが普通である。このことから、上記ステップ307では、特には図示していないが、evl. icostとして、変数bw、pm、f1、f2、及びintvの各値によって運指コストテーブルから値を抽出した後、抽出値に時間間隔に応じた係数を乗算し、その乗算結果をevl. icostとして格納している。

【0048】

図7は、図4に示す運指評価処理内でステップ206として実行される禁止事項検索処理のフローチャートである。次に図7を参照して、その検索処理について詳細に説明する。

【0049】

例えば親指や小指はその長さ、手における位置などの理由により、鍵盤上の黒鍵の押鍵はそれ以外の指よりも困難である。同じ指を使った連続する押鍵は、離鍵から押鍵までの時間間隔、押鍵する鍵間の距離（音高（ピッチ）差）などに応じて難易度（コスト）が変化する。時間間隔が短く、且つ鍵間の距離が大きいような場合、その押鍵は不可能か、或いは極めて困難となる。他にも不可能、或いは極めて困難な運指は存在する。このようなことから、検索処理では、指定音に割り当てられた運指が不可能、或いは極めて困難と考えられる条件（禁止事項）毎に、それを満たしているか否か検索により確認するようになっている。

【0050】

先ず、ステップ401では、親指や小指といった比較的短い指で演奏者から見て離れた場所にある黒鍵を押鍵するような運指を禁止条件に該当しているとして検出するための黒鍵の指使い検出処理を実行する。次のステップ402では、和音を発音させる場合に、或る指で押鍵する鍵より高音の鍵をその指より低音側に位置する指で押鍵する、或いはその逆に押鍵するといったことを禁止事項として検出するための和音での指の逆転検出処理を実行する。その後はステップ403に移行する。

【0051】

ステップ403では、複数の鍵を押鍵する状況下で、同時に押鍵していることが不可能と考えられる指の割り当てを禁止条件として検出するための指の開き幅の限界検出処理を実行する。続くステップ404では、同時に押鍵していることが不可能な数の押鍵を禁止条件として検出するための多声条件検出処理を実行する。その実行後、一連の処理を終了する。

【0052】

上記ステップ401～4の各検出処理では、禁止条件を満たしていることが確認された場合、me. iNPとして0以外の値を格納（更新）する。それにより、図4に示す運指評価処理では、ステップ207の判定がYESとなって、その指定音のコストは最大値に更新されることとなる。

【0053】

図8は、図4に示す運指評価処理内でステップ208として実行されるコスト補正処理のフローチャートである。その運指評価処理内で実行されるサブルーチン処理では、最後に図8を参照してその補正処理について詳細に説明する。

【0054】

先ず、ステップ501では、運指に従って指定音を発音させる際の状況を示す各種値を対応する変数に代入するための処理を実行する。次のステップ502では、変数meの値で指定される演奏データが示す楽音（指定音）の発音開始時刻に発音中の他の楽音、或いはその発音終了時刻前に発音を開始する他の楽音に着目してコストを補正するための多声音の補正処理を実行する。続くステップ503では、指定音の押鍵に必要な、指越え、或いは指くぐりという比較的に高度な演奏方法に着目してコストを補正するための指越え／指くぐりでの補正処理を実行する。その次のステップ504では、分散和音とそれに続く分散和音間の相互関係に着目してコストを補正するための分散和音での補正処理を実行する。その後はステップ505に移行する。

10

【0055】

ステップ505では、単音が続く音符群とそれに続く和音間の相互関係に着目してコストを補正するための短音→和音遷移での補正処理を実行する。続くステップ506では、和音とそれに続く単音の音符群間の相互関係に着目してコストを補正するための和音→短音遷移での補正処理を実行する。その次のステップ507では、和音とそれに続く和音間の相互関係に着目してコストを補正するための和音→和音遷移での補正処理を実行する。その次に移行するステップ508では、和音とその直前に位置する和音間の相互関係に着目してコストを補正するための前方和音補正処理を実行する。一連の処理はその後に終了する。

【0056】

20

このようにして、コスト補正処理では、互いに異なる内容毎に、その内容に着目してコストを必要に応じて補正するための処理を実行するようになっている。それにより、コスト補正処理を実行する前の指定音のコスト（me. icost）の値は、該当する内容の有無、その数に応じて補正される（以上のようなコストの算出についての参考技術文献としては、本出願人が出願した特願2004-261020号が挙げられる）。

【0057】

図4に示す全体処理内で実行されるサブルーチン処理の説明に戻る。

上述したように本実施の形態では、表示させる指番号の選択は、休符、特殊な指使い、及び難易度を条件にして行うようにしている。実際の選択に用いる条件の種類は、そのなかから一つをユーザーに指定させるようにしている。その指定は、図4に示す全体処理内でステップ105として表示・非表示設定処理の実行時に問い合わせ、ユーザーに行わせている。その指定結果により、設定処理では図9～図12に示す各サブルーチン処理の少なくとも一つが実行される。このことから、図9～図12を参照して、ユーザーが指定した条件の種類別に、設定処理内で実行されるサブルーチン処理について詳細に説明する。

30

【0058】

図9は、休符箇所を考慮した表示・非表示設定処理のフローチャートである。その設定処理は、ユーザーが条件として休符を指定した場合に実行される。始めに図9を参照して、その設定処理について詳細に説明する。

【0059】

先ず、ステップ601では、変数meに、先頭に位置する演奏データのインデクス値を代入する。その代入後に移行するステップ602では、変数meの値で指定される演奏データが最後に位置するものではないか否か判定する。その演奏データが最後に位置するものであった場合、判定はNOとなり、ここで一連の処理を終了する。そうでない場合には、判定はYESとなってステップ603に移行する。

40

【0060】

ステップ603では、me. cDispの値として1を格納して、表示設定を行う。次のステップ604では、変数mepに、現在、対象としている演奏データの直前に位置する演奏データのインデクス値（me. prev）を代入する。その後に移行するステップ605では、変数mepの値で指定される演奏データが存在するか否か判定する。現在、対象としている演奏データが先頭に位置するものであった場合、その前の演奏データは存

50

在しないことから、判定はNOとなってステップ611に移行する。そうでない場合には、判定はYESとなってステップ606に移行する。

【0061】

ステップ606では、変数lRestTに、me.lTimeの値からmep.iTimeの値を減算し、その減算結果にmep.lGateの値を加算して得られる値(=me.lTime-mep.iTime+mep.lGate:直前の鍵を離鍵すべきタイミングとなってから次の鍵を押鍵すべきタイミングとなるまでの時間間隔を示す値)を代入する。また、変数lStepTに、me.lTimeの値からmep.iTimeの値を減算した値(=me.lTime-mep.iTime:直前の鍵を押鍵すべきタイミングとなってから次の鍵を押鍵すべきタイミングとなるまでの時間間隔を示す値)を代入する。それらの代入を行った後はステップ607に移行して、変数lRestTの値は定数iResの値の2倍以上か否か判定する。定数iResは、一定以上、無音期間が続く箇所を、指番号の表示の対象となる休符箇所として検出するために用意したものである。例えば4分音符相当の時間を示す値である。このことから、一定以上、無音期間が続く箇所であった場合、判定はYESとなってステップ611に移行する。そうでない場合には、判定はNOとなり、ステップ608でme.iDispとして0を格納し非表示設定を行う。その後はステップ609に移行する。

10

【0062】

ステップ609では、変数lStepTの値は定数iResの値の2倍以上であり、且つ変数lRestTの値は定数iResの1/4以上か否か判定する。それら何れの大小関係が満たされている場合、判定はYESとなり、ステップ610でme.iDispに1を格納した後、ステップ611に移行する。そうでない場合には、つまりそれらの大小関係の少なくとも一つが満たされていない場合には、判定はNOとなってステップ611に移行する。

20

【0063】

このようにして本実施の形態では、一定以上、長く押鍵すべき鍵は指番号を表示する対象としている。これは、そのような鍵を押鍵すべき指で押鍵することで、それ以降の演奏をより適切な運指で行えるようになるからである。つまりそのような鍵は、休符直後に押鍵すべき鍵と演奏上の性格に類似点が多いためである。

【0064】

ステップ611では、変数meの値がme.pHTailと一致するか否か、つまり対象としている演奏データが和音の最も音高の高い構成音のものか否か判定する。その構成音のものであった場合、判定はYESとなってステップ612に移行する。そうでない場合には、判定はNOとなってステップ617に移行し、変数meにme.nextの値を代入した後、上記ステップ602に戻る。

30

【0065】

ステップ612では、変数mehに、me.pHTopの値を代入する。続くステップ613では、変数mehの値が変数meの値と一致しないか否か判定する。それらの値が一致する場合、判定はNOとなって上記ステップ617に移行する。そうでない場合には、判定はYESとなってステップ614に移行する。

40

【0066】

ステップ614では、meh.cDispの値は1か否か判定する。その値が1であった場合、判定はYESとなり、ステップ616で和音の構成音の演奏データ全てデータiDispの値を1にする。その後、ステップ617に移行する。そうでない場合には、判定はNOとなり、ステップ615で変数mehにmeh.nextの値を代入した後、上記ステップ613に戻る。

【0067】

me.pHTopの値は和音の先頭に位置する構成音の演奏データのインデクス値であり、その演奏データが示す楽音の直前に条件を満たす休符箇所が存在していれば、そのデータiDispとして1が格納されている。このことから、ステップ611~616の処

50

理により、条件を満たす休符箇所の直後に位置する和音の構成音の演奏データは全て表示設定されることになる。

【0068】

図10は、特殊な指使いを考慮した表示・非表示設定処理のフローチャートである。その設定処理は、ユーザーが条件として特殊な指使いを指定した場合に実行される。次に図10を参照して、その設定処理について詳細に説明する。ここでは特殊な指使いとして、指越え、指くぐりのみを想定する。また、図9に示す休符箇所を考慮した表示・非表示設定処理と同じ、或いは基本的に同じ内容のステップの処理には同一の符号を付している。同一の符号を付したステップの処理の説明は省略する。

【0069】

図10に示す表示・非表示設定処理では、ステップ603の処理の実行後、ステップ701に移行して、変数`men`に、次の演奏データのインデクス値を代入する。その代入後に移行するステップ702では、変数`men`の値で指定される演奏データが存在するか否か判定する。その演奏データが存在する場合、判定はYESとなり、ステップ703で変数`iPassF`に0を代入した後、ステップ704に移行する。そうでない場合には、つまり対象とする演奏データが最後に位置するものであった場合には、判定はNOとなってステップ714に移行する。

【0070】

ステップ704では、`me.iHand`の値が1であり、`me.cf ig`の値が0であり、`men.cf ig`の値が0より大きく、且つ`me.iPos X`の値が`men.iPos X`の値より大きいかが判定する。それらの関係は、右手の演奏において、親指で押鍵すべき鍵の次に、より音高の低い鍵を親指以外の指で押鍵すべき状況の場合に全て満足する。このことから、そのような状況であった場合、判定はYESとなってステップ705に移行し、指越えを行うべき状況であることを示す値の1を変数`iPassF`に代入した後、ステップ706に移行する。そうでない場合には、判定はNOとなって次にそのステップ706の処理を実行する。

【0071】

ステップ706では、`me.iHand`の値が1であり、`me.cf ig`の値が0より大きく、`men.cf ig`の値が0であり、且つ`me.iPos X`の値が`men.iPos X`の値より小さいかが判定する。それらの関係は、右手の演奏において、親指以外の指で押鍵すべき鍵（指定音）の次に、より音高の高い鍵を親指で押鍵すべき状況の場合に全て満足する。このことから、そのような状況であった場合、判定はYESとなってステップ706に移行し、指くぐりを行うべき状況であることを示す値の2を変数`iPassF`に代入した後、ステップ708に移行する。そうでない場合には、判定はNOとなって次にそのステップ708の処理を実行する。

【0072】

ステップ708～711では、左手の演奏を対象に、指越え、指くぐりの何れかの指使いを行うべき状況か否かの判定を行い、その判定結果に応じた値を変数`iPassF`に代入するための処理が行われる。それにより、親指で押鍵すべき鍵の次に、より音高の高い鍵を親指以外の指で押鍵すべき状況、つまり指越えを行うべき状況であれば1、親指以外の指で押鍵すべき鍵の次に、より音高の低い鍵を親指で押鍵すべき状況、つまり指くぐりを行うべき状況であれば2、が変数`iPassF`に代入されることとなる。ステップ712には、ステップ710、或いは711の処理の実行後に移行する。

【0073】

ステップ712では、変数`iPassF`の値が0でないかが判定する。その値が0でない場合、判定はYESとなり、ステップ713で変数`me`に`me.next`の値を代入した後、上記ステップ602に戻る。そうでない場合には、判定はNOとなり、ステップ714で`me.cDisp`の値を0に更新した後、そのステップ713に移行する。

【0074】

図11は、難易度を考慮した表示・非表示設定処理のフローチャートである。その設定

10

20

30

40

50

処理は、ユーザーが条件として難易度を指定した場合に実行される。次に図11を参照して、その設定処理について詳細に説明する。図9に示す休符箇所を考慮した表示・非表示設定処理と同じ、或いは基本的に同じ内容のステップの処理には同一の符号を付し、同一の符号を付したステップの処理の説明は省略する。

【0075】

その難易度を示す値はデータ `icost` として演奏データに格納されている。対象となる難易度の範囲は、図16に示すように、下限値、上限値によりユーザーに指定させるようにしている。ユーザーが指定した下限値、上限値に相当するコスト値（難易度）はそれぞれ変数 `minCost`、`maxCost` に代入される。

【0076】

10
まず、ステップ801では、変数 `me` に、先頭に位置する演奏データのインデクス値を代入する。その代入後に移行するステップ802では、変数 `me` の値が0（`NULL`）でないか否か判定する。その演奏データが最後に位置するものであった場合、データ `next` の値として0（`NULL`）が格納されている。このことから、その場合、判定は`NO`となり、ここで一連の処理を終了する。そうでない場合には、判定は`YES`となってステップ803に移行する。

【0077】

20
ステップ803では、`me.cdisp` の値を0に更新する。続くステップ804では、`me.icost` の値が変数 `minCost` の値以上、且つ変数 `maxCost` の値以下か否か判定する。`me.icost` の値がユーザーの指定範囲内の難易度を示していた場合、判定は`YES`となり、ステップ805で`me.cdisp` の値を1に更新した後、ステップ611に移行する。そうでない場合には、判定は`NO`となってそのステップ611に移行する。それにより、難易度がユーザーの指定範囲内の難易度となっている構成音を有する和音は、全ての構成音で指番号を表示させる。

【0078】

図12は、難易度の正規化処理のフローチャートである。その正規化処理は、ユーザーが条件として正規化した難易度を指定した場合に実行される。次に図12を参照して、その正規化処理について詳細に説明する。

【0079】

30
その難易度を示す値はデータ `icostN` として演奏データに格納されている。正規化した難易度は0～100の数値で表すようにしている。対象となる難易度の範囲は、図17に示すように、下限値、上限値を数値によりユーザーに指定させるようにしている。ユーザーが指定した下限値、上限値に相当するコスト値（難易度）はそれぞれ変数 `minCost`、`maxCost` に代入される。それにより、正規化した難易度を考慮した表示・非表示の設定を行うためのサブルーチン処理は、図11に示す表示・非表示設定処理において、ステップ804のみが異なっている。つまり`me.icost` の代わりに`me.icostN` を用いることのみが異なっている。このことから、正規化した難易度を考慮した表示・非表示の設定を行うためのサブルーチン処理についての詳細な説明は省略する。

【0080】

40
まず、ステップ901では、変数の初期化を行う。その初期化により、変数 `CostAve`、`Ncount`、及び`maxCost` にはそれぞれ0を代入し、変数 `minCost` には100を代入する。続くステップ902では、変数 `me` に、先頭に位置する演奏データのインデクス値を代入する。その代入後はステップ903に移行して、変数 `me` の値で指定される演奏データが最後に位置するものではないか否か判定する。その演奏データが最後に位置するものであった場合、判定は`NO`となってステップ910に移行する。そうでない場合には、判定は`YES`となってステップ904に移行する。

【0081】

50
ステップ904では、変数 `CostAve` に、それまでの値に変数 `me.icost` の値を加算した結果を代入し、変数 `Ncount` の値をインクリメントする。次のステップ905では、`me.icost` の値が変数 `maxCost` の値より大きいのか否か判定する

。前者が後者より大きい場合、判定はYESとなり、ステップ706で変数maxCostにme. icostの値を代入した後、ステップ907に移行する。そうでない場合には、判定はNOとなってそのステップ907に移行する。

【0082】

ステップ907では、me. icostの値が変数minCostの値より小さいか否か判定する。前者が後者より小さい場合、判定はYESとなり、ステップ708で変数minCostにme. icostの値を代入し、次のステップ909で変数meにme. nextの値を代入した後、上記ステップ903に戻る。そうでない場合には、判定はNOとなってそのステップ909に移行する。

【0083】

上記ステップ903～909で形成される処理ループをステップ903の判定がNOとなるまで繰り返し実行することにより、コスト値（難易度）の累算値が変数CostAveにストアされ、演奏データの総数が変数Ncountにストアされ、コスト値の最大値が変数maxCostにストアされ、コスト値の最小値が変数minCostにストアされる。ステップ903の判定がNOとなって移行するステップ910以降では、そのようにして取得されたデータを用いて難易度を正規化するための処理が行われる。

【0084】

先ず、ステップ910では、変数CostAveに、それまでの値を変数Ncountの値で割った減算結果、つまりコスト値の平均値を代入する。次のステップ911では、変数meに、先頭に位置する演奏データのインデクス値を代入する。その次に移行するステップ912では、変数meの値で指定される演奏データが最後に位置するものではないか否か判定する。その演奏データが最後に位置するものであった場合、判定はNOとなり、ここで一連の処理を終了する。そうでない場合には、判定はYESとなってステップ913に移行する。

【0085】

ステップ913では、変数idifに、me. icostの値から変数CostAveの値を引いた減算結果を代入する。続くステップ914では、変数idifの値が0以上か否か判定する。me. icostの値がコスト値の平均値未満であった場合、変数idifの値は0未満となるから、判定はNOとなり、ステップ915で変数idenに変数CostAveの値から変数minCostの値を引いた減算結果を代入し、次のステップ916でme. icostNに、変数idifの値を変数idenの値で割った除算結果に50を乗算し、その乗算結果に50を加算して得られる値（ $= 50 + idif / iden \cdot 50$ ）を代入し、その次のステップ917で変数meにme. nextの値を代入した後、上記ステップ912に戻る。そうでない場合には、判定はYESとなり、ステップ918で変数idenに変数maxCostの値から変数CostAveの値を引いた減算結果を代入した後、上記ステップ916に移行する。

【0086】

このようにして本実施の形態では、コスト値の最小値を0、その最大値を100として、その間を等分に分ける形でコスト値を正規化している。正規化を行う方法は、そのようなものに限定されるわけではなく、様々な方法を採用しても良い。

【0087】

図13は、図3に示す全体処理内でステップ106として実行される描画処理のフローチャートである。次に図13を参照して、その描画処理について詳細に説明する。

先ず、ステップ1001では、変数meに、表示上、先頭となる演奏データのインデクス値（図中「meDSt」と表記）を代入する。続くステップ1002では、変数meの値で指定される演奏データに相当する音符の画像データを生成して表示部15に表示させる。上述したように、その音符の種類はデータlGateから特定し、表示位置は、データlTimeから特定する。

【0088】

ステップ1002に続くステップ1003では、me. idispの値が1か否か判定

10

20

30

40

50

する。その値が1であった場合、判定はYESとなり、ステップ1004でme. c f i gの値に1を加えた値を指番号としてme. D r a w P o sが示す座標に表示させた後、ステップ1005に移行する。そうでない場合には、判定はNOとなってそのステップ1005に移行する。

【0089】

ステップ1005では、変数meの値が表示上の終端となる演奏データのインデクス値（図中「me D E d」と表記）と一致するか否か判定する。表示すべき音符を全て表示させた場合、それらの値が一致することから、判定はYESとなり、ここで一連の処理を終了する。そうでない場合には、判定はNOとなり、ステップ1006で変数meにme. n e x tの値を代入した後、上記ステップ1002に戻る。

10

【0090】

この描画処理を実行することにより、ユーザーが指定（選択）した条件の種類に応じて、図14～図17に示すような楽譜の何れかが表示部15に表示されることとなる。

なお、本実施の形態では、ユーザーが指定可能な条件は1種類のみとしているが、複数の条件を一度に指定できるようにしても良い。また、運指を表す指番号は、楽譜と併せて予め表示させるようにしているが、その指番号が示す指で押鍵を行うべきタイミングとなった場合にのみ表示させるようにしても良い。その場合、指番号ではなく、押鍵すべき指を表す画像、別の文字情報、などによってユーザーに通知しても良い。

【図面の簡単な説明】

【0091】

20

【図1】本実施の形態による運指表示装置の構成を説明する図である。

【図2】演奏データの構成を説明する図である。

【図3】全体処理のフローチャートである。

【図4】運指評価処理のフローチャートである。

【図5】2音間コスト算出処理のフローチャートである。

【図6】運指コストテーブルの構成を説明する図である。

【図7】禁止事項検索処理のフローチャートである。

【図8】コスト補正処理のフローチャートである。

【図9】休符箇所を考慮した表示・非表示処理のフローチャートである。

【図10】特殊な指使いを考慮した表示・非表示処理のフローチャートである。

30

【図11】難易度を考慮した表示・非表示処理のフローチャートである。

【図12】難易度の正規化処理のフローチャートである。

【図13】描画処理のフローチャートである。

【図14】楽譜上、休符の直後に位置する音符の発音に用いるべき指の指番号を表示させた場合の運指表示例を説明する図である。

【図15】指越え・指くぐりなどの特殊な指使いをして音符を発音させるべき指の指番号を表示させた場合の運指表示例を説明する図である。

【図16】音符を発音させる難易度によりその発音に用いるべき指の指番号を表示させた場合の運指表示例を説明する図である。

【図17】正規化した難易度により音符の発音に用いるべき指の指番号を表示させた場合の運指表示例を説明する図である。

40

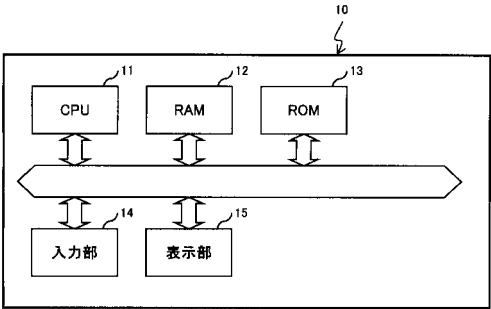
【符号の説明】

【0092】

- 10 運指表示装置
- 11 CPU
- 12 RAM
- 13 ROM
- 14 入力部
- 15 表示部

【図 1】

本実施の形態による運指表示装置の構成を説明する図



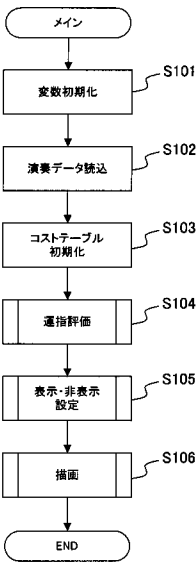
【図 2】

演奏データの構成を説明する図

MidEvent[0]	Time	発音開始時刻 (Tick)
	Date	発音時刻 (Tick)
	TimeSec	発音開始時刻 (絶対時間 msec)
	Pitch	ピッチ
	PosK	鍵盤の座標
	DrawPos	描画用座標
	Clg	割断された運指
	cDisp	表示の非表示のフラグ (0: 非表示 1: 表示)
	cHand	パート (1: 右手 2: 左手)
	icost	評価値 (この音を弾くための動作コスト)
	icostN	正規化した評価値
	cisHarm	自分自身が和音かどうか (和音でない: 1, 1-和音 (構成音-1))
	iArpS	アルペジオの先頭フラグ
	iArpE	アルペジオの末端フラグ
	pHTop	和音の先頭 (一番下の音) へのポイント
	pHTail	和音の終端 (一番上の音) へのポイント
	cTendency	音列の高低傾向 (0: 横断 - 上昇 - 下降 (減り幅 2 変化あり))
	cTendSteps	極値までの変化の段階
	cisSO	ポリフォニーとなっているかどうかのフラグ (絶対値が音数)
	pPHTop	前の音列群の和音の先頭 (一番下の音) へのポイント
	pPHTail	前の音列群の和音の終端 (一番上の音) へのポイント
	pNHTop	次の音列群の和音の先頭 (一番下の音) へのポイント
	pNHTail	次の音列群の和音の終端 (一番上の音) へのポイント
	iNP	禁止条件に該当するかどうか (0: 該当しない)
	prev	前の演奏データ (0 で先頭)
	next	後の演奏データ (0 で末端)
MidEvent[1]		
	...	
MidEvent[N]		
	...	

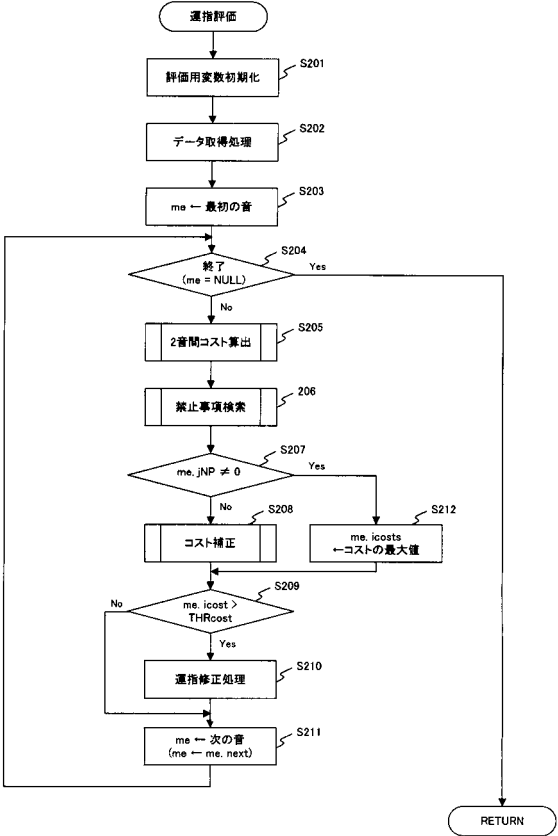
【図 3】

全体処理のフローチャート



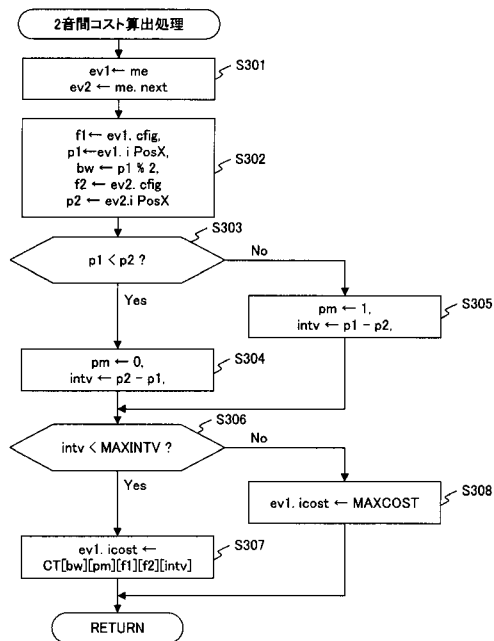
【図 4】

運指評価処理のフローチャート



【図 5】

2音間コスト算出処理のフローチャート



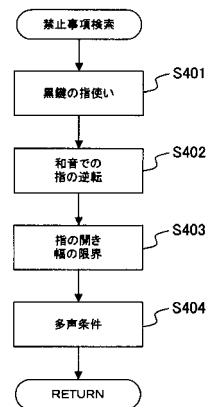
【図 6】

運指コストテーブルの構成を説明する図

[bw]	[pm]	[fig1]	[fig2]	[intv]
				[0]
				
				[MAXINTV - 1]
		[0]		⋮
			[4]	⋮
		[0]		⋮
			[4]	⋮
		[0]		⋮
			[4]	⋮
		[1]		⋮
			[1]	⋮

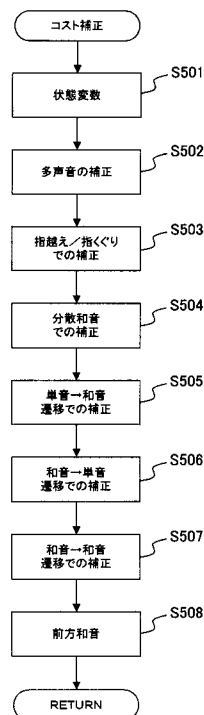
【図 7】

禁止事項検索処理のフローチャート



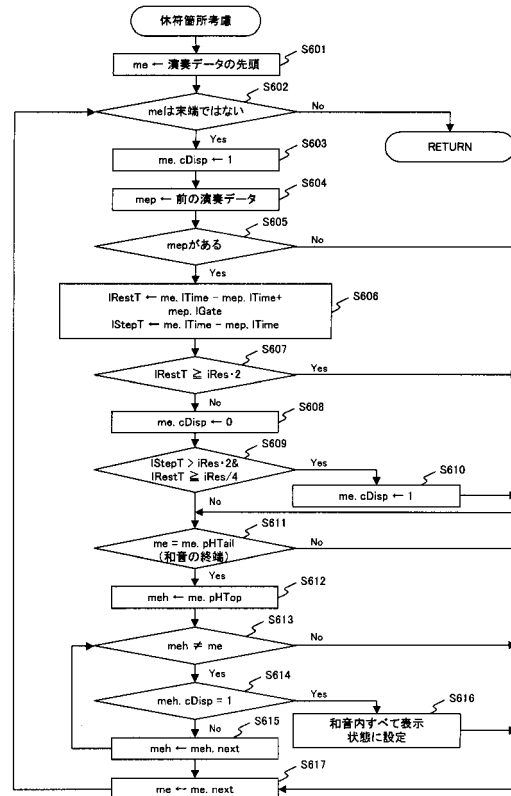
【図 8】

コスト補正処理のフローチャート



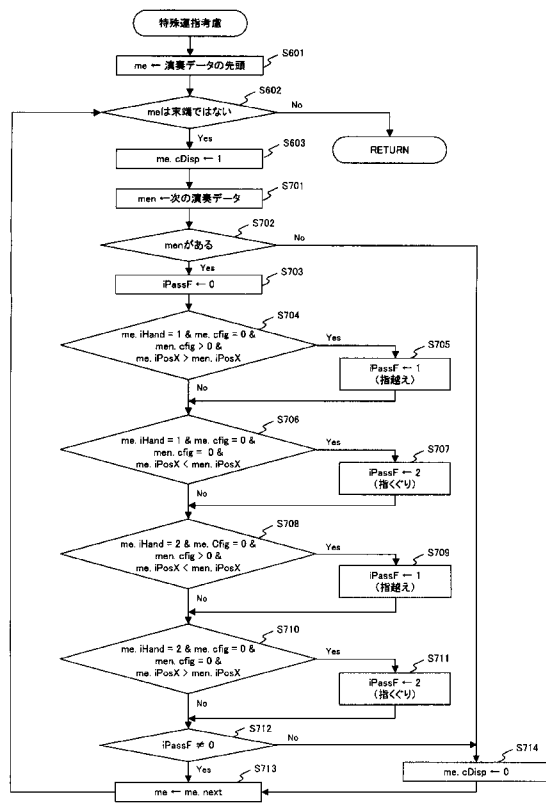
【図 9】

休符箇所を考慮した表示・非表示処理のフローチャート



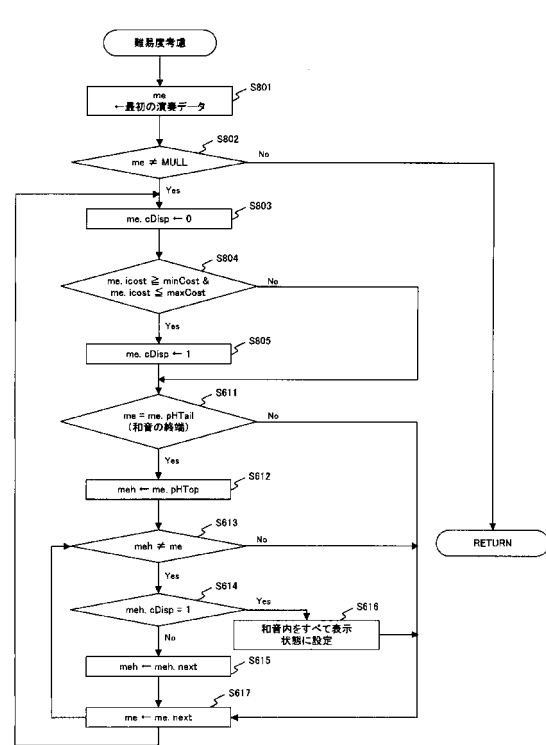
【図 10】

特殊な指使いを考慮した表示・非表示処理のフローチャート



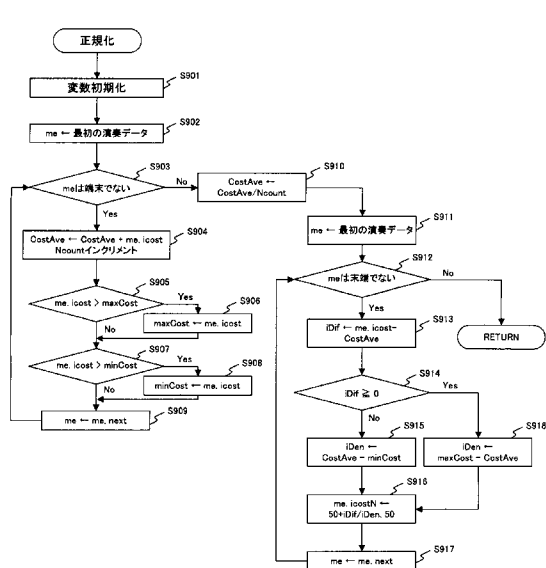
【図 11】

難易度を考慮した表示・非表示処理のフローチャート



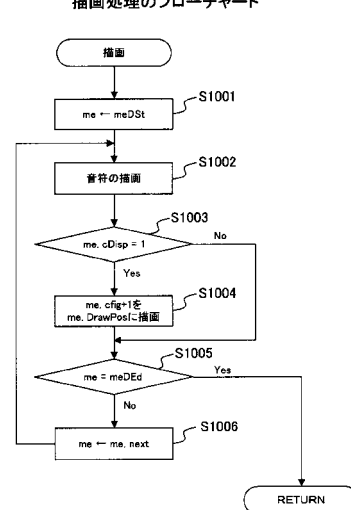
【図 12】

難易度の正規化処理のフローチャート



【図 13】

描画処理のフローチャート



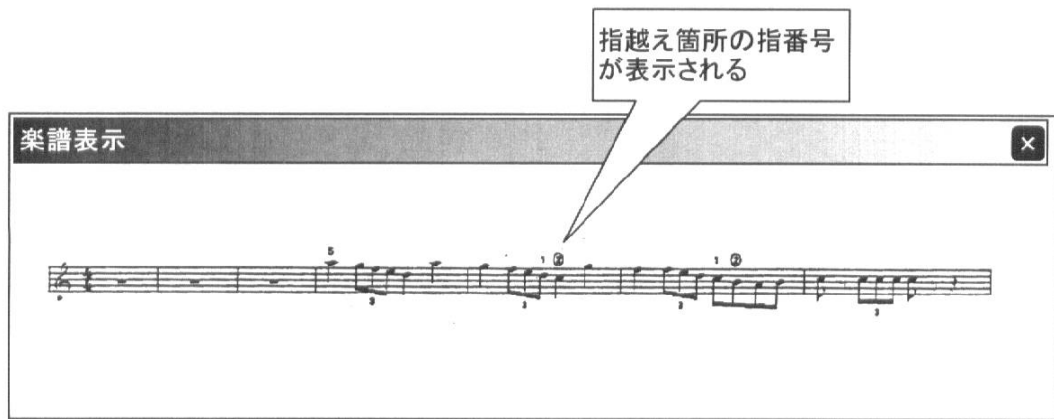
【図 1 4】

楽譜上、休符の直後に位置する音符の発音に用いるべき指の指番号を表示させた場合の運指表示例を説明する図



【図 1 5】

指越え・指くぐりなどの特殊な指使いをして音符を発音させるべき指の指番号を表示させた場合の運指表示例を説明する図



【図 16】

音符を発音させる難易度によりその発音に用いるべき指の指番号を表示させた場合の運指表示例を説明する図

楽譜表示

同音を異なる指で押鍵する部分が難しいと判断され、指番号が表示される



表示設定

■ 下限値	■ 上限値
Lv.4	Lv.15
Lv.5	Lv.14
Lv.6	Lv.13
Lv.7	Lv.12

表示更新

【図 17】

正規化した難易度により音符の発音に用いるべき指の指番号を表示させた場合の運指表示例を説明する図

楽譜表示



表示設定

■ 下限値	■ 上限値
20	100
30	90
40	80
50	70

表示更新

フロントページの続き

- (56)参考文献 特開2006-064745 (JP, A)
特開2005-165029 (JP, A)
特開2001-142461 (JP, A)
特開2002-323890 (JP, A)
特開平11-338465 (JP, A)
実開昭58-192671 (JP, U)
特開平10-207346 (JP, A)
特開2003-066958 (JP, A)

(58)調査した分野(Int.Cl., DB名)

G10H 1/00- 7/12
G10G 1/00- 7/02
G09B 15/00- 15/08