

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第6427994号
(P6427994)

(45) 発行日 平成30年11月28日 (2018.11.28)

(24) 登録日 平成30年11月9日 (2018.11.9)

(51) Int. Cl. F I
G 1 O G 1/02 (2006.01) G 1 O G 1/02
G 1 O H 1/00 (2006.01) G 1 O H 1/00 Z

請求項の数 8 (全 51 頁)

(21) 出願番号	特願2014-135028 (P2014-135028)	(73) 特許権者	000001443
(22) 出願日	平成26年6月30日 (2014.6.30)		カシオ計算機株式会社
(65) 公開番号	特開2016-12103 (P2016-12103A)		東京都渋谷区本町1丁目6番2号
(43) 公開日	平成28年1月21日 (2016.1.21)	(74) 代理人	100074099
審査請求日	平成29年6月21日 (2017.6.21)		弁理士 大菅 義之
		(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	大野 弘
		(56) 参考文献	特開2003-202860 (JP, A) 特開2001-324919 (JP, A)
			最終頁に続く

(54) 【発明の名称】 楽器演奏のための動作情報生成装置、電子楽器、方法、およびプログラム

(57) 【特許請求の範囲】

【請求項1】

楽曲を構成する楽音夫々の発音タイミング情報と、音高情報と、演奏に用いる手及び指を示す情報と、を含む演奏情報から、右手及び左手の指夫々が複数の演奏操作子のいずれの位置にあるかを示す指位置を推定する位置推定手段と、

前記位置推定手段により推定された前記右手及び左手の指位置同士の関係が予め定められた条件を満たすか否かを判別する判別手段と、

前記判別手段にて条件を満たすと判別された場合に、前記位置推定手段にて推定された前記右手及び左手の指位置のうち、少なくとも一方の手の指位置を補正する位置補正手段と、

前記位置補正手段により補正された前記指位置に基づいて、演奏操作ガイド情報を生成するガイド情報生成手段と、

を備えることを特徴とする演奏操作ガイド装置。

【請求項2】

前記判別手段は、前記位置推定手段により推定された右手の各指位置と左手の各指位置のうち、前記左手の親指位置が、前記右手の親指位置より右側にあるか否かを判別する、ことを特徴とする請求項1に記載の演奏操作ガイド装置。

【請求項3】

前記位置補正手段は、

前記演奏情報に含まれる発音タイミング情報に基づき、演奏している指が前記右手及び

左手のいずれに存在しているかを判別する演奏指判別手段をさらに有する、
ことを特徴とする請求項 1 または 2 のいずれかに記載の演奏操作ガイド装置。

【請求項 4】

前記位置補正手段は、

前記演奏指判別手段により、前記右手に演奏している指が存在していると判別された場合は、前記左手の指位置を、当該左手方向に補正する一方、前記左手に演奏している指が存在していると判別された場合は、前記右手の指位置を、当該右手方向に補正する、

ことを特徴とする請求項 3 に記載の演奏操作ガイド装置。

【請求項 5】

前記位置補正手段は、

前記演奏指判別手段により、前記右手及び左手の両方に演奏している指が存在していると判別された場合は、前記右手及び左手の少なくとも一方の指位置を、当該右手方向及び左手方向以外の方向に補正する、

ことを特徴とする請求項 3 または 4 のいずれかに記載の演奏操作ガイド装置。

【請求項 6】

請求項 1 から 5 のいずれかに記載の演奏操作ガイド装置と、

演奏者に演奏操作を行わせる演奏操作子と、

前記演奏操作子における演奏操作に応じて楽音を生成して出力する楽音出力部とを備える電子楽器。

【請求項 7】

演奏操作ガイド装置が、

楽曲を構成する楽音夫々の発音タイミング情報と、音高情報と、演奏に用いる手及び指を示す情報と、を含む演奏情報から、右手及び左手の指夫々が複数の演奏操作子のいずれの位置にあるかを示す指位置を推定し、

前記推定された前記右手及び左手の指位置同士の関係が予め定められた条件を満たすか否か判別し、

前記条件を満たすと判別された場合に、前記推定された右手及び左手の指位置のうち、少なくとも一方の手の指位置を補正し、

前記補正された前記指位置に基づいて、演奏操作ガイド情報を生成する、演奏操作ガイド方法。

【請求項 8】

楽曲を構成する楽音夫々の発音タイミング情報と、音高情報と、演奏に用いる手及び指を示す情報と、を含む演奏情報から、右手及び左手の指夫々が複数の演奏操作子のいずれの位置にあるかを示す指位置を推定する位置推定ステップと、

前記推定された前記右手及び左手の指位置同士の関係が予め定められた条件を満たすか否か判別する判別ステップと、

前記条件を満たすと判別された場合に、前記推定された右手及び左手の指位置のうち、少なくとも一方の手の指位置を補正する位置補正ステップと、

前記補正された前記指位置に基づいて、演奏操作ガイド情報を生成するガイド情報生成ステップと、

をコンピュータに実行させる演奏操作ガイドプログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、鍵盤楽器における運指ガイド等の演奏動作のガイドを行うための動作情報を生成する技術に関する。

【背景技術】

【0002】

例えば鍵盤楽器における演奏動作を視覚的にガイドするために、従来、鍵盤楽器の画像上で演奏における指の運指をガイドする技術が知られている（例えば特許文献 1）。

10

20

30

40

50

【先行技術文献】

【特許文献】

【0003】

【特許文献1】特許第3521838号公報

【発明の概要】

【発明が解決しようとする課題】

【0004】

ここで、例えば鍵盤楽器を演奏する場合には、さまざまな演奏動作上のテクニック(指越え・指くぐり、手を移動させる、手を広げる・狭めるなど)が存在する。従って、運指を習得するだけでなく、これらの演奏動作上のテクニックを習得することも望まれる。また、鍵盤楽器等の演奏に際しては、現在だけでなく、次以降の音形によって、どのように指や手の位置を準備させるかといった準備動作も重要なテクニックの一つであった。

10

【0005】

しかし、従来の運指ガイド技術は、単にあるタイミングで押鍵すべき鍵盤の位置とその鍵盤を押鍵する指を指示するだけであり、演奏動作上のテクニックを習得することは困難であるという問題点を有していた。

【0006】

本発明は、演奏情報に付加されている運指情報から演奏動作上のテクニックに関する動作情報を自動的に抽出可能とし、それにより両手パートが衝突しない運指とともに演奏動作上のテクニックも演奏ガイド可能とすることを目的とする。

20

【課題を解決するための手段】

【0007】

態様の一例では、楽曲を構成する楽音夫々の発音タイミング情報と、音高情報と、演奏に用いる手及び指を示す情報と、を含む演奏情報から、右手及び左手の指夫々が複数の演奏操作子のいずれの位置にあるかを示す指位置を推定する位置推定手段と、前記位置推定手段により推定された前記右手及び左手の指位置同士の関係が予め定められた条件を満たすか否かを判別する判別手段と、前記判別手段にて条件を満たすと判別された場合に、前記位置推定手段にて推定された前記右手及び左手の指位置のうち、少なくとも一方の手の指位置を補正する位置補正手段と、前記位置補正手段により補正された前記指位置に基づいて、演奏操作ガイド情報を生成するガイド情報生成手段と、を備える。

30

【発明の効果】

【0008】

本発明によれば、演奏情報に付加されている運指情報から演奏動作上のテクニックに関する動作情報を自動的に抽出することが可能となり、それにより両手パートが衝突しない運指とともに演奏動作上のテクニックも演奏ガイドすることが可能となる。

【図面の簡単な説明】

【0009】

【図1】本発明による動作情報生成装置の実施形態を示すブロック図である。

【図2】動作情報生成装置の実施形態を組み込んだ演奏ガイド装置のハードウェア構成の一例を示す図である。

40

【図3】演奏情報のデータ構成例を示す図である。

【図4】動作データのデータ構成例を示す図である。

【図5】本実施形態の全体動作処理の例を示すフローチャートである。

【図6】動作情報生成の処理の例を示すフローチャートである。

【図7】次鍵の検索の処理の例を示すフローチャートである。

【図8】Nextバッファ更新の処理の例を示すフローチャートである。

【図9】パート毎の状態確認の処理の例を示すフローチャートである。

【図10】離鍵確認の処理の例を示すフローチャートである。

【図11】押鍵検索の処理の例を示すフローチャートである。

【図12】準備確認の処理の例を示すフローチャート(その1)である。

50

【図 1 3】準備確認の処理の例を示すフローチャート（その 2）である。

【図 1 4】準備確認の処理の例を示すフローチャート（その 3）である。

【図 1 5】準備確認の処理の例を示すフローチャート（その 4）である。

【図 1 6】鍵盤距離の取得の処理の例を示すフローチャートである。

【図 1 7】パート同士の状態比較処理の例を示すフローチャートである。

【図 1 8】動作データ出力の処理の例を示すフローチャート（その 1）である。

【図 1 9】動作データ出力の処理の例を示すフローチャート（その 2）である。

【図 2 0】奥行き位置計算の処理の例を示すフローチャートである。

【図 2 1】次回開始時刻の計算の処理の例を示すフローチャートである。

【図 2 2】演奏情報の記憶例を示す図である。

10

【図 2 3】準備動作を考慮しないときの動作データの出力例を示す図である。

【図 2 4】準備動作を考慮したときの動作データの出力例を示す図である。

【図 2 5】演奏ガイドの表示例を示す図である。

【図 2 6】左手の押鍵ガイドがその演奏進行に伴って右手の押鍵の準備ガイドに衝突しうる楽譜例を示す図である。

【図 2 7】衝突回避制御を行わないときの衝突が発生する場合の右手パートに関する動作データの出力例（その 1）を示す図である。

【図 2 8】衝突回避制御を行うときの衝突が発生する場合の右手パートに関する動作データの出力例（その 1）を示す図である。

【図 2 9】同じ押鍵タイミングでの右手および左手の接近により両者の演奏ガイドが衝突しうる楽譜例を示す図である。

20

【図 3 0】衝突回避制御を行わなわれないときの衝突が発生する場合の右手パートおよび左手パートに関する動作データの出力例（その 2）を示す図である。

【図 3 1】衝突回避制御を行うときの衝突が発生する場合の右手パートおよび左手パートに関する動作データの出力例（その 2）を示す図である。

【発明を実施するための形態】

【0010】

以下、本発明を実施するため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、本発明による演奏操作ガイド装置 100 の実施形態を示すブロック図である。この演奏操作ガイド装置 100 は、演奏情報読み出し手段 101、操作状態推定手段 102、パート同士の状態比較手段 103、演奏操作ガイド手段 104 を備える。また、内部または外部に、記憶部 105 が具備される。

30

【0011】

演奏情報読み出し手段 101 は、曲を構成する各楽音の、発音を行う発音タイミングに関する情報と、音高に関する情報と、演奏者の演奏操作に関する情報と、を含む演奏情報 108 を記憶部 105 から読み出す。

【0012】

操作状態推定手段 102 は、演奏情報読み出し手段 101 によって読み出された演奏情報 108 夫々に基づいて、その演奏情報 108 における演奏者の演奏操作状態を推定する。

40

【0013】

パート同士の状態比較手段 103 は、操作状態推定手段 102 が推定した演奏操作情報をパート間で比較することにより、演奏操作に不都合が生じないように、パート毎の演奏操作状態を補正する。

【0014】

演奏操作ガイド手段 104 は、操作状態推定手段 102 により推定されパート同士の状態比較手段 103 により補正された第 1 の演奏操作状態 106 と、第 1 の演奏操作状態 106 より後に推定された第 2 の演奏操作状態 107 とに基づいて、演奏者の演奏操作に関するガイドを行う。

【0015】

50

上述の構成において、パート同士の状態比較部 103 は、操作状態推定手段 102 が推定した演奏操作状態が示す各指の演奏位置がパート間で衝突する状態にある場合に、演奏操作状態中の各指の演奏位置に関する情報を補正してよい。

【0016】

より具体的には、操作状態推定手段 102 は、各指の奥行き方向の位置に関する情報をさらに含む演奏操作状態を推定する。そして、パート同士の状態比較手段 103 は、操作状態推定手段 102 が推定した演奏操作状態が示す親指に最も近い演奏位置の音高をパート毎に取得する。そして、パート同士の状態比較手段 103 は、その音高のうち左手のパートの音高が右手のパートの音高よりも高い場合に、左手のパートおよび右手のパートともに第 1 の演奏操作状態を含むならば、演奏操作状態中の各指の奥行き方向の位置に関する情報を補正する。また、パート同士の状態比較手段 103 は、左手のパートおよび前記右手のパートのうち的一方のパートのみが第 1 の演奏操作状態を含むならば、他方のパートの演奏操作状態中の前記各指の音高に関する情報を補正する。

10

【0017】

ここまでの構成において、演奏操作ガイド装置 100 は、鍵盤楽器の演奏操作に関するガイドを行うものとしてよい。このとき、操作状態推定手段 102 は、演奏者の第 1 及び第 2 の演奏操作状態 106, 107 として、親指で鍵盤を押鍵中の状態及びこの状態から親指以外の指で親指の上を越えて移動させて次以降のタイミングにて押鍵する状態を推定してよい。この場合に、演奏操作ガイド手段 104 は、親指が演奏を行うべき指としてガイドするとともに、親指以外の指を演奏の準備を行うべき指としてガイドしてよい。また、操作状態推定手段 102 が、演奏者の第 1 及び第 2 の演奏操作状態 106, 107 として、親指で鍵盤を押鍵中の状態及びこの状態から親指以外の指で押鍵中に親指で親指以外の指の下をくぐらせて移動させて次以降のタイミングにて押鍵する状態を推定してよい。この場合に、演奏操作ガイド手段 104 は、親指以外の指を演奏を行うべき指としてガイドするとともに、親指を演奏の準備を行うべき指としてガイドしてよい。さらに、操作状態推定手段 102 は、演奏者の第 1 及び第 2 の演奏操作状態 106, 107 として、鍵盤の所定の鍵が押鍵されている状態及びその押鍵されている鍵から所定の基準値よりも離れた位置の鍵が次に押鍵される状態を推定してよい。この場合に、演奏操作ガイド手段 104 は、現在の押鍵状態から次の押鍵状態への移動の準備をガイドしてよい。

20

【0018】

以上の実施形態の構成により、操作状態推定手段 102 が、演奏情報読み出し手段 101 によって読み出された演奏情報 108 夫々に基づいて、演奏情報 108 における演奏者の演奏操作状態を、第 1 および第 2 の演奏操作状態 106, 107 として推定することができる。これらの第 1 および第 2 の演奏操作状態 106, 107 を用いて、演奏操作ガイド手段 104 が演奏ガイドを行うことにより、各動作タイミングにおいて、演奏を行うべき指の運指ガイドだけでなく、それ以降の演奏の準備を行うべき指のガイドも行うことができる。演奏の準備の動作としては、他の指による親指越えや、親指による指くぐり、広範囲での指の移動などの動作を含むことができ、これにより演奏動作のテクニックに関するガイドを行うことが可能となる。

30

【0019】

さらに、本実施形態では、左手と右手のパートで練習を行うための演奏ガイドを実行するときに、楽譜上、左手パートと右手パートの間で、演奏される指同士が衝突しそうになったり、あるいは次に演奏する準備を行うべき指を置く場所と演奏中の指が衝突しそうになった場合に、パート同士の状態比較手段 103 が、それらの状態を検知して補正することが可能となる。

40

【0020】

具体的には、演奏中の指同士が衝突しそうになった場合は、各パートの演奏中の指および演奏の準備を行うべき指全体を奥行き方向の前後にずらして演奏ガイドをすることにより、衝突を回避できる。例えば鍵盤楽器の場合、一方のパートが奥の黒鍵側にある場合は、そのパートの指全体がさらに黒鍵側に移動するように、また他方のパートが手前の白鍵

50

側にある場合、そのパートの指全体がさらに白鍵側に移動するように、演奏位置がずらされる。また、次に演奏の準備を行うべき指を置く場所と演奏中の指が衝突しそうになった場合には、左手と右手の各パートで指がオーバーラップしている分だけ、演奏の準備を行うべき指のほうを音高指定方向（例えば鍵盤楽器の鍵盤の横方向）にずらして演奏ガイドを行うことにより、衝突を回避できる。このようにして、本実施形態では、両手パートが衝突しない運指ガイドが可能となる。

【0021】

図2は、図1の機能構成を有する演奏動作ガイド装置100の実施形態のハードウェア構成の一例を示す図である。この装置100は、CPU（中央演算処理装置）201、ROM（リードオンリーメモリ）202、RAM（ランダムアクセスメモリ）203、入力部204、表示部205、およびMIDIインタフェース部（以下「MIDI I/F」と表記する）206が、バス207によって相互に接続された構成を備える。MIDI I/F 206は、外部（内蔵でもよい）の電子楽器208に接続される。

10

【0022】

CPU 201は、ROM 202に記憶された制御プログラムに従って、当該演奏操作ガイド装置100の全体の制御を行う。

【0023】

ROM 202は、後述する各フローチャートで示される制御動作を実行する制御プログラムを記憶する。また、ROM 202は、複数の楽曲の図1に示される演奏情報108などを記憶する。

20

【0024】

RAM 203は、制御プログラムの実行に必要な、後述する図3、図4に示される、演奏情報、動作データなどの各種情報を、一時的に記憶する。

【0025】

入力部204は、ユーザによるキーボード、マウス、各種スイッチ等による入力操作を検出し、その検出結果をCPU 201に通知する。

【0026】

表示部205は、CPU 201の制御によって送られてくるデータを液晶ディスプレイなどに出力する。例えば、表示部205は、後述する図25のような表示形式で、演奏動作ガイドを表示する。

30

【0027】

MIDI I/F 206は、CPU 201が、ROM 202またはRAM 203上の演奏情報を、MIDI (Musical Instrument Digital Interface) のデータフォーマットで、電子楽器208に順次転送することにより、演奏動作ガイドを行う楽曲を自動演奏させることができる。

【0028】

本実施形態による演奏操作ガイド装置100の動作は、後述する各フローチャート等で実現される図1の各部の機能処理を搭載した制御プログラムを、CPU 201がROM 202から読み出して順次実行することで実現される。そのプログラムは、特には図示しない可搬型記録媒体を介して提供されてもよく、或いは特には図示しないネットワーク接続装置によりネットワークから取得できるようにしてもよい。

40

【0029】

以下、図1の機能構成と図2のハードウェア構成を有する本実施形態による演奏操作ガイド装置100の動作について、以下に詳細に説明する。

【0030】

図3は、演奏動作ガイドを行う曲の演奏情報Note[0]、Note[1]、・・・、Note[N-1]のデータ構成例を示す図であり、図1の演奏情報108に対応する。演奏情報は、図2において、CPU 201が電子楽器208に対して演奏ガイドをさせるにあたって必要な、インデックス番号(0、1、・・・、N-1)(Nは1曲分の演奏情報の数)、発音を開始する発音開始時刻Timeと発音継続時間Gate(図1の「発

50

音タイミングに関する情報」に対応)、音高Pitch(図1の「音高に関する情報」に対応)、右手(値「0」)または左手(値「1」)を示すトラック番号Hand(図1の「パートに関する情報」に対応)、および指番号Finger(図1の「演奏する指に関する情報」に対応)の各情報を記憶する。演奏情報Note[0]、Note[1]、・・・、Note[N-1]は、図2において、CPU201によってROM202からRAM203に転送された後に、順次読み出されて処理されるが、発音開始時刻Timeの順に並ぶように、インデックス番号(0、1、・・・、N-1)が付与されている。

【0031】

図4は、図2のCPU201による後述する制御処理によってRAM203上に生成される動作データframe[0]、frame[1]、・・・、frame[FrameCount-1]のデータ構成例を示す図であり、図1の動作情報107、108に対応する。frame[0]、frame[1]、・・・、frame[FrameCount-1]はそれぞれ、フレーム番号0、1、・・・、FrameCount-1に対応する動作データである。すなわち、動作データの出力単位はフレームである。FrameCountは、RAM203上に生成される動作データのフレームの総数を示す。動作データは、図2において、CPU201が表示部205に演奏動作ガイドを表示させるにあたって必要な、右手(値「0」)または左手(値「1」)を示すトラック番号「Hand」、5本の各指の演奏位置「position[0]~position[4]」(指を置くべき鍵盤番号)(図1の「各指の演奏位置に関する情報」に対応)、そこへ動くにあたっての動作を開始・終了する時刻である動作開始時間「start」と動作終了時間「term」(図1の「動作タイミングに関する情報」に対応)、各指の状態「status[0]~status[4]」(それぞれ、値「-1」が準備動作、値「1」が押鍵)(図1の「各指の演奏のガイド/演奏準備のガイドを指示する情報」に対応)、奥行き方向の位置「Zpos」(図1の「奥行き方向の位置に関する情報」に対応)などの情報を記憶する。

【0032】

以下、本実施形態の動作について、フローチャートを用いて説明する。以下の説明において、図1の各機能部分と図2の各ハードウェア部分を随時参照するものとする。また、図3で説明した演奏情報Note[0]~Note[N-1]、および図4で説明した動作データframe[0]~frame[FrameCount-1]は、RAM203に記憶されるものとする。さらに、以下の説明で用いる各変数も、RAM203に記憶されるものとする。また、以下の説明では、ある変数の名称が、その変数が記憶している値を指す場合がある。

【0033】

図5は、本実施形態全体動作処理を示すメインフローチャートである。この処理は、図2において、CPU201が、ROM202に記憶された制御プログラムを実行する動作として実現される。

【0034】

本実施形態では、ユーザが入力部204から楽曲を選択することにより、CPU201が、その楽曲1曲分の演奏情報Note[i](0 ≤ i ≤ N-1)(図3参照)をROM202からRAM203に読み出し、その読み出した各演奏情報Note[i]に対して順次、動作情報を一括して生成する。

【0035】

処理が起動すると、CPU201は、RAM203上の演奏情報や各種変数を初期化し(ステップS501)、RAM203から図3に例示される構成の演奏情報を読み込み(ステップS502)、RAM203上に図4に例示されるデータフォーマットで動作データを生成し(ステップS503)、プログラムを終了する。

【0036】

図6は、図5のステップS503の動作情報生成処理の例を示すフローチャートである。

10

20

30

40

50

【0037】

まず、CPU201は、各変数を初期化する(ステップS601)。CPU201は、演奏情報検索ポインタ変数meTail[0]に、RAM203に図3に例示されるフォーマットで記憶されている演奏情報Note[0]、Note[1]、・・・、Note[N]のうちから、トラック番号(図3参照)を参照することにより、右手パートの最初の演奏情報へのポインタを設定する。同様に、CPU201は、演奏情報検索ポインタ変数meTail[1]に、上記演奏情報Note[0]、Note[1]、・・・、Note[N]のうちから、左手パートの最初の演奏情報へのポインタを設定する。動作開始時刻変数start[0]とstart[1]には、時刻0が設定される。start[0]は右手パート用、start[1]は左手パート用である。この変数は、ステップS607の動作データ出力の処理で使用される。現在のタイミングにおいて各指を押鍵中とすべき演奏情報へのポインタ変数Hand[0][0]~Hand[0][4](これら4つの変数は右手の親指から小指に対応)およびHand[1][0]~Hand[1][4](これら4つの変数は左手の親指から小指に対応)には、未定義値NULLが設定される。以降、この変数を、「押鍵対象演奏情報ポインタ変数」と呼ぶ。また、図4の動作データの総数を示す変数FrameCountの値が「0」に初期設定される。この変数は、ステップS607の動作データ出力の処理で操作される。

10

【0038】

次に、CPU201は、次鍵の検索の処理を実行する(ステップS602)。この処理は、各パートの各指の次に押鍵する演奏情報へのポインタ変数Next[0][0]~Next[0][4](これら4つの変数は右手パートの親指から小指に対応)およびNext[1][0]~Next[1][4](これら4つの変数は左手パートの親指から小指に対応)の各値を算出する処理である。以降、この変数を、「次タイミング演奏情報ポインタ変数」と呼ぶ。ステップS602は、最初の実行であって、ステップS601により、右手パートの演奏情報検索ポインタ変数meTail[0]と左手パートの演奏情報検索ポインタ変数meTail[1]には、それぞれ各パートの最初の演奏情報へのポインタが設定されているこのため、ステップS602での次鍵の検索の処理により、曲の先頭の発音タイミングに対して、各パート毎に、各指においてその次のタイミングで押鍵されるべき各演奏情報へのポインタ変数Next[0][0]~Next[0][4]およびNext[1][0]~Next[1][4]が予め取得されることになる。この次タイミング演奏情報ポインタ変数は、ステップS605のパート毎の状態確認処理の中の準備確認の処理(後述する図9のステップS908および図12~図15参照)で参照される。なお、このステップS602の次鍵の検索の処理は、図9のステップS904またはS905の処理の後、ステップS906の処理に先立って行うようにしてもよい。

20

30

【0039】

その後、CPU201は、ステップS603で、右手パート演奏情報検索ポインタ変数meTail[0]および左手パート演奏情報検索ポインタ変数meTail[1]がそれぞれ参照する演奏情報Note[0]、Note[1]、・・・、Note[N]のうちの発音開始時刻Time(図3参照)が早い方、すなわち、右手パートと左手パートのうちの先に演奏すべき演奏情報を、検索ポインタ変数mePtrに設定しながら、ステップS604で検索ポインタ変数mePtrの内容が未定義値NULLとなってRAM203上の演奏情報の検索が全て終了したと判定するまで、以下のステップS605、S606、およびS607の一連の処理を繰り返し実行する。

40

【0040】

なお、右手パート演奏情報検索ポインタ変数meTail[0]および左手パート演奏情報検索ポインタ変数meTail[1]の各内容は、ステップS605の中の押鍵検索の処理(後述する図9のステップS907および図11のS1110参照)で、現在の押鍵タイミングに対応して各パート毎に同時に押鍵すべき演奏情報が検索される毎に、それらの同時に押鍵すべき演奏情報の次に押鍵すべき演奏情報へのポインタとして、順次更新される。

50

【 0 0 4 1 】

まず、CPU 201は、パート毎の状態確認の処理を実行する（ステップS 6 0 5）。ここでは、CPU 201は、図1の操作状態推定手段102の機能に対応する処理を実行する。この結果、右手パート出力バッファ用演奏情報ポインタ変数

```
prep[0][0] ~ prep[0][4]
```

（これら4つの変数は右手パートの親指から小指に対応）に、右手パートのそれぞれの指で押鍵を行う演奏情報と押鍵の準備動作を行う演奏情報へのポインタが得られる。同様に、左手パート出力バッファ用演奏情報ポインタ変数

```
prep[1][0] ~ prep[1][4]
```

（これら4つの変数は左手パートの親指から小指に対応）に、左手パートのそれぞれの指で押鍵を行う演奏情報と押鍵の準備動作を行う演奏情報へのポインタが得られる。

10

【 0 0 4 2 】

次に、CPU 201は、パート同士の状態比較の処理を実行する（ステップS 6 0 6）。ここでは、CPU 201は、図1のパート同士の状態比較手段103の機能に対応する処理を実行する。ここでは、CPU 201は、前述したように、左右パート間で、押鍵を行う指同士が衝突しそうになったり、あるいは次の押鍵のために押鍵の準備動作を行う指と押鍵を行う指が衝突しそうになったりする状態を検知する。そして、CPU 201は、右手パート鍵盤奥行き方向状態補正值`zOffset[0]`と左手パート鍵盤奥行き方向状態補正值`zOffset[1]`、および右手パート鍵盤横方向状態補正值`xOffset[0]`と左手パート鍵盤横方向状態補正值`xOffset[1]`を、それぞれ生成する。右手パート鍵盤奥行き方向状態補正值`zOffset[0]`は、現在のタイミングで右手パートについて押鍵を行う指または押鍵の準備動作を行う指の鍵盤上での演奏位置を、衝突の検知状態に応じて鍵盤の奥行き方向で補正するための値である。同様に、左手パート鍵盤奥行き方向状態補正值`zOffset[1]`は、現在のタイミングで左手パートについて押鍵を行う指または押鍵の準備動作を行う指の鍵盤上での演奏位置を、衝突の検知状態に応じて鍵盤の奥行き方向で補正するための値である。一方、右手パート鍵盤横方向状態補正值`xOffset[0]`は、現在のタイミングで右手パートについて押鍵を行う指または押鍵の準備動作を行う指の鍵盤上での配置位置を、衝突の検知状態に応じて鍵盤の横方向で補正するための値である。同様、左手パート鍵盤横方向状態補正值`xOffset[1]`は、現在のタイミングで左手パートについて押鍵を行う指または押鍵の準備動作を行う指の鍵盤上での配置位置を、衝突の検知状態に応じて鍵盤の横方向で補正するための値である。

20

30

【 0 0 4 3 】

最後に、CPU 201は、動作データ出力の処理を実行する（ステップS 6 0 7）。動作データ出力の処理では、CPU 201は、図1の操作状態推定手段102の機能に対応する処理を実行する。この結果、CPU 201は、図4で説明したデータフォーマットで、演奏ガイドを実行するための動作データ（図1の「動作情報108」に対応）を出力する。

【 0 0 4 4 】

CPU 201は、以上のステップS 6 0 5、S 6 0 6、およびS 6 0 7の一連の処理を、演奏情報の発音タイミング毎に実行することにより、各動作タイミングにおける演奏ガイドのための動作データを一括して出力することが可能となる。

40

【 0 0 4 5 】

図7は、図6のステップS 6 0 2の次鍵の検索の処理の例を示すフローチャートである。

【 0 0 4 6 】

CPU 201はまず、パート検索用変数`iH`に値0を設定する（ステップS 7 0 1）。パート検索用変数`iH`は、右手または左手のうちどちらを処理するかを示す情報であり、その値が0なら右手、1なら左手を示す。その後、CPU 201は、ステップS 7 0 7でパート検索用変数`iH`の値を+1しながら、ステップS 9 0 2で変数`iH`の値が2に達したと判定するまで、変数`iH`が示す右手および左手の各パート毎に、以下のステップS 7

50

03からS706の一連の処理を実行する。これにより、CPU201は、演奏ガイド開始時の押鍵タイミングに対する、右手パート次タイミング演奏情報ポインタ変数Next[0][0]~Next[0][4]および左手パート次タイミング演奏情報ポインタ変数Next[1][0]~Next[1][4]を設定する。

【0047】

次に、CPU201は、指番号指定変数ifigに値「0」を格納する（ステップS703）。

【0048】

次に、CPU201は、ステップS706で指番号指定変数ifigの値を+1ずつしながら、ステップS704で変数ifigの値が片手の指の最大数「5」に達したと判定するまで、ステップS705で、変数ifigが示す指に対応する次タイミング演奏情報ポインタ変数Next[iH][ifig]を更新する処理を実行する。ここで、ifig=0ならば親指、ifig=1ならば人差し指、ifig=2ならば中指、ifig=3ならば薬指、ifig=4ならば小指が、それぞれ指定されたことになる。

【0049】

図8は、図7のステップS705のNext[iH][ifig]の更新処理の例を示すフローチャートである。

【0050】

まず、CPU201は、現在のパートiHおよび現在の指番号ifigが示す指に対応する次タイミング演奏情報ポインタ変数Next[iH][ifig]に未定義値NULLを格納する（ステップS801）。

【0051】

次に、CPU201は、演奏情報検索用変数iに、現在のパートiHに対応する演奏情報検索ポインタ変数meTail[iH]に+1した値を格納する（ステップS802）。これは、現在のパートの演奏情報の次の演奏情報から検索が行われるという意味である。

【0052】

次に、CPU201は、ステップS806で演奏情報検索用変数iの値を+1ずつしながら、ステップS803で変数iの値が演奏情報の最大数（図3の例では、演奏情報の最大数=N）に達したと判定するまで、各演奏情報Note[i]（現在の演奏情報のインデックス番号+1 i N-1）毎に、以下のステップS804とS805の判定処理を実行する。

【0053】

すなわち、CPU201はまず、演奏情報検索用変数iが示す演奏情報Note[i]中の右手または左手を示すトラック番号Hand（図3参照）（図8中では「Note[i].Hand」と記載）が、パート検索用変数iHの値が示す現在処理中のパート番号に等しいか否かを判定する（ステップS804）。なお、図8中の「==」は、その両辺の値同士が「等しい」ことを検査する演算子である。

【0054】

ステップS804の判定がNOならば、演奏情報検索用変数iが示す演奏情報Note[i]は、現在処理対象としている手（右手または左手）以外で演奏する音符であるので、CPU201は、ステップS806の処理に移行して、次の演奏情報の検索に進む。

【0055】

ステップS804の判定がYESならば、CPU201はさらに、演奏情報検索用変数iが示す演奏情報Note[i]中の指番号Finger（図3参照）（図8中では「Note[i].Finger」と記載）が、現在処理中の指番号ifig（図7のステップS703、S704、S706参照）に等しいか否かを判定する（ステップS805）。

【0056】

ステップS805の判定がNOならば、演奏情報検索用変数iが示す演奏情報Note

10

20

30

40

50

[i] は、現在処理対象としている指以外で演奏する音符であるので、CPU 201 は、ステップ S 806 の処理に移行して、次の演奏情報の検索に進む。

【0057】

ステップ S 805 の判定が YES になると、CPU 201 は、現在のパート番号 i H および現在の指番号 i f i g が示す指に対応する次タイミング演奏情報ポインタ変数 Next [i H] [i f i g] に、トラック番号（パート番号）と指番号が一致して検索された演奏情報 Note [i] へのポインタ値（記憶アドレス等）を格納して、次タイミング演奏情報ポインタ変数 Next [i H] [i f i g] の更新処理を終了する。

【0058】

ステップ S 804 または S 805 の判定が NO の状態が続き、やがて全ての演奏情報についての検索が終了してステップ S 803 の判定が NO になると、CPU 201 は、図 8 のフローチャートで示される、i f i g で表される指に関する次タイミング演奏情報ポインタ変数 Next [i H] [i f i g] の内容がステップ S 801 で NULL に設定されたまま、その更新処理を終了する。そしてこの図 8 処理は、図 7 で、i f i g による指の指示を順次変えながら繰り返して実行されて、現在処理中のパート（右手・左手）の全ての指について順次処理が行われる。

【0059】

以上のようにして、図 7 と図 8 のフローチャートで示される処理の例によって、図 6 のステップ S 602 の次鍵の検索の処理が実現される。

【0060】

図 9 は、図 6 のステップ S 605 のパート毎の状態確認の処理の例を示すフローチャートである。

【0061】

CPU 201 はまず、パート検索用変数 i H に値 0 を設定する（ステップ S 901）。前述したように、パート検索用変数 i H の値は、0 なら右手、1 なら左手を示す。その後、CPU 201 は、ステップ S 909 でパート検索用変数 i H の値を + 1 しながら、ステップ S 902 で変数 i H の値が 2 に達したと判定するまで、変数 i H が示す右手および左手の各パート毎に、以下のステップ S 903 から S 908 の一連の処理を実行して、動作データ（図 4 参照）を一括して生成する。

【0062】

まず、CPU 201 は、検索ポインタ変数 me P t r の値が現在のパート i H に対応する演奏情報検索ポインタ変数 me T a i l [i H] の値と等しいか否かを判定する（ステップ S 903）。

【0063】

本実施形態では、パート検索用変数 i H の値の制御（ステップ S 901、S 902、S 909）によって、右手パート（i H = 0）における押鍵の状態または押鍵の準備動作の状態を検出する処理と左手パート（i H = 1）における押鍵の状態または押鍵の準備動作の状態を検出する処理が常にペアで処理される。これは、後述するように、本実施形態では、図 6 のステップ S 606 のパート同士の状態比較の処理において、右手パートの演奏状態と左手パートの演奏状態を比較して衝突が発生するかどうかを判定する必要があるためである。

【0064】

そして、前述したように、図 6 のステップ S 603 において、検索ポインタ変数 me P t r には、右手パート演奏情報検索ポインタ変数 me T a i l [0] と左手パート演奏情報検索ポインタ変数 me T a i l [1] のうち、各ポインタが示す演奏情報の発音開始時刻 Time（図 3 参照）が早い方の演奏情報へのポインタが格納される。なお、左右パート同時に発音される場合には、me P t r = me T a i l [0] = me T a i l [1] となっている。

【0065】

me P t r = me T a i l [i H] となってステップ S 903 の判定が YES なら、

10

20

30

40

50

CPU201は、現在のパートiHに対応する押鍵動作フラグ変数pushKey[iH]に値「1」を設定する。一方、mePtr meTail[iH]となってステップS903の判定がNOなら、pushKey[iH]に値「0」を設定する。押鍵動作フラグ変数pushKey[iH]は、右手パートと左手パートのうち早く押鍵または押鍵の準備動作のガイド（以下「押鍵動作」と記載する）が開始されたほうのタイミングで、現在のパートiHがその早いほうの押鍵動作のタイミングであるか（値「1」）、否か（値「0」）を示すフラグである。図17のフローチャートの説明で後述するように、CPU201は、図6のステップS606のパート同士の状態比較の処理において、右手パートと左手パートの比較を行うときに、右手パートおよび左手パートのうちどちらが早く押鍵動作に入ったかを判別することにより、適切な衝突回避の制御を行う。この判別のために、CPU201は、ステップS903で設定された押鍵動作フラグ変数pushKey[iH]の値を参照する。

10

【0066】

ステップS904またはS905の処理の後、CPU201は、離鍵確認の処理を実行する（ステップS906）。なお、ここで、図6のステップS602の「次鍵の検索」の処理をステップS906の処理に先立って行うこととしてもよい。離鍵確認の処理では、押鍵対象演奏情報ポインタ変数Hand[iH][0]~Hand[iH][4]が示す現在のパートiHの5本の各指の押鍵状態を示す演奏情報のうち、発音期間（発音開始時刻Time+発音継続時間Gate）（図3参照）が、演奏情報検索ポインタ変数meTail[iH]が示す現在のパートiHで最も早く押鍵が行われる演奏情報の発音開始時刻Timeよりも前に終了する指があれば、その指の押鍵状態が終了させられる。

20

【0067】

次に、CPU201は、押鍵検索の処理を実行する（ステップS907）。押鍵検索の処理では、演奏情報検索ポインタ変数meTail[iH]が示す現在のパートiHで最も早く押鍵が行われる演奏情報と同時に押鍵を行うべき現在のパートiHに属する演奏情報が検索される。そして、その検索された演奏情報がmeTail[iH]が示す演奏情報とともに、各演奏情報に登録されている指に対応する押鍵対象演奏情報ポインタ変数Hand[iH][0]~Hand[iH][4]にセットされる。また、次タイミング演奏情報ポインタ変数Next[iH][0]~Next[iH][4]のうち、上記セットが行われた指に対応するポインタの内容が、そのセットが行われた指においてその次のタイミングで押鍵されるべき現在のパートiHに属する演奏情報へのポインタ値に更新される。なお、同じパートに属し同時に発音開始される演奏情報はRAM203上では連続して記憶されている。このため、上記押鍵検索の処理により同時に発音開始される演奏情報が検索された後は、演奏情報検索ポインタ変数meTail[iH]の内容は、それらの同時に押鍵すべき演奏情報を飛び越して、その次に押鍵すべき演奏情報へのポインタとして設定される。

30

【0068】

続いて、CPU201は、準備確認の処理を実行する（ステップS908）。この結果、演奏情報検索ポインタ変数meTail[iH]が示す演奏情報中の発音開始時刻Time（図3参照）に対応する押鍵タイミングにおいて、パートiHの5本の各指で押鍵または押鍵の準備動作を行う演奏情報へのポインタが、出力バッファ用演奏情報ポインタ変数prep[iH][0]~prep[iH][4]（これら4つの変数はパートiHの親指から小指に対応）に得られる。この処理が、ステップS901、S902、およびS909で制御されるパート検索用変数iHの値「0」および「1」毎に実行されることにより、右手パートに対応する出力バッファ用演奏情報ポインタ変数prep[0][0]~prep[0][4]と、左手パートに対応する出力バッファ用演奏情報ポインタ変数prep[1][0]~prep[1][4]が算出される。

40

【0069】

上述の準備確認の処理では、演奏情報が順次読み込まれながら、曲の各再生時点において、再生時点における演奏情報に対応する演奏状態とその再生時点以降における演奏情報

50

に対応する演奏状態とが比較される。この結果、例えば、親指で押鍵中に親指以外の指で親指の上を越えて移動させて次以降のタイミングにて押鍵する第1の状態（指越え）が検出される。または、親指以外の指で押鍵中に親指で親指以外の指の下をくぐらせて移動させて次以降のタイミングにて押鍵する第2の状態（指くぐり）が検出される。また、第1および第2のいずれの状態でもなく、任意の指で押鍵中の鍵から所定の基準値に比較して近い位置の異なる鍵を次以降のタイミングにてその任意の指以外の指で押鍵する第3の状態が検出される。あるいは、任意の指で押鍵中の鍵と同じ鍵を次のタイミングにてその任意の指以外の指で押鍵する第4の状態が検出される。さらには、現在押鍵されている鍵から所定の基準値よりも離れた位置の鍵が次に押鍵される状態が検出される。

【0070】

10

以上のステップS906の離鍵確認の処理、ステップS907の押鍵検索の処理、ステップS908の準備確認の処理は、図1の操作状態推定手段102の機能を実現する。

【0071】

図10は、図9のステップS906の離鍵確認の処理の例を示すフローチャートである。

【0072】

まず、CPU201は、指番号指定変数*i f i g*に値「0」を格納する（ステップS1001）。

【0073】

次に、CPU201は、ステップS1006で指番号指定変数*i f i g*の値を+1ずつしながら、ステップS1002で変数*i f i g*の値が片手の指の最大数「5」に達したと判定するまで、ステップ1003からS1005の一連の処理を実行する。なお、変数*i f i g*の各値の意味は、図7の場合と同じである。

20

【0074】

CPU201は、現在のパート*i H*（ステップS901、S902、S909参照）および現在の指番号*i f i g*が示す押鍵対象演奏情報ポインタ変数*H a n d [i H] [i f i g]*の値が未定義値*N u l l*でないか否かを判定する（ステップS1003）。なお、図10中の「!=」は、その両辺の値同士が「等しくない」ことを検査する演算子である。前述のように、押鍵対象演奏情報ポインタ変数*H a n d [i H] [i f i g]*は、現在のパート*i H*の*i f i g*で表される指が押鍵中である場合の演奏情報へのポインタ変数であるので、この*H a n d [i H] [i f i g]*の値が*N u l l*でない場合は、その指が押鍵中であることを意味する。

30

【0075】

ステップS1003の判定がNOならば、*H a n d [i H] [i f i g]*の値が*N u l l*でありその指は押鍵中では無いので、CPU201は、ステップS1006の処理に移行し、次の指の処理に移行する。

【0076】

押鍵対象演奏情報ポインタ変数*H a n d [i H] [i f i g]*の値が未定義値でなくステップS1003の判定がYESならば、CPU201は、ポインタ変数*H a n d [i H] [i f i g]*が指す演奏情報の発音開始時刻*T i m e*（図3参照）にその演奏情報の発音継続時間*G a t e*を加算することにより、その演奏情報の発音終了時刻を算出する。そして、CPU201は、ポインタ変数*H a n d [i H] [i f i g]*が指す演奏情報の上記発音終了時刻が、演奏情報検索ポインタ変数*m e T a i l [i H]*が示す現在のパート*i H*で最も早く押鍵が行われる演奏情報の発音開始時刻*T i m e*よりも前である（小さい）か否かを判定する（ステップS1004）。

40

【0077】

ポインタ変数*H a n d [i H] [i f i g]*が指す演奏情報の発音終了時刻が*m e T a i l [i H]*が示す演奏情報の発音開始時刻*T i m e*よりも前でステップS1004の判定がYESならば、CPU201は、ポインタ変数*H a n d [i H] [i f i g]*に未定義値*N U L L*を格納して、現在の指番号*i f i g*が示す指を離鍵状態にする（ステップS

50

1005)。

【0078】

その後、CPU201は、ステップS1006の処理に移行し、次の指の処理に移行する。

【0079】

5本全ての指についての確認が終了してステップS1002の判定がNOになると、CPU201は、図10のフローチャートで示される図9のステップS906の離鍵確認の処理を終了する。

【0080】

以上のようにして、図10のフローチャートの処理の例で示される図9のステップS906の離鍵確認の処理によって、押鍵対象演奏情報ポインタ変数Hand[iH][0]~Hand[iH][4]が示す現在のパートiHの各指で押鍵中とされている演奏情報のうち、現在処理中のmeTail[iH]が示す現在のパートiHで最も早く押鍵が行われる演奏情報の発音開始時刻よりも発音終了時刻が前となる指について、離鍵状態とされる。

【0081】

図11は、図9のステップS907の押鍵検索の処理の例を示すフローチャートである。

【0082】

CPU201はまず、演奏情報検索用変数iに、現在の処理対象のmeTail[iH]が示す現在のパートiHで最も早く押鍵が行われる演奏情報のインデックス番号index(図3参照)の値をセットする。つまり、同時発音を行う演奏情報を検索するための開始位置が、現在処理対象の演奏情報自身とされる(ステップS1101)。

【0083】

次に、CPU201は、ステップS1108で演奏情報検索用変数iの値を+1ずつしながら、ステップS1102で変数iの値が演奏情報の最大数(図3の例では、演奏情報の最大数=N)に達したと判定するまで、ステップS1103からS1107までの一連の処理を実行する。

【0084】

CPU201はまず、演奏情報検索用変数iが示す演奏情報Note[i]中の右手または左手を示すトラック番号Hand(図3参照)(図8中では「Note[i].Hand」と記載)が、パート検索用変数iHの値が示す現在のパート(図9のステップS901、S902、S912参照)に等しいか否かを判定する(ステップS1103)。

【0085】

ステップS1103の判定がNOならば、演奏情報検索用変数iが示す演奏情報Note[i]は、現在処理中のパート(右手・左手の別)ではないので、CPU201は、ステップS1108の処理に移行して、次の演奏情報の検索に進む。

【0086】

ステップS1103の判定がYESならば、CPU201は、演奏情報検索用変数iが示す演奏情報Note[i]中の発音開始時刻Time(図3参照)(図11中では「Note[i].Time」と記載)が、現在処理対象の演奏情報検索ポインタ変数meTail[iH]が示す演奏情報の発音開始時刻Time(図3参照)(図11中では「meTail[iH].Time」と記載)と同時刻であるか否かを判定する(ステップS1104)。

【0087】

同時刻の演奏情報が見つかりステップS1104の判定がYESならば、CPU201は、指番号指定変数ifigに、現在検索されている演奏情報Note[i]中の指番号Finger(図11中では「Note[i].Finger」と記載)を格納する(ステップS1105)。なお、最初の処理では、「i=meTail[iH]のインデックス番号」であるから(ステップS1101参照)、ステップS1104の判定は、「No

10

20

30

40

50

`te[i].Time = meTail[iH].Time == meTail[iH].Time?`」という自分自身を自分自身と比較する判定になる。このため、ステップS1104の判定は、必ずYESとなる。

【0088】

次に、CPU201は、検索された演奏情報`Note[i]`の指番号`Finger`（図3参照）を、現在の指番号`ifig`に格納する（ステップS1105）。

【0089】

その後、CPU201は、現在の指番号`ifig`が示す押鍵対象演奏情報ポインタ変数`Hand[iH][ifig]`に、検索された演奏情報`Note[i]`へのポインタを格納する（ステップS1106）。前述したように、最初の処理では、ステップS1105の判定は必ずYESとなるため、現在処理対象の演奏情報検索ポインタ変数`meTail[iH]`が示す演奏情報へのポインタは、必ず押鍵対象演奏情報ポインタ変数`Hand[iH][ifig]`に登録されることになる。

【0090】

最後に、CPU201は、検索された演奏情報`Note[i]`に対応する指番号`ifig`（ステップS1105参照）が示す指に対応する次タイミング演奏情報ポインタ変数`Next[iH][ifig]`を更新する処理を実行する（ステップS1107）。この処理の例は、前述した図8のフローチャートで示される。ただし、図8のステップS802では、CPU201は、演奏情報検索用変数`i`に、検索された現在の演奏情報のカウンタ値`i`に+1した値を格納する。これは、検索された同時発音を行う現在の演奏情報`Note[i]`の次の演奏情報から、次に押鍵される演奏情報の検索が行われるという意味である。

【0091】

ステップS1107の処理の後、CPU201は、ステップS1108の処理に移行して、次の演奏情報の検索に進む。

【0092】

ここで、図3で説明したように、演奏情報`Note[0]`、`Note[1]`、・・・、`Note[N]`は、発音開始時刻`Time`の順に並ぶように、インデックス番号（0、1、・・・、N）が付与されている。従って、ステップS1104の判定で、現在の演奏情報検索用変数`i`が示す演奏情報`Note[i]`中の発音開始時刻`Time`が、現在処理対象の`meTail[iH]`が示す演奏情報の発音開始時刻`Time`と同時刻ではなく、判定結果がNOならば（ステップS1104の判定がNO）、現在の演奏情報検索用変数`i`が示す演奏情報`Note[i]`以降には同時刻の演奏情報は存在しない。そこでこの場合には、CPU201は、押鍵検索の処理を終了する。この場合、同時発音を行うとして検索された演奏情報については、図9のステップS908では、現在の処理対象の`meTail[iH]`が示す演奏情報とともに、押鍵対象演奏情報ポインタ変数`Hand[iH][0]~Hand[iH][4]`にセットされて一括して処理される。このため、以降は図9のステップS902からS909までの処理ループを実行する必要はない。このため、CPU201は、現在のパート`iH`の処理対象の演奏情報検索ポインタ変数`meTail[iH]`に、同時発音を行う演奏情報のグループを飛び越えて、最後に捨てS1104の判定がNOとなったときの演奏情報検索用変数`i`が示す演奏情報`Note[i]`へのポインタを設定する（ステップS1110）。

【0093】

ただし、このステップS1110の処理が実行されるのは、現在のパート`iH`が他方のパートと同じかそれよりも早い押鍵動作のタイミングであって現在のパート`iH`に対応する押鍵動作フラグ変数`pushKey[iH]`に値「1」が設定されている場合のみとする（ステップS1109の判定がYES）。現在のパート`iH`が他方のパートよりも遅い押鍵動作のタイミングであって現在のパート`iH`に対応する押鍵動作フラグ変数`pushKey[iH]`に値「0」が設定されている場合には、ステップS1109の判定はNOとなる。この場合には、現在のパート`iH`の処理対象の演奏情報検索ポインタ変数`meT`

a i l [i H] の内容は更新されずに、図 11 のフローチャートで示される図 9 のステップ S 907 の押鍵検索の処理を終了する。後述するように、図 6 のステップ S 607 の動作データ出力の処理では、p u s h K e y [i H] に値「0」が設定されていて右手パートと左手パートの衝突が発生していないときには動作データ出力の処理は実行されない(図 18 のステップ S 1803 の判定が NO となる)。このため、p u s h K e y [i H] に値「0」が設定されているときには、図 11 のステップ S 1109 の判定が NO となって演奏情報検索ポインタ変数 m e T a i l [i H] の内容は更新されず、そのパート i H に関しては、図 5 のステップ S 603 で p u s h K e y [i H] に値「1」が設定されるまで m e T a i l [i H] の更新による演奏ガイドの表示進行が待たれることになる。

【0094】

10

ステップ S 1110 の処理の後、C P U 201 は、図 11 のフローチャートで示される図 9 のステップ S 907 の押鍵検索の処理を終了する。

【0095】

また、全ての演奏情報についての検索が終了してステップ S 1102 の判定が NO になった場合も、C P U 201 は、図 11 のフローチャートで示される図 9 のステップ S 907 の押鍵検索の処理を終了する。

【0096】

図 12 から図 15 は、図 9 のステップ S 908 の準備確認の処理の例を示すフローチャートである。

【0097】

20

この図 12 から図 15 の準備確認の処理を通して、p r e p [i H] [0] ~ p r e p [i H] [4] には、図 6 のステップ S 607 の動作データ出力の処理において図 4 の動作データを出力するために対象となる現在のパート i H の各指の演奏情報が格納される。この場合の演奏情報には、対応する動作データの出力タイミングにおいて、現在のパート i H の各指で押鍵を行う演奏情報と、次の動作タイミングでの押鍵の準備動作を行う演奏情報がある。まず、C P U 201 は、図 12 のステップ S 1201 から S 1208 までの一連の処理によって、押鍵対象演奏情報ポインタ変数 H a n d [i H] [0] ~ H a n d [i H] [4] の内容を、いったん出力バッファ用演奏情報ポインタ変数 p r e p [i H] [0] ~ p r e p [i H] [4] にコピーする。このコピー処理では、まずは押鍵対象となっている演奏情報が格納されているポインタ変数 H a n d [i H] [0] ~ H a n d [i H] [4] の内容が p r e p [i H] [0] ~ p r e p [i H] [4] にコピーされる。その後、図 13 から図 15 のフローチャートでは、現在のパート i H の各指の次タイミング演奏情報ポインタ N e x t [i H] [0] ~ N e x t [i H] [4] の中から準備動作の対象となる演奏情報が判別される。そして、その判別された演奏情報が、後述する図 15 のステップ S 1239 で、p r e p [i H] [0] ~ p r e p [i H] [4] に追加的に登録される。

30

【0098】

C P U 201 はまず、準備動作判定用変数 M o v i n g 、 P a s s i n g に、共に値「0」を格納する(ステップ S 1201)。

【0099】

40

次に、C P U 201 は、指番号指定変数 i f i g と押鍵指数カウンタ変数 f i g u s e に、共に値「0」を格納する(ステップ S 1202)。

【0100】

その後、C P U 201 は、ステップ S 1208 で指番号指定変数 i f i g の値を + 1 ずつしながら、ステップ S 1203 で変数 i f i g の値が片手の指の最大数「5」に達したと判定するまで、ステップ S 1204 から S 1207 までの一連の処理を実行する。

【0101】

まず、C P U 201 は、現在のパート i H の現在の指番号 i f i g に対応する出力バッファ用演奏情報ポインタ変数 p r e p [i H] [i f i g] に、共に未定義値 N U L L を格納する。また、C P U 201 は、現在の指番号 i f i g に対応するステップデータ s t

50

`ep[ifig]`に値「-1」を格納する(以上、ステップS1204)。ステップデータ`step[ifig]`は、現在の指番号`ifig`に対応する指が、どの押鍵段階にあるかを示す。`step[ifig] = 0`ならばその指を今回押鍵すべきであることを示し、`step[ifig] = 1, 2, …`ならばその指を次回、その次、…で押鍵すべきであることを示す。ステップS1204で設定される`step[ifig] = -1`は、その指についてはまだ押鍵段階が決定されていないことを示す。

【0102】

次に、CPU201は、現在のパートiHの現在の指番号`ifig`に対応する押鍵対象演奏情報ポインタ変数`Hand[iH][ifig]`の内容を出力バッファ用演奏情報ポインタ変数`prep[iH][ifig]`にコピーする(ステップS1205)。

10

【0103】

次に、CPU201は、現在の指番号`ifig`に対応する出力バッファ用演奏情報ポインタ変数`prep[iH][ifig]`の内容が未定義値NULLでないか否かを判定する(ステップS1206)。

【0104】

`prep[iH][ifig]`の内容が未定義値NULLでなくステップS1206の判定がYESならば、現在のパートiHの現在の指番号`ifig`に対応する押鍵対象演奏情報ポインタ変数`Hand[iH][ifig]`の内容がNULLではなかったということなので、CPU201は、現在の指番号`ifig`に対応するステップデータ`step[ifig]`に今回押鍵を示す値「0」を格納する。さらに、CPU201は、押鍵指数カウンタ変数`figuse`の値を+1インクリメントする(図12中では「++」という演算子で記載)(以上、ステップS1207)。

20

【0105】

`prep[iH][ifig]`の内容が未定義値NULLであってステップS1206の判定がNOである場合、またはステップS1207の処理の後、CPU201は、ステップS1208の処理に移行し、次の指に対する処理を実行する。

【0106】

以上のステップS1203からS1208までの一連の処理の後、5本の指についてのコピー処理が終了しステップS1203の判定がNOとなった時点で、押鍵対象演奏情報ポインタ変数`Hand[iH][0] ~ Hand[iH][4]`のうち押鍵を示す演奏情報へのポインタが出力バッファ用演奏情報ポインタ変数`prep[iH][0] ~ prep[iH][4]`にコピーされている。この場合、ステップデータ`step[0] ~ step[4]`のうち押鍵状態にある指に対応するデータには値「0」が格納され、それ以外の指に対応するデータには未定義状態を示す値「-1」が格納される状態となる。

30

【0107】

図12のステップS1203の判定がNOとなった後、CPU201は、図13のステップS1209からS1219の一連の処理を実行する。ここでは、現在のパートiHの各指の次タイミング演奏情報ポインタ変数`Next[iH][0] ~ Next[iH][4]`のうちから、出力バッファ用演奏情報ポインタ変数`prep[iH][0] ~ prep[iH][4]`において演奏情報へのポインタが定義されていない指に対応するデータであって、かつ発音開始時刻が最も早い演奏情報へのポインタとそれに対応する指番号が、それぞれ変数`NotePre`と`fig`に取得される。

40

【0108】

すなわち、CPU201はまず、押鍵指数カウンタ変数`figuse`の値が5より小さいか否かを判定する(ステップS1209)。

【0109】

もし、押鍵指数カウンタ変数`figuse`の値が5になっていると、現在のパートiHの5本全ての指が押鍵中を示していることになり(図12のステップS1206とS1207参照)、出力バッファ用演奏情報ポインタ変数`prep[iH][0] ~ prep[iH][4]`において演奏情報へのポインタが定義されていない指はない。この場合には

50

、図13のステップS1209の判定がNOとなって、CPU201は、図9のステップS908の準備確認の処理を終了する。この場合は、準備動作判定用変数Moving、Passingの値は図12のステップS1201で共に値「0」とされたままになる。そして、出力バッファ用演奏情報ポインタ変数prep[iH][0]~prep[iH][4]に設定されている5つの演奏情報は、全て押鍵対象演奏情報ポインタ変数Hand[iH][0]~Hand[iH][4]に設定されていた、同時刻に発音開始する5つの演奏情報がそのまま格納される。この結果、後述する図6のステップS607の動作データ出力の処理（後述する図19のステップS1815 S1816参照）において、各指の状態status[0]~status[4]（図4参照）として共に押鍵を示す値「1」が設定される。すなわち、後述する図19のステップS1817 S1818の処理は一度も実行されず、各指の状態status[0]~status[4]（図4参照）のいずれにも、準備動作を示す値「-1」は設定されないことになる。

10

【0110】

押鍵指数カウンタ変数figuseの値が5より小さくステップS1209の判定がYESならば、CPU201は、現在の押鍵比較段階変数stepに初期値「1」を格納する（ステップS1210）。押鍵比較段階変数stepの値は、現在の押鍵タイミングで押鍵中とされる演奏情報に対して何タイミング後に押鍵される演奏情報の比較が行われるかを示す。step=1であれば、現在押鍵中の演奏情報に対して次のタイミングにて押鍵される演奏情報の比較が行われることを示す。step=2であれば、現在押鍵中の演奏情報に対して次の次のタイミングにて押鍵される演奏情報の比較が行われることを示す。押鍵比較段階変数stepの値は、ステップS1209で値「1」に初期設定された後、後述する図15のステップS1240で+1ずつインクリメントされ、ステップS1211からS1219の探索が繰り返し実行される。

20

【0111】

次に、CPU201は、現在の押鍵比較段階stepにおいて、まだ押鍵や準備動作が決定されていない指のうちで発音開始時刻が最も早い演奏情報へのポインタを格納する変数NotePreに未定義値NULLを格納し、それに対応する指番号を格納する変数figにどの指でもないことを示す値「-1」を格納する（ステップS1211）。

【0112】

次に、CPU201は、ステップS1212で指番号指定変数ifigに初期値「0」（親指）を設定した後、ステップS1219でifigの値を+1ずつインクリメントしながら、ステップS1213で片手の指の最大数「5」に達したと判定するまで、以下のステップS1214からS1218の一連の処理を実行する。

30

【0113】

CPU201はまず、現在の指番号ifigに対応するステップデータstep[ifig]の値が、指番号ifigの指についてはまだ押鍵段階が決定されていないことを示す値「-1」（図12のステップS1204参照）であるか否かを判定する（ステップS1214）。

【0114】

現在の指番号ifigに対応するステップデータstep[ifig]の値が「-1」以外でありステップS1214の判定がNOならば、その指番号ifigの指は押鍵または押鍵の準備動作をすることが決定済みであるため、CPU201は、ステップS1219の処理に移行し、次の指番号の処理に進む。

40

【0115】

現在の指番号ifigに対応するステップデータstep[ifig]の値が「-1」でありステップS1214の判定がYESならば、CPU201は、現在のパートiHの現在の指番号ifigに対応する次タイミング演奏情報ポインタ変数Next[iH][ifig]の値が未定義値NULLでないか否かを判定する（ステップS1215）。

【0116】

Next[iH][ifig]の値が未定義値NULLであってステップS1215の

50

判定がNOならば、その指番号 *i f i g* に対応する指について次以降のタイミングで押鍵されることはない。このため、CPU 201は、ステップS 1 2 1 9の処理に移行し、次の指番号の処理に進む。

【0117】

Next [*i H*] [*i f i g*] の値が未定義値NULLではなくステップS 1 2 1 5の判定がYESならば、CPU 201は、まだ押鍵や準備動作が決定されていない指のうちで発音開始時刻が最も早い演奏情報へのポインタを格納する変数NotePreの内容が未定義値NULLではなく、かつ変数NotePreが示す演奏情報の発音開始時刻Time (図13中では「NotePre.Time」と記載) が現在の指番号 *i f i g* に対応する次タイミング演奏情報ポインタ変数Next [*i H*] [*i f i g*] が示す演奏情報の発音開始時刻Time (図13中では「Next [*i H*] [*i f i g*].Time」と記載) よりも早い(小さい)か否かを判定する(ステップS 1 2 1 6)。

10

【0118】

ステップS 1 2 1 6の判定がYESならば、現在の変数NotePreが示す演奏情報の発音開始時刻のほうが、現在の指番号 *i f i g* に対応する次タイミング演奏情報ポインタ変数Next [*i H*] [*i f i g*] が示す演奏情報の発音開始時刻よりも早い。この場合には、CPU 201は、変数NotePreの内容をそのまま維持し、ステップS 1 2 1 9の処理に移行し、次の指番号の処理に進む。

【0119】

ステップS 1 2 1 6の判定がNOならば、現在の変数NotePreがまだ定義されていなかった(NULL)か、または、現在のパート*i H*の現在の指番号 *i f i g* に対応する次タイミング演奏情報ポインタ変数Next [*i H*] [*i f i g*] が示す演奏情報の発音開始時刻が、現在の変数NotePreが示す演奏情報の発音開始時刻より早いことになる。そこでこの場合には、CPU 201は、Next [*i H*] [*i f i g*] に格納されているポインタ値を変数NotePreにコピーする(ステップS 1 2 1 7)。

20

【0120】

さらに、CPU 201は、現在の指番号指定変数 *i f i g* の値を変数NotePreに対応する指番号を保持する変数 *f i g* にコピーする(ステップS 1 2 1 8)。

【0121】

その後、CPU 201は、ステップS 1 2 1 9の処理に移行し、次の指番号 *i f i g* に対する処理に進む。

30

【0122】

以上のステップS 1 2 1 4からS 1 2 1 8までの一連の処理が、各指番号 *i f i g* について繰り返し実行され、やがて指番号 *i f i g* の値が5になってステップS 1 2 1 3の判定がNOとなった時点で、変数NotePreに、まだ押鍵や準備動作が決定されていない指のうちで、その指による次の押鍵が他の指と比べて、現在の押鍵比較段階*s t e p*において発音開始時刻が最も早い演奏情報へのポインタが得られる。また、変数 *f i g* に、NotePreが示す演奏情報に対応する指番号が得られる。最初は、押鍵比較段階*s t e p* = 1である(ステップS 1 2 1 0参照)。従って、変数NotePreと *f i g* には、現在の押鍵に対して次に押鍵されるべき、発音開始時刻が最も早い演奏情報へのポインタと、それに対応する指番号が得られる。

40

【0123】

ステップS 1 2 1 3の判定がNOになると、CPU 201は、図14のフローチャートのステップS 1 2 2 0からS 1 2 3 7までの一連の処理を実行する。ここでは、上述の変数NotePreに得られた演奏情報に対応する演奏状態が、出力バッファ用演奏情報ポインタ変数

```
p r e p [ i H ] [ 0 ] ~ p r e p [ i H ] [ 4 ]
```

のうち押鍵を示す演奏情報の演奏状態と比較される。

【0124】

まず、CPU 201は、変数NotePreの内容が未定義値NULLでないか否かを判定する(ステップS 1 2 2 0)。図13のステップS 1 2 1 3からS 1 2 1 9の実行状

50

態によっては、変数 `NotePre` の内容が図 13 のステップ `S 1 2 1 1` で未定義値 `NUL L` に設定された後、全ての指番号 `i f i g` について、図 13 のステップ `S 1 2 1 6` の判定が `Y E S` にならず、ステップ `S 1 2 1 3` の判定が `N O` になる場合があり得る。この場合には、変数 `NotePre` には発音開始時刻が最も早い演奏情報へのポイントが得られない状態となり、`NotePre` の値は未定義値 `NUL L` となる。これによりステップ `S 1 2 2 0` の判定が `N O` となった場合には、`C P U 2 0 1` は、図 9 のステップ `S 9 0 8` の準備確認の処理を終了する。

【 0 1 2 5 】

変数 `NotePre` の内容が未定義値 `NUL L` でなくステップ `S 1 2 2 0` の判定が `Y E S` ならば、`C P U 2 0 1` は、変数 `NotePre` の内容出力バッファ用演奏情報ポインタ変数 `pre p [i H] [0] ~ pre p [i H] [4]` のいずれかへ登録するか否かを示すセット変数 `set` に、登録を示す値「1」をひとまず格納する（ステップ `S 1 2 2 1`）。このセット変数 `set` の値は、後述する各種演奏状態の判断において非登録を示す値「0」にされ得る（図 14 のステップ `S 1 2 3 0`、`S 1 2 3 3`、`S 1 2 3 6`）。

【 0 1 2 6 】

次に、`C P U 2 0 1` は、現在の押鍵に対して次に押鍵されるべき発音開始時刻が最も早い演奏情報の指番号 `f i g` に対応するステップデータ `step [i f i g]` に、現在の押鍵比較段階変数 `step` の値（初回は値「1」）を格納する（ステップ `S 1 2 2 2`）。これは、2 回目以降に実行される図 13 のステップ `S 1 2 1 4` の判定において、すでに処理された指番号については再度その判定が `Y E S` となって選択されてしまわないようにするための制御である。また、ステップデータ `step [0] ~ step [4]` の各値を見れば、各指が現在の押鍵タイミング以降の何番目に押鍵されるべき演奏情報であるかを知ることができ、演奏ガイドの制御に活用することも可能になるというメリットも有する。

【 0 1 2 7 】

次に、`C P U 2 0 1` は、ステップ `S 1 2 2 3` で指番号指定変数 `i f i g` に初期値「0」（親指）をセットした後、ステップ `S 1 2 3 7` で `i f i g` の値を + 1 ずつインクリメントしながら、ステップ `S 1 2 2 4` で指番号 `i f i g` の値が片手の指の最大数「5」に達したと判定するまで、以下のステップ `S 1 2 2 5` から `S 1 2 3 6` までの一連の処理を実行する。この処理では、図 13 のステップ `S 1 2 1 4` から `S 1 2 1 8` の一連の処理で得られた、まだ押鍵や準備動作が決定されていない指による次の押鍵情報に対して、現在押鍵している指（音）との関係がチェックされる。

【 0 1 2 8 】

`C P U 2 0 1` はまず、指番号 `i f i g` の値が現在の押鍵に対して次に押鍵されるべき発音開始時刻が最も早い演奏情報の指番号 `f i g` に等しくなく、かつ現在のパート `i H` の現在の指番号 `i f i g` に対応する出力バッファ用演奏情報ポインタ変数 `pre p [i H] [i f i g]` の値が未定義値 `NUL L` でなく、かつ指番号 `i f i g` に対応するステップデータ `step [i f i g]` が押鍵を示す値「0」に等しいか否か（図 12 のステップ `S 1 2 0 7` 参照）を判定する（ステップ `S 1 2 2 5`）。一番目の条件は、同じ指同士の関係は調べる必要が無いので、`i f i g` と `f i g` が異なっていることをチェックしている。二番目の条件は、調査対象とする指番号 `i f i g` の演奏情報がまだ決定されておらず、`pre p [i H] [i f i g]` が `NUL L` の場合は、そのような指 `i f i g` との関係性を調べる必要が無いからである。三番目の条件は、調査対象とする指番号 `i f i g` が押鍵されている指かどうかを調べているものである。この判定は、現在のパート `i H` の現在の指番号 `i f i g` に対応する出力バッファ用演奏情報ポインタ変数 `pre p [i H] [i f i g]` が示す演奏情報が押鍵を示す演奏情報であるか否かの判定である。

【 0 1 2 9 】

ステップ `S 1 2 2 5` の判定が `N O` ならば、`pre p [i H] [i f i g]` は押鍵を示す演奏情報を示してはいないため、ステップ `S 1 2 3 7` の処理に移行し、次の指番号の処理に進む。

【 0 1 3 0 】

`prep[iH][ifig]` が示す演奏情報が押鍵を示す演奏情報であってステップ `S1225` の判定が `YES` ならば、`CPU201` は、`prep[iH][ifig]` が示す押鍵の演奏情報と、変数 `NotePre` が示す、まだ押鍵や準備動作が決定されていない指のうちで現在の押鍵に対して次に押鍵されるべき発音開始時刻が最も早い演奏情報と、の鍵盤距離を取得する（ステップ `S1226`）。鍵盤距離は、指番号が大きいほうの演奏情報の音高 `Pitch`（図3参照）を鍵盤番号に換算した値から、指番号が小さいほうの演奏情報の音高 `Pitch` を鍵盤番号に換算した値を減算した結果として算出される。この処理の詳細は、図16のフローチャートを用いて後述する。

【0131】

次に、`CPU201` は、ステップ `S1226` で取得した鍵盤距離が0よりも小さいか否かを判定する（ステップ `S1227`）。 10

【0132】

鍵盤距離が0よりも小さくステップ `S1227` の判定が `YES` ならば、現在関係を調査している2つの指の位置関係が、正常な手の向きにおける指の位置と逆転していることを表している。そこでこの場合、`CPU201` は、現在の押鍵に対して次に押鍵されるべき発音開始時刻が最も早い演奏情報の指番号 `fig` と現在の押鍵の指番号 `ifig` のいずれかが親指の指番号「0」であるか否かを判定する（ステップ `S1228`）。すなわち、現在関係を調査している2つの指のいずれかが、親指であるかを調べる。

【0133】

ステップ `S1228` の判定が `YES` の場合には、`CPU201` は、親指で押鍵中に親指以外の指で親指の上を越えて移動させて次以降のタイミングにて押鍵する第1の状態である、または、親指以外の指で押鍵中に親指で親指以外の指の下をくぐらせて移動させて次以降のタイミングにて押鍵する第2の状態であると判定する。 20

【0134】

この場合には、`CPU201` は、準備動作判定用変数の一つである指の交差を示す指交差変数 `Passing` に値「1」を格納する（ステップ `S1229`）。

【0135】

その後、`CPU201` は、ステップ `S1237` の処理に移行し、次の指番号を有する押鍵の演奏情報との比較処理に進む。

【0136】 30

ステップ `S1228` の判定が `NO` の場合には、`CPU201` は、親指以外の指同士の交差はあり得ないとして、変数 `NotePre` の内容を出力バッファ用演奏情報ポインタ変数 `prep[iH][0] ~ prep[iH][4]` のいずれかへ登録するか否かを示すセット変数 `set` に、非登録を示す値「0」を格納する（ステップ `S1230`）。すなわち、変数 `NotePre` が示す演奏情報は、現在押鍵している指との関係上、指を交差させないと準備ができない音であることが判ったので、この音については、準備動作をする演奏情報としては登録されない。

【0137】

その後、`CPU201` は、ステップ `S1237` の処理に移行し、次の指番号を有する押鍵の演奏情報との比較処理に進む。 40

【0138】

鍵盤距離が0よりも小さくなくステップ `S1227` の判定が `NO` ならば、`CPU201` は、ステップ `S1226` で取得した鍵盤距離が基準値よりも大きいか否かを判定する（ステップ `S1231`）。基準値は例えば、指の間隔×半音での2～4鍵分程度の値である。例えば、親指と小指の間隔としては、指の間隔（4 - 0）×2～4半音で、8半音～16半音程度とすることができ、12半音とすれば1オクターブを基準とすることとなる。

【0139】

鍵盤距離が基準値よりも大きくステップ `S1231` の判定が `YES` ならば、`CPU201` は、鍵盤を弾く手の移動範囲が大きいと判断して、セット変数 `set` に、非登録を示す値「0」を格納する（ステップ `S1232`）。すなわち、変数 `NotePre` が示す演奏 50

情報は、準備動作をする演奏情報としては登録されない。

【0140】

次に、CPU201は、押鍵比較段階変数stepの値が「1」であるか否かを判定する(ステップS1233)。

【0141】

ステップS1233の判定がYES、すなわち鍵盤の移動が現在の押鍵の次の動作タイミングで実行されると判定された場合には、鍵盤距離が基準値よりも大きい音が、現在の押鍵のすぐ次の動作タイミングで実行されることから、指の跳躍が必要となる場合である。そこで、CPU201は、準備動作判定用変数の一つである指の手の移動(跳躍)を示す移動変数Movingに値「1」を格納する(ステップS1234)。

10

【0142】

このように、変数NotePreが示す演奏情報は、準備動作をする演奏情報としては登録されないが、移動変数Movingと一時バッファ用演奏情報ポインタ変数plist[NotePre.Finger]へのセットが行われる。

【0143】

鍵盤距離が基準値よりも大きくはなくステップS1231の判定がNOならば、CPU201は、ステップS1226で取得した鍵盤距離が0で、かつ押鍵比較段階変数stepの値が「1」ではないか否かを判定する(ステップS1235)。この判定がYESとなる場合は、任意の指で押鍵中の鍵と同じ鍵を、現在押鍵中の指とは異なる指で、次ではない、より先のタイミングで押鍵することを表している。

20

【0144】

ステップS1235の判定がNOの場合には、CPU201は、セット変数setに登録を示す値「1」が格納された状態を維持する。すなわち、変数NotePreが示す演奏情報は、準備動作をする演奏情報として登録可能な状態にされる。その後、CPU201は、ステップS1237の処理に移行し、次の指番号を有する押鍵の演奏情報との比較処理に進む。

【0145】

任意の指で押鍵中の鍵と同じ鍵を、次の次以降のタイミングにてその任意の指以外の指で押鍵する状態であってステップS1235の判定がYESとなる場合は、CPU201は、セット変数setに非登録を示す値「0」を格納する(ステップS1236)。すなわち、変数NotePreが示す演奏情報は、準備動作をする演奏情報としては登録されない。

30

【0146】

その後、CPU201は、ステップS1237の処理に移行し、次の指番号を有する押鍵の演奏情報との比較処理に進む。

【0147】

CPU201は、以上のステップS1225からS1236までの一連の処理を、0から4までの指番号ifigについて実行する。これにより、CPU201は、変数NotePreに得られた演奏情報に対応する演奏状態を、出力バッファ用演奏情報ポインタ変数prep[iH][0]~prep[iH][4]のうち押鍵を示す演奏情報の演奏状態と比較する。そして、CPU201は、NotePreが示す演奏情報へのポインタを、準備動作をする演奏情報へのポインタとしてprep[iH][0]~prep[iH][4]に登録可能かどうかを判別し、セット変数setへの値のセットを行う。

40

【0148】

現在の変数NotePreの演奏情報について、0から4までの全ての指番号ifigに対応する上述の制御動作が終了し、ステップS1224の判定がNOになると、CPU201は、図15のフローチャートの制御処理を実行する。

【0149】

図15において、CPU201は、セット変数setに値「1」がセットされているか否か、つまり登録可能であると判定されているか否かを判定する(ステップS1238)

50

。

【0150】

ステップS1238の判定がYESならば、CPU201は、変数NotePreが示す演奏情報中の指番号Finger（図3参照）（図15では「NotePre.Finger」と記載）に対応する出力バッファ用演奏情報ポインタ変数prep[iH][NotePre.Finger]に、変数NotePreの値を格納する（ステップS1239）。このようにして、変数NotePreが示す演奏情報へのポインタが、準備動作を行う演奏情報へのポインタとして、prep[iH][NotePre.Finger]に登録される。

【0151】

10

セット変数setの値が「0」でステップS1238の判定がNOならば、ステップS1239の実行はスキップされ、変数NotePreが示す演奏情報は、準備動作をする演奏情報としては出力バッファ用演奏情報ポインタ変数prep[iH][NotePre.Finger]へは登録されない。

【0152】

ステップS1239の処理の後またはステップS1238の判定がNOならば、CPU201は、押鍵比較段階変数stepの値を+1インクリメントする（ステップS1240）。いままで、step=1、すなわち現在の押鍵の動作タイミングの次の動作タイミングに対応する準備動作に関する処理であったなら、これ以降、step=2となることにより、現在の押鍵の動作タイミングの次の次の動作タイミングに対応する準備動作に関する処理が実行されることになる。

20

【0153】

CPU201は、図15のステップS1240の処理の後、図13のステップS1211の処理に戻る。そして、CPU201は、図13のステップS1211からS1219までの一連の処理によって、次タイミング演奏情報ポインタ変数Next[iH][0]~Next[iH][4]のうちから、出力バッファ用演奏情報ポインタ変数prep[iH][0]~prep[iH][4]において演奏情報へのポインタが定義されていない指に対応するデータであって、かつ上述のように最初に変数NotePreに得られた演奏情報の次に発音開始時刻が早い演奏情報へのポインタとそれに対応する指番号を、それぞれ変数NotePreとfigに取得する。そして、CPU201は、これらの変数NotePreおよびfigについて、図14および図15のフローチャートの処理を再度実行する。この結果、CPU201は、変数NotePreが示す現在の押鍵に対して次の次のタイミングで押鍵されるべき発音開始時刻が最も早い演奏情報へのポインタが、準備動作を行う演奏情報へのポインタとして、prep[iH][NotePre.Finger]に登録される。

30

【0154】

CPU201は、以上の制御処理を繰り返し実行する結果、図13のフローチャートの実行において変数NotePreに演奏情報へのポインタが得られなくなり、図14のステップS1220の判定がNOになると、図9において演奏情報検索ポインタ変数meta[iH]が示す現在のパートiHで最も早く押鍵が行われる演奏情報に対応するステップS908の準備確認の処理を終了する。

40

【0155】

図16は、図14のステップS1226の鍵盤距離の取得の処理の例を示すフローチャートである。

【0156】

CPU201はまず、図14の出力バッファ用演奏情報ポインタ変数prep[iH][ifig]が示す演奏情報と、変数NotePreが示す演奏情報のうち、指番号Finger（図3参照）が大きいほうの演奏情報を変数note fにセットし、指番号Fingerが小さいほうの演奏情報を変数note tにセットする（ステップS1601）

。

50

【 0 1 5 7 】

次に、CPU 201は、演奏情報note fの音高Pitch（図3参照）（図16中では「note f . Pitch」と記載）をピッチ変数pfに格納する。そして、CPU 201は、次式により鍵盤位置を算出し鍵盤位置変数key fに格納する（ステップS 1602）。

【 0 1 5 8 】

$$\text{key f} = \text{OctDist} \times (\text{pf} / 12) + \text{KeyDist}[\text{pf} \% 12] \quad \dots (1)$$

【 0 1 5 9 】

ここで、OctDistは定数であり値「14」を持つ。KeyDist[] = {0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12}であり、KeyDist[0] = 0、KeyDist[1] = 1、KeyDist[2] = 2、KeyDist[3] = 3、KeyDist[4] = 4、KeyDist[5] = 6、KeyDist[6] = 7、KeyDist[7] = 8、KeyDist[8] = 9、KeyDist[9] = 10、KeyDist[10] = 11、KeyDist[11] = 12という値を有する。上記（1）式において、KeyDist[pf % 12]は、ピッチ変数pfの値を12で除算した剰余「pf % 12」（「%」は剰余演算を示す）をインデックスとするKeyDist[pf % 12]値である。なお、定数OctDistは、1オクターブの鍵盤距離を表す定数である。1オクターブ間は12半音であり、OctDistも12とすることも可能であるが、ここでは、便宜上、全ての白鍵 - 白鍵間の鍵盤距離を2とみなして、つまり、ミ - ファ間、シ - ド間も2半音であるとみなして、OctDistとして14という値を用いている。

【 0 1 6 0 】

続いて、CPU 201は、ステップS 1602と同様に、演奏情報note tについても、その音高Pitch（図16中では「note t . Pitch」と記載）をピッチ変数ptに格納する。そして、CPU 201は、次式により鍵盤位置を算出し鍵盤位置変数key tに格納する（ステップS 1603）。

【 0 1 6 1 】

$$\text{key t} = \text{OctDist} \times (\text{pt} / 12) + \text{KeyDist}[\text{pt} \% 12] \quad \dots (2)$$

【 0 1 6 2 】

その後、CPU 201は、ステップS 1602で算出した鍵盤位置変数key fの値からステップS 1603で算出した鍵盤位置変数key tの値を減算し、その減算結果を鍵盤距離変数distに格納する（ステップS 1604）。

【 0 1 6 3 】

次に、CPU 201は、現在のパート検索用変数iH（図9のステップS 902とS 912参照）の値が2であるか否か、すなわち、現在のパートが左手であるか否かを判定する（ステップS 1605）。

【 0 1 6 4 】

CPU 201は、現在のパートが左手でステップS 1605の判定がYESならば、鍵盤距離変数distの値を - 1倍する、すなわち符号を逆にする（ステップS 1606）。右手は低音側（左側）から順に親指、人差し指、・・・、小指の順になっているが、左手は高音側（右側）から順に親指、人差し指、・・・、小指の順になっていることを加味して、後述するように、演算上、効率良く処理するためである。

【 0 1 6 5 】

CPU 201は、ステップS 1606の処理の後、またはステップS 1605の判定がNOの場合に、図16のフローチャートで示される図14のステップS 1226の鍵盤距離の取得の処理を終了する。

【 0 1 6 6 】

鍵盤距離の取得においては、指番号が大きい指の鍵盤位置から指番号が小さい指の鍵盤

10

20

30

40

50

位置を減算して鍵盤距離 $d i s t$ を算出している。従って、右手の場合、例えば人差し指の鍵盤位置が親指の鍵盤位置よりも大きな音高の位置にあれば、鍵盤距離 $d i s t$ は正の値となる。また、右手において、親指で押鍵中に人差し指を親指の上を越えて親指の左側の鍵盤位置を押鍵しようとしているときには、指番号の大きい人差し指の鍵盤位置から親指の鍵盤位置を減算して得られる鍵盤距離 $d i s t$ は 0 より小さい値となる。同様に、人差し指で押鍵中に親指を人差し指の下をくぐって人差し指の右側の鍵盤位置を押鍵しようとしているときには、指番号の大きい人差し指の鍵盤位置から親指の鍵盤位置を減算して得られる鍵盤距離 $d i s t$ はやはり 0 より小さい値となる。

【0167】

左手については、得られた鍵盤距離 $d i s t$ の値を - 1 倍することにより、右手と全く同じ関係にすることができる。すなわち、左手の場合、普通に左手を鍵盤上に置くと、人差し指の鍵盤位置は親指の鍵盤位置よりも小さい音高の位置になるため、その場合、指番号の大きい人差し指の鍵盤位置から親指の鍵盤位置を減算するとマイナスになってしまう。これを右手と同じ関係にするために、- 1 を乗算することによりプラス値にしている。この状態で、左手において、親指で押鍵中に人差し指を親指の上を越えて親指の右側の鍵盤位置を押鍵しようとしているときには、指番号の大きい人差し指の鍵盤位置から親指の鍵盤位置を減算して得られる鍵盤距離 $d i s t$ は 0 より大きな値となり、これを - 1 倍することにより、最終的な鍵盤距離 $d i s t$ はマイナス値となる。同様に、人差し指で押鍵中に親指を人差し指の下をくぐって人差し指の左側の鍵盤位置を押鍵しようとしているときには、指番号の大きい人差し指の鍵盤位置から親指の鍵盤位置を減算して得られる鍵盤距離 $d i s t$ はやはり 0 より大きな値となり、これを - 1 倍することにより、最終的な鍵盤距離 $d i s t$ はマイナス値となる。

【0168】

以上のようにして、親指がからんだ指越えまたは指くぐりが行われたときだけ、図 16 のフローチャートにより算出される鍵盤距離変数 $d i s t$ の値が 0 より小さい値となるようにすることができる。この状態が、図 14 のステップ S 1 2 2 7 で判定される。

【0169】

図 17 は、図 6 のステップ S 6 0 6 のパート同士の状態比較の処理の例を示すフローチャートである。

【0170】

C P U 2 0 1 はまず、パート間の状態補正フラグ変数 $c o n f l i c t$ に、異常がないことを示す値「0」を初期設定する(ステップ S 1 7 0 1)。

【0171】

次に、C P U 2 0 1 は、右手パートに対応する出力バッファ用演奏情報ポインタ変数 $p r e p [0] [0] \sim p r e p [0] [4]$ の指し示す演奏情報のうち、親指を含む最も親指に近い演奏情報を変数 $m e R$ に格納する(ステップ S 1 7 0 2)。具体的には、右手の各指のうち、押鍵または準備で使用している指のうち、もっとも親指に近いものを取得して $m e R$ に格納する。

【0172】

同様に、C P U 2 0 1 は、左手パートに対応する出力バッファ用演奏情報ポインタ変数 $p r e p [1] [0] \sim p r e p [1] [4]$ の指し示す演奏情報のうち、親指を含む最も親指に近い演奏情報を変数 $m e L$ に格納する(ステップ S 1 7 0 3)。具体的には、左手の各指のうち、押鍵または準備で使用している指のうち、もっとも親指に近いものを取得して $m e L$ に格納する。

【0173】

その後、C P U 2 0 1 はまず、変数 $m e L$ が示す演奏情報の音高 $P i t c h$ (図 3 参照)(図 17 中では「 $m e L . P i t c h$ 」と表記)が、変数 $m e R$ が示す演奏情報の音高 $P i t c h$ (図 17 中では「 $m e R . P i t c h$ 」と表記)よりも、高い音高であるか否かを判定する(ステップ S 1 7 0 4)。右手の親指と左手の親指は、それぞれ、他方の手に最も近い位置にある指であり、従って、これは、左手パートの指が右手パートの指の演

10

20

30

40

50

奏領域に入り込んでいるか否かを判定していることになる。

【0174】

ステップS1704の判定がYESならば、CPU201は次に、右手パートの押鍵動作フラグ変数pushKey[0]および左手パートの押鍵動作フラグ変数pushKey[1]ともに値「1」であるか否か、すなわち、両パートとも同時に押鍵を行う状態となっているか否かを判定する(ステップS1705)(図9のステップS903~S905を参照)。

【0175】

ステップS1705の判定がYESならば、CPU201は、パート間の状態補正フラグ変数conflictに異常(左右パートの衝突)が発生したことを示す値「1」をセットする。また、CPU201は、右手パート用の鍵盤奥行き方向状態補正值zOffset[0]または左手パート用の鍵盤奥行き方向状態補正值zOffset[1]のうち、鍵盤奥行き方向で、奥方向(黒鍵側)にあるパートに対応する鍵盤奥行き方向状態補正值に、例えば正の値「1」または「2」等をセットし、手前方向(白鍵側)にあるパートに対応する鍵盤奥行き方向状態補正值に、例えば負の値「-1」または「-2」等をセットする(以上、ステップS1706)。

10

【0176】

ステップS1705の判定がNOならば、CPU201は次に、右手パートの押鍵動作フラグ変数pushKey[0]の値が「1」であるか否かを判定する(ステップS1707)。ステップS1705で両パートとも同時に押鍵を行う状態となっているか否かを判定しているため、ステップS1707での判定は右手パートのみが押鍵を行う状態となっているか否かの判定となる。この場合は、左手パートは、押鍵の準備を行う状態となっている。

20

【0177】

ステップS1707の判定がYESならば、CPU201は、パート間の状態補正フラグ変数conflictに異常(左右パートの衝突)が発生したことを示す値「1」をセットする。また、CPU201は、左手パート用の鍵盤横方向状態補正值xOffset[1]に、ステップS1704で判定された音高の差に応じた量だけ左側方向への移動を指示するための補正值をセットする(以上、ステップS1708)。すなわちこの処理では、右手パートが押鍵を行う状態となっているので、その右手パートに衝突しそうな左手パートの押鍵準備指を、鍵盤横方向の左側方向に逃がすことにより、衝突を防止する調整が行われる。

30

【0178】

ステップS1707の判定がNOならば、CPU201は最後に、左手パートの押鍵動作フラグ変数pushKey[1]の値が「1」であるか否かを判定する(ステップS1709)。ステップS1709での判定は左手パートのみが押鍵を行う状態となっているか否かの判定となる。この場合は、右手パートは、押鍵の準備動作を行う状態となっている。

【0179】

ステップS1709の判定がYESならば、CPU201は、パート間の状態補正フラグ変数conflictに異常(左右パートの衝突)が発生したことを示す値「1」をセットする。また、CPU201は、右手パート用の鍵盤横方向状態補正值xOffset[0]に、ステップS1704で判定された音高の差に応じた量だけ右側方向への移動を指示するための補正值をセットする(以上、ステップS1710)。すなわちこの処理では、左手パートが押鍵を行う状態となっているので、その左手パートに衝突しそうな押鍵の準備動作の状態となっている右手パートの各指を、鍵盤横方向の右側方向に逃がすことにより、衝突を防止する調整が行われる。

40

【0180】

図18および図19は、図6のステップS607の動作データ出力の処理の例を示すフローチャートである。

50

【 0 1 8 1 】

CPU201はまず、パート検索用変数*iH*に値0を設定する(ステップS1801)。前述したように、パート検索用変数*iH*の値は、0なら右手、1なら左手を示す。その後、CPU201は、図19のステップS1826でパート検索用変数*iH*の値を+1しながら、図18のステップS1802で変数*iH*の値が2に達したと判定するまで、変数*iH*が示す右手および左手の各パート毎に、以下の図18のステップS1803から図19のS1825の一連の処理を実行して、動作データ(図4参照)を一括して生成する。

【 0 1 8 2 】

始めに、CPU201は、現在のパート*iH*に対応する押鍵動作フラグ変数pushKey[iH]の値が「1」であるか、またはパート間の状態補正フラグ変数conflictに異常(左右パートの衝突)が発生したことを示す値「1」がセットされているかを判定する(ステップS1803)。

【 0 1 8 3 】

ステップS1803の判定がYESならば、CPU201は、図18のステップS1804から図19のステップS1826までの一連の処理を実行して、動作データ(図4参照)の出力処理を実行する。すなわち、現在のパート*iH*に対応する押鍵動作フラグ変数pushKey[iH]の値が「1」である、つまり現在のパート*iH*が押鍵を行う状態になっているならば、無条件に動作データの出力処理が実行される。また、現在のパート*iH*が押鍵を行う状態になっておらず押鍵の準備動作を行う状態になっている場合であっても、conflictに異常が発生したことを示す値「1」がセットされている、つまり左右パートの衝突が発生しそうな状態になっているならば、動作データの出力処理が実行される。これは、他方のパートが押鍵を行う状態になっていてその状態が変化する毎に、現在のパート*iH*のほうも衝突しないように押鍵の準備動作を行う各指の位置を移動させ、その移動後の状態で押鍵の準備動作を行う現在のパート*iH*のほうもガイド表示する必要があるためである。ただし、図11のステップS1109からステップS1110に移行する処理の説明で前述したように、現在のパート*iH*が押鍵を行う状態になっておらず押鍵の準備動作を行う状態になっていて押鍵動作フラグ変数pushKey[iH]の値が「1」でなければ、現在のパート*iH*の演奏情報検索ポインタ変数meTail[iH]の内容は更新されない。つまり、現在のパート*iH*が押鍵を行う状態になって押鍵動作フラグ変数pushKey[iH]の値が「1」になるまでは、以下に説明する動作データの出力処理では、現在のパート*iH*の各指の演奏位置は、衝突回避のための補正をした上で同じ押鍵の準備動作の状態を表示し続けることになる。

【 0 1 8 4 】

この動作データの出力処理では、現在のパート*iH*について、図6のステップS606のパート同士の状態比較の処理でセットされた、鍵盤奥行き方向状態補正值zOffset[iH](図17のステップS1706の説明を参照)、または鍵盤横方向状態補正值xOffset[iH](図17のステップS1708または1710の説明を参照)に基づいて修正が行われる。すなわち、出力される動作データの奥行き位置frame[i].Zposの修正(図19のステップS1823)と、または出力される動作データの各指の演奏位置(鍵盤番号)frame[i].position[ifig]の修正(図19のステップS1814)が行われる。

【 0 1 8 5 】

なお、ステップS1803の判定がNO、すなわち、現在のパート*iH*の各指が押鍵の準備動作の状態になっていて押鍵動作フラグ変数pushKey[iH]の値が「1」ではなく、かつ左右パートの衝突が発生しておらずconflict=1でなければ、図18のフローチャートの処理を終了して、パート*iH*については、動作データの出力は行われない(図18のステップS1803:NO)。この場合には、前述した図6のステップS605のパート毎の状態確認の処理における図9のステップS907の押鍵検索の処理において、押鍵を行う指ifig(図11のステップS1105)の検索および押鍵対象演奏情報ポインタ変数Hand[iH][ifig]への設定が行われた後に、図11の

ステップS 1 1 0 9の判定が行われる。ステップS 1 1 0 9では、現在のパートi Hの押鍵動作フラグ変数pushKey[i H]が値「1」か否かが判定される。そして、ステップS 1 1 0 9の判定がNOならば、図11のステップS 1 1 1 0が実行されない結果、現在のパートi Hの処理対象の演奏情報検索ポインタ変数meTail[i H]の内容は更新されずに、やがて図6のステップS 6 0 3に戻る。この結果、図6のステップS 6 0 3での次の演奏情報の検索において、再び同じ演奏情報が検索されることになり、これがやがて他のパートの演奏情報よりも発音開始時刻が早くなって押鍵の状態を示すようになったときに、そのタイミングで図18のステップS 1 8 0 3の判定がYESとなって動作データが出力される。

【0186】

10

図18のステップS 1 8 0 3の判定がYESになると、始めに、CPU 2 0 1は、ステップS 1 8 0 4からS 1 8 1 0の一連の処理を実行する。ここでは、変数の初期設定と、今回出力する動作データframe[i]のための今回の動作開始時間start、動作終了時間term、トラック番号Handの算出と設定等が行われる。

【0187】

まず、CPU 2 0 1は、動作データ(図4)の各フレームframe[i]を指示するためのフレームカウント用変数iに、現時点におけるフレームの総数FrameCountの値を格納する(ステップS 1 8 0 4)。このフレームの総数FrameCountは、図6のステップS 6 0 1で値「0」に初期化され、後述する図19のステップS 1 8 2 4で+1ずつインクリメントされる。これにより、動作データの数が増やされてゆく。

20

【0188】

次に、CPU 2 0 1は、ステップS 1 8 0 5からS 1 8 1 0で、今回の動作タイミングにおける動作データframe[i]を生成するときの、トラック番号Hand、動作開始時間startと動作終了時間term(図4参照)を決定する。今回の検索ポインタ変数mePtrに対応する演奏情報、およびそれと同時に押鍵または準備動作される出力バッファ用演奏情報ポインタ変数prep[i H][0]~prep[i H][4]が示す各演奏情報に対して、演奏ガイドを行うためには、次のような制御が必要となる。すなわち、その演奏ガイドは、mePtrが指す演奏情報の発音開始時刻mePtr.Timeより前からガイド表示を開始し、発音開始時刻mePtr.Timeでガイドを終了(実際に押鍵)する必要がある。

30

【0189】

そこで、CPU 2 0 1はまず、現在のパートi Hに対応する動作開始時間start[i H](同名の変数に格納されている)が、今回の演奏情報の発音開始時刻mePtr.Timeよりも大きいか否かを判定する(ステップS 1 8 0 5)。動作開始時間start[i H]は、一番最初は図6のステップS 6 0 1で値「0」に初期設定されている。また、動作途中では、一つ前の検索ポインタ変数mePtrの値における動作データ出力の処理の最後の図19のステップS 1 8 2 5で、次の検索ポインタ変数mePtrの値のための動作開始時間start[i H]が算出されている(後述する図21参照)。

【0190】

40

上述のようにして算出されている今回の動作開始時間start[i H]が今回の演奏情報の発音開始時刻mePtr.Timeよりも後(大きい値)になっているときは、演奏ガイドは今回の演奏情報の発音開始時刻mePtr.Timeよりも前になければならないので、動作開始時間start[i H]の修正が必要となる。

【0191】

すなわち、CPU 2 0 1は、ステップS 1 8 0 5の判定がYESならばさらに、現在のパートi Hに対応する前回の演奏情報の発音開始時刻laston[i H](同名の変数に格納されている)が、今回の演奏情報の発音開始時刻mePtr.Timeよりも小さいか否かを判定する(ステップS 1 8 0 6)。

【0192】

50

ステップS1806の判定がYESならば、CPU201は、前回発音開始時刻 $laston[iH]$ から、前回発音開始時刻 $laston[iH]$ から今回の演奏情報の発音開始時刻 $mePtr.Time$ までの経過時間に所定の比率 $RATIO$ (例えば $80\% = 0.8$)を乗算して得られる時間だけ経過した時点、動作開始時間 $start[iH]$ として設定する(ステップS1808)。すなわち、次式の演算が実行される。

【0193】

$$start[iH] = laston[iH] + (mePtr.Time - laston[iH]) \times RATIO \cdots (3)$$

【0194】

これにより、前回の音の発音開始時刻から、今回の音の発音開始時刻までの時間の、例えば 80% ($RATIO$ で表される比率)だけ経過した時点、動作開始時間 $start[iH]$ として設定することとなる。

【0195】

一方、前回発音開始時刻 $laston[iH]$ が $mePtr.Time$ よりも小さくなくステップS1806の判定がNOの場合には、仕方がないので、前回発音開始時刻 $laston[iH]$ を今回の動作開始時間 $start[iH]$ にセットする(ステップS1807)。

【0196】

算出されている動作開始時間 $start[iH]$ が今回の演奏情報の発音開始時刻 $mePtr.Time$ よりも大きくはなくステップS1805の判定がNOの場合には、CPU201は、その算出されている動作開始時間 $start[iH]$ をそのまま使用する。

【0197】

以上のようにして今回の動作開始時間 $start[iH]$ が決定した後、CPU201は、 $mePtr$ の次の値による次回の処理のために、今回の演奏情報の発音開始時刻 $mePtr.Time$ を前回発音開始時刻 $laston[iH]$ としてセットする(ステップS1809)。

【0198】

その後、CPU201は、フレームカウント用変数 i の値に対応する動作データ $frame[i]$ のトラック番号 $Hand$ (図18では「 $frame[i].Hand$ 」と記載)に、現在のパート検索用変数 iH の値(図18のステップS1802, 図19のS1826参照)を格納する。また、CPU201は、動作データ $frame[i]$ の動作開始時間 $start$ (図18では「 $frame[i].start$ 」と記載)に、上述のように算出した今回の動作開始時間 $start[iH]$ を格納する。さらに、CPU201は、動作データ $frame[i]$ の動作終了時間 $term$ (図18では「 $frame[i].term$ 」と記載)に、今回の演奏情報の発音開始時刻 $mePtr.Time$ を格納する(以上、ステップS1810)。

【0199】

ステップS1810の後、CPU201は、図19のステップS1811からS1825までの一連の処理を実行する。ここでは、今回の動作データ $frame[i]$ のための5本の各指の演奏位置「 $position[0] \sim position[4]$ 」(指を置くべき鍵盤番号)、各指の状態「 $status[0] \sim status[4]$ 」(それぞれ、値「-1」が準備動作、値「1」が押鍵)、奥行き方向の位置「 $Zpos$ 」などの情報の生成と設定が行われる(図4参照)。

【0200】

CPU201は、ステップS1811で指番号指定変数 $ifig$ に初期値「0」(親指)を設定した後、ステップS1821で $ifig$ の値を+1ずつインクリメントしながら、ステップS1812で片手の指の最大数「5」に達したと判定するまで、以下のステップS1813からS1820の一連の処理を実行する。

【0201】

10

20

30

40

50

まず、CPU201は、現在のパートiHの現在の指番号ifigに対応する出力バッファ用演奏情報ポインタ変数prep[iH][ifig]の値が未定義値NULLでないか否かを判定する(ステップS1813)。

【0202】

ステップS1813の判定がYESならば、現在処理中の指ifigについて、出力用の演奏情報prep[iH][ifig]が決定されているので、CPU201は、prep[iH][ifig]の音高Pitch(図3参照)(図19中では「prep[iH][ifig].Pitch」と記載)に、現在のパートiHの鍵盤横方向状態補正值xOffset[iH]を加算して得られる値を、今回の動作データframe[i]の指番号ifigに対応する演奏位置position[ifig](図19中では「frame[i].position[ifig]」と記載)に格納する(ステップS1814)。すなわち、現在処理中の指ifigに関しての指位置(position)のデータとして、prep[iH][ifig]で決定されている演奏情報の音高に対して、前述した右手パートと左手パートの衝突を考慮した鍵盤横方向の補正值(図17のステップS1708およびS1710の説明を参照)を考慮して補正した値を格納する。

10

【0203】

次に、CPU201は、現在のパートiHの現在の指番号ifigに対応する出力バッファ用演奏情報ポインタ変数prep[iH][ifig]が示す演奏情報の発音開始時刻Time(図19中では「prep[iH][ifig].Time」と記載)が、今回の演奏情報の発音開始時刻mePtr.Timeと等しいか否か、すなわちprep[iH][ifig]とmePtrが指す演奏情報の各発音開始時刻Timeが等しいか否かを判定する(ステップS1815)。これは、指番号ifigの押鍵がmePtr.Timeの押鍵と同時であるか否かを判定することとなる。

20

【0204】

ステップS1815の判定がYESならば、CPU201は、今回の動作データframe[i]の指番号ifigに対応する状態status[ifig](図19中では「frame[i].status[ifig]」と記載)に、押鍵を示す値「1」を格納する(ステップS1816)。

【0205】

その後、CPU201は、ステップS1821の処理に移行し、次の指番号の処理に進む。

30

【0206】

ステップS1815で、現在のパートiHの現在の指番号ifigに対応する出力バッファ用演奏情報ポインタ変数prep[iH][ifig]の発音開始時刻Timeが、今回の演奏情報mePtrの発音開始時刻Timeと等しくなく、ステップS1815の判定がNOならば、CPU201は、prep[iH][ifig]>mePtr.Timeであるか否かを判定する(ステップS1817)。

【0207】

ステップS1817の判定がYESならば、現在のパートiHの現在の指番号ifigに対応するprep[iH][ifig]が示す演奏情報は、準備動作を行うべきものである。このため、CPU201は、今回の動作データの指番号ifigに対応する状態frame[i].status[ifig]に、準備動作を示す値「-1」を格納する(ステップS1818)。

40

【0208】

その後、CPU201は、ステップS1821の処理に移行し、次の指番号の処理に進む。

【0209】

ステップS1817の判定がNOならば、現在の指番号ifigに対応する出力バッファ用演奏情報ポインタ変数prep[iH][ifig]には演奏情報へのポインタが格納されていないことを意味する。この場合には、CPU201は、今回の動作データの指

50

番号 $i f i g$ に対応する状態 $f r a m e [i] . s t a t u s [i f i g]$ に、データ無しを示す値「0」を格納する（ステップ $S 1 8 1 9$ ）。

【0210】

その後、CPU 201は、ステップ $S 1 8 2 1$ の処理に移行し、次の指番号の処理に進む。

【0211】

ステップ $S 1 8 1 3$ に戻って、現在の指番号 $i f i g$ に対応する出力バッファ用演奏情報ポインタ変数 $p r e p [i H] [i f i g]$ の値が未定義値 $N U L L$ であって、ステップ $S 1 8 1 3$ の判定が $N O$ ならば、CPU 201は、今回の動作データの指番号 $i f i g$ に対応する演奏位置 $f r a m e [i] . p o s i t i o n [i f i g]$ に、データ無しを示す値「-1」を格納し、また、状態 $f r a m e [i] . s t a t u s [i f i g]$ に、データ無しを示す値「0」を格納する（ステップ $S 1 8 2 0$ ）。

10

【0212】

その後、CPU 201は、ステップ $S 1 8 2 1$ の処理に移行し、次の指番号の処理に進む。

【0213】

以上のステップ $S 1 8 1 3$ から $S 1 8 2 0$ までの一連の処理が、5本の全ての指番号 $i f i g$ について実行されると、ステップ $S 1 8 1 2$ の判定が $N O$ となる。

【0214】

この結果、CPU 201は、奥行き位置計算の処理を実行し、その結果得られる奥行き方向の位置を、変数 $z p o s$ に得る（ステップ $S 1 8 2 2$ ）。この奥行き位置計算の処理の詳細については、図20を用いて後述する。

20

【0215】

CPU 201は、変数 $z p o s$ に得た奥行き方向の位置に、現在のパート $i H$ の鍵盤奥行き方向状態補正值 $z O f f s e t [i H]$ を加算して得られる値を、現在の動作データに $f r a m e [i] . Z p o s$ （図4参照）として格納する（ステップ $S 1 8 2 3$ ）。ここでは、前述した右手パートと左手パートの衝突を考慮しないで算出された奥行き方向の位置 $z p o s$ が、衝突を考慮した鍵盤奥行き方向の補正值（図17のステップ $S 1 7 0 6$ の説明を参照）で補正され、その補正結果が $f r a m e [i] . Z p o s$ に格納される。

【0216】

30

CPU 201は、動作データ $f r a m e [i]$ のフレームの総数 $F r a m e C o u n t$ を+1インクリメントする（ステップ $S 1 8 2 4$ ）。

【0217】

最後に、CPU 201は、次回開始時刻の計算の処理を実行し、次の検索ポインタ変数 $m e P t r$ の値のための動作開始時間 $s t a r t [i H]$ を算出する（ステップ $S 1 8 2 5$ ）。この次回開始時刻の算出処理の詳細については、図21を用いて後述する。

【0218】

その後、CPU 201は、ステップ $S 1 8 2 6$ でパート検索用変数 $i H$ の値を+1ずつインクリメントしながら、図18のステップ $S 1 8 0 2$ で変数 $i H$ の値が2に達したと判定するまで、変数 $i H$ が示す右手および左手の各パート毎に、以下の図18のステップ $S 1 8 0 3$ から図19の $S 1 8 2 5$ の一連の処理を実行し、図18のステップ $S 1 8 0 2$ で変数 $i H$ の値が2に達したと判定されると（ステップ $S 1 8 0 2 : N O$ ）、図18および図19のフローチャートで示される図6のステップ $S 6 0 7$ の動作データ出力の処理を終了する。

40

【0219】

図20は、図19のステップ $S 1 8 2 2$ の奥行き位置計算の処理の例を示すフローチャートである。この処理では、現在の各指の押鍵や準備動作の状態から、手を鍵盤の奥行き方向に対して、どの位置に置くのが相応しいかを判定する。ここでは、鍵盤奥行き方向の手の位置として、 $z p o s = 1 \sim 4$ の4段階で手の位置を定義し、このいずれの奥行き位置が相応しいかを判断している。

50

【 0 2 2 0 】

まず、CPU 201は、奥行き位置変数 $zpos$ に暫定的に値「1」を格納する（ステップ S 2001）。この値は、鍵盤上の手前側に手を置き、全部の指が白鍵を押鍵する位置にあることを指示する。

【 0 2 2 1 】

次に、CPU 201は、親指に対応する出力バッファ用演奏情報ポインタ変数 $prep[iH][0]$ の値が未定義値 $NULL$ でないか否かを判定する（ステップ S 2002）。

【 0 2 2 2 】

ステップ S 2002の判定が YES ならば、親指に対応する演奏情報が存在することになるので、CPU 201は、 $prep[iH][0]$ が示す演奏情報の音高データ $prep[iH][0].Pitch$ を、ピッチ変数 p に格納する（ステップ S 2003）。

10

【 0 2 2 3 】

さらに、CPU 201は、ピッチ変数 p の値に対応する黒鍵判定定数 $BlackKey[p]$ の値が「1」であるか否か、すなわち親指に対応する演奏情報が示す音高 $Pitch$ が黒鍵であるか否かを判定する（ステップ S 2004）。いま、黒鍵判定定数 $BlackKey[] = \{0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0\}$ である。値「0」は白鍵を示し、値「1」は黒鍵を示す。この配列の並びは、実際の鍵盤楽器における白鍵と黒鍵の一般的な並びに対応している。この配列値を、ピッチ変数 p をキー値として調べることにより、演奏情報が示す音高 $Pitch$ が黒鍵であるか否かを判定することができる。実際には、音高データ $prep[iH][0].Pitch$ の値を、12で除した余りの値を用いる等して、オクターブ内での音高を算出して判定を行う。

20

【 0 2 2 4 】

ステップ S 2004の判定が YES ならば、親指に対応する演奏情報が黒鍵であることになるので、CPU 201は、奥行き位置変数 $zpos$ に値「4」を格納する（ステップ S 2005）。この値は、片手の5本の指全てを鍵盤上の黒鍵が押せる程度奥側に寄せて置くことを指示する。すなわち、親指が黒鍵側（奥側）に置かれるということは、5本の全ての指が自然に黒鍵が押せる程度の奥側に置かれることを意味する。

【 0 2 2 5 】

ステップ S 2005の処理の後、CPU 201は、図20のフローチャートで示される図19のステップ S 1822の奥行き位置計算の処理を終了する。

30

【 0 2 2 6 】

一方、ステップ S 2002に戻って、親指に対応する出力バッファ用演奏情報ポインタ変数 $prep[iH][0]$ の値が未定義値 $NULL$ になっておりステップ S 2002の判定が NO ならば、CPU 201は、小指に対応する出力バッファ用演奏情報ポインタ変数 $prep[iH][4]$ の値が未定義値 $NULL$ でないか否かを判定する（ステップ S 2006）。

【 0 2 2 7 】

ステップ S 2006の判定が YES ならば、小指に対応する演奏情報が存在することになるので、CPU 201は、 $prep[iH][4]$ が示す演奏情報の音高データ $prep[iH][4].Pitch$ を、ピッチ変数 p に格納する（ステップ S 2007）。

40

【 0 2 2 8 】

さらに、CPU 201は、ピッチ変数 p の値に対応する黒鍵判定定数 $BlackKey[p]$ の値が「1」であるか否か、すなわち小指に対応する演奏情報が示す音高 $Pitch$ が黒鍵であるか否かを判定する（ステップ S 2008）。

【 0 2 2 9 】

ステップ S 2008の判定が YES ならば、小指に対応する演奏情報が黒鍵であることになるので、CPU 201は、奥行き位置変数 $zpos$ に値「3」を格納する（ステップ S 2009）。この値は、人差し指、中指、薬指、小指の4本の指を鍵盤上の黒鍵が押せる程度奥側に寄せて置くことを指示する。すなわち、小指が黒鍵側（奥側）に置かれると

50

いうことは、上記 4 本の指が自然に黒鍵が押せる程度の奥側に置かれることを意味する。

【0230】

ステップ S 2009 の処理の後、CPU 201 は、図 20 のフローチャートで示される図 19 のステップ S 1822 の奥行き位置計算の処理を終了する。

【0231】

親指に加えて、小指に対応する出力バッファ用演奏情報ポインタ変数 $prep[iH][4]$ の値が未定義値 $NULL$ になっておりステップ S 2006 の判定が NO ならば、CPU 201 は、ステップ S 2010 で変数 i に値「1」を設定した後、ステップ S 2016 で変数 i の値を +1 ずつインクリメントしながら、ステップ S 2011 で変数 i の値が 4 に達したと判定するまで、以下のステップ S 2012 から S 2015 までの一連の処理

10

【0232】

まず CPU 201 は、変数 i の値が示す指（「1」なら人差し指、「2」なら中指、「3」なら薬指）に対応する出力バッファ用演奏情報ポインタ変数 $prep[iH][i]$ の値が未定義値 $NULL$ でないか否かを判定する（ステップ S 2012）。

【0233】

ステップ S 2012 の判定が YES ならば、CPU 201 は、 $prep[iH][i]$ が示す演奏情報の音高データ $prep[iH][i].Pitch$ を、ピッチ変数 p に格納する（ステップ S 2013）。

【0234】

さらに、CPU 201 は、ピッチ変数 p の値に対応する黒鍵判定定数 $BlackKey[p]$ の値が「1」であるか否か、すなわち変数 i の値に対応する指の演奏情報が示す音高 $Pitch$ が黒鍵であるか否かを判定する（ステップ S 2014）。

20

【0235】

ステップ S 2014 の判定が YES ならば、CPU 201 は、奥行き位置変数 $zpos$ に値「2」を格納する（ステップ S 2015）。この値は、人差し指、中指、薬指の 3 本の指を鍵盤上の黒鍵が押せる程度奥側に寄せて置くことを指示する。すなわち、上記 3 本の何れかの指が黒鍵側（奥側）に置かれるということは、上記 3 本の指が自然に黒鍵が押せる程度の奥側に置かれることを意味する。

【0236】

ステップ S 2015 の処理の後、CPU 201 は、図 20 のフローチャートで示される図 19 のステップ S 1822 の奥行き位置計算の処理を終了する。

30

【0237】

変数 i の値が 4 になりステップ S 2011 の判定が NO になった場合も、CPU 201 は、図 20 のフローチャートで示される図 19 のステップ S 1822 の奥行き位置計算の処理を終了するが、この場合は、いずれの指も黒鍵ではない場合に相当するので、奥行き位置変数 $zpos$ の値は、ステップ S 2001 で初期設定された値「1」となっている。

【0238】

以上の奥行き位置計算の処理では、各指が押鍵をするのか準備動作をするのかにかかわらず、手を置く位置が計算される。

40

【0239】

図 21 は、図 19 のステップ S 1825 の次回開始時刻の計算の処理の例を示すフローチャートである。

【0240】

CPU 201 は、以下のステップ S 2101 から S 2112 の一連の処理によって、演奏情報検索用変数 iH の次の値に対応する次回の図 18 および図 19 の動作データ出力の処理のために、次回の動作開始時間 $start[iH]$ を算出する。ここでは、現在発音されている演奏情報のうち、最も早く離鍵される音の演奏情報を基に、次回の動作開始時間 $start[iH]$ が設定される。

【0241】

50

まず、CPU201は、今回の演奏情報の発音開始時刻`mePtr . Time`を暫定的に次の動作開始時間`start[iH]`に設定する(ステップS2101)。

【0242】

次に、今回の演奏情報の発音開始時刻`mePtr . Time`に今回の演奏情報の発音継続時間`mePtr . Gate`(図3参照)を加算した値が、その演奏情報のノートオフ時刻`NoteOff`(同名の変数に格納される)として設定される(ステップS2102)。

【0243】

次に、CPU201は、ステップS2103で指番号指定変数`ifig`に初期値「0」(親指)を設定した後、ステップS2110で`ifig`の値を+1ずつインクリメントしながら、ステップS2104で片手の指の最大数「5」に達したと判定するまで、以下のステップS2105からS2109までの一連の処理を実行する。

【0244】

まず、CPU201は、現在のパート`iH`の現在の指番号`ifig`に対応する出力バッファ用演奏情報ポインタ変数`prep[iH][ifig]`の値が未定義値`NULL`でないか否かを判定する(ステップS2105)。

【0245】

ステップS2105の判定が`YES`ならば、CPU201は、現在のパート`iH`の現在の指番号`ifig`に対応する出力バッファ用演奏情報ポインタ変数が示す演奏情報の、発音開始時刻`prep[iH][ifig].Time`が、今回の演奏情報の発音開始時刻`mePtr . Time`以下か否か、すなわち指番号`ifig`の押鍵が`mePtr . Time`の押鍵と同時若しくは、`prep[iH][ifig]`に入っている音が、今回の演奏情報`mePtr`よりも前に押鍵されて、まだ押鍵継続中である音のいずれかであるか否かを判定する(ステップS2106)。

【0246】

ステップS2106の判定が`YES`ならば、CPU201は、現在のパート`iH`の現在の指番号`ifig`に対応する出力バッファ用演奏情報ポインタ変数が示す演奏情報の発音開始時刻`prep[iH][ifig].Time`に、同じ演奏情報の発音継続時間`prep[iH][ifig].Gate`に比率`RATIO`(例えば80% = 0.8)を乗算して得た値を加算し、その加算結果を、現在調査中の演奏情報`prep[iH][ifig]`の動作終了時間として、`term`に設定する(ステップS2107)。

【0247】

その後、CPU201は、次の動作開始時間`start[iH]`が、現在調査中の演奏情報`prep[iH][ifig]`の動作終了時間`term`よりも小さく、かつ、現在調査中の演奏情報`prep[iH][ifig]`の動作終了時間`term`がノートオフ時刻`NoteOff`よりも小さいか否かを判定する(ステップS2108)。

【0248】

ステップS2108の判定が`YES`ならば、ステップS2107で算出した動作終了時間`term`が、現在発音されている演奏情報の中で、ここまでに調べた音のうち暫定的に最も早く離鍵するものである。この場合、CPU201は、その動作終了時間`term`を新たなノートオフ時刻`NoteOff`として設定する(ステップS2109)。

【0249】

ステップS2108の判定が`NO`ならば、CPU201は、現在のノートオフ時刻`NoteOff`を維持する。

【0250】

ステップS2109の処理の後、またはステップS2108の処理の判定が`NO`である場合、または、現在のパート`iH`の現在の指番号`ifig`に対応する出力バッファ用演奏情報ポインタ変数`prep[iH][ifig]`が示す演奏情報が、同時若しくはそれ以前に押鍵されるものではなくステップS2106の判定が`NO`の場合、あるいは`prep[iH][ifig]`が未定義値`NULL`でステップS2105の判定が`NO`ならば、C

10

20

30

40

50

P U 2 0 1 は、ステップ S 2 1 1 0 の処理に移行し、i f i g の値を + 1 インクリメントして、次の指番号の処理に進む。

【 0 2 5 1 】

全ての指番号 i f i g に対する処理が終了しステップ S 2 1 0 4 の判定が N O になると、C P U 2 0 1 は、次の動作開始時間 s t a r t がノートオフ時刻 N o t e O f f よりも前（小さい）か否かを判定する（ステップ S 2 1 1 1 ）。

【 0 2 5 2 】

ステップ S 2 1 1 1 の判定が Y E S ならば、C P U 2 0 1 は、ノートオフ時刻 N o t e O f f を次の動作開始時間 s t a r t [i H] に設定し、N O ならば、現在の動作開始時間 s t a r t [i H] の値を維持して、図 2 1 のフローチャートで示される図 1 9 のステップ S 1 8 2 5 の次回開始時刻の計算の処理を終了する。

【 0 2 5 3 】

以上説明した図 5 から図 2 1 のフローチャートにより、R A M 2 0 3 上に生成される、図 4 で説明した動作データ f r a m e [0] ~ f r a m e [F r a m e C o u n t - 1] は、動作タイミング毎に、どの指を押鍵するかという情報のみならず、どの指を次の押鍵のために準備するかという情報も、各指の状態 s t a t u s [0] ~ s t a t u s [4] として保有することが特徴である。このような動作データを用いることにより、C P U 2 0 1 は、表示部 2 0 5 のディスプレイ上で、次のような演奏動作ガイドを実現することができる。

【 0 2 5 4 】

例えば、親指の状態を示す s t a t u s [0] に押鍵を示す値「1」がセットされ、親指以外の指の状態を示す s t a t u s [1] ~ s t a t u s [4] のうちのいずれかに準備動作を示す値「-1」がセットされている場合、C P U 2 0 1 は、親指の位置 p o s i t i o n [0] で表される鍵を親指で押鍵するとともに、親指以外の指の状態を示す s t a t u s [1] ~ s t a t u s [4] のうち準備動作を示す値「-1」がセットされている指について、その指に対応する p o s i t i o n [1] ~ p o s i t i o n [4] のいずれかで表される鍵の位置で、その指を準備（配置）させるように動作する。この場合において、親指の位置 p o s i t i o n [0] と、準備すべき他の指の位置 p o s i t i o n [1] ~ p o s i t i o n [4] の鍵の位置との関係によっては、通常の指の左右の位置関係と逆転した位置関係になっている場合がある。このような場合、C P U 2 0 1 は、「親指以外の指を親指の上を越えて次の押鍵位置まで移動させておきなさい」という準備動作をガイド表示させることが可能である。

【 0 2 5 5 】

また例えば、親指以外の指の状態を示す s t a t u s [1] ~ s t a t u s [4] のうちのいずれかに押鍵を示す値「1」がセットされ、親指の状態を示す s t a t u s [0] に準備動作を示す値「-1」がセットされている場合も同様に、押鍵を示す指の鍵の位置と、親指が準備動作を行うべき鍵の位置との位置関係によっては、C P U 2 0 1 は、「親指を親指以外の指の下をくぐらせて次の押鍵位置まで移動させておきなさい」という準備動作をガイド表示させることが可能である。

【 0 2 5 6 】

これらの演奏ガイド表示は、演奏動作上のテクニックに関する動作情報であるということが出来る。このようにして、本実施形態の制御処理によって、演奏情報に付加されている運指情報（図 3 の「指番号」）から演奏動作上のテクニックに関する動作情報を自動的に抽出することが可能となり、それにより運指とともに演奏動作上のテクニックも演奏ガイドすることが可能となる。

【 0 2 5 7 】

図 2 2 は、演奏情報の記憶例を示す図、図 2 3 は、図 2 2 の演奏情報に対応して、準備動作を考慮しないときの動作データの出力例を示す図、図 2 4 は、図 2 2 の演奏情報に対応して、図 5 から図 2 1 のフローチャートで示される制御処理により、準備動作を考慮したときの動作データの出力例を示す図、図 2 5 は、演奏ガイドの表示例を示す図である。

10

20

30

40

50

ただし、この例の場合は、片手のみのデータであるので、上記の実施例中で、右手と左手の衝突に関する調整（例えば図17のパート同士の状態比較の処理）は必要としていない場合の例である。図23および図24において、濃いグレー色のデータ部分は押鍵動作が指定されている部分である。図24において、薄いグレー色のデータ部分は押鍵の準備動作が指定されている部分である。

【0258】

図22の演奏情報は、図3で説明したデータ構成例に従って、1行に1組の演奏情報、すなわち1行につき1音のデータが並んでいる。

【0259】

この演奏データに対して、準備動作を考慮しない場合には、図23に例示されるように、各指の演奏位置 `position[0] ~ position[4]` 毎に、音高が設定されると共に（濃い灰色部分）（値「-1」（色無しの部分）は音高設定無しを示す）、該当する状態 `status[0] ~ status[4]` に、押鍵を行う場合は値「1」（濃い灰色部分）が、行わない場合は値「0」（色無しの部分）が設定されるだけである。従って、このような動作データに基づいて、演奏ガイドの表示が行われる場合には、図23の例えば*の行位置では、音高67の鍵が `position[0]` の親指で押鍵された後に、*の次の行位置で、音高「67」の鍵の左隣りの音高「66」の鍵が `position[1]` の人差し指で押鍵される。これは、親指の押鍵に対して人差し指によって指越えを行って次の押鍵を行う演奏テクニックを必要とする例である。しかし、図22に例示される準備動作を考慮しない動作データでは、現在の押鍵に対して次にどのような押鍵がなされるかは知ることができないため、単純な演奏ガイド表示しか行うことができない。

【0260】

これに対して、図22に例示される演奏データに対して、本実施形態の制御処理により準備動作を考慮した場合には、図24に例示されるように、各指の演奏位置 `position[0] ~ position[4]` 毎に、押鍵を行う指の音高（濃い灰色部分）と準備動作を行う指の音高（薄い灰色部分）が設定されると共に（値「-1」（色無しの部分）は音高設定無しを示す）、該当する状態 `status[0] ~ status[4]` に、押鍵を行う場合は値「1」（濃い灰色部分）が、準備動作を行う場合は値「-1」（薄い灰色部分）が、どちらも行わない場合は値「0」（色無しの部分）を設定することが可能となる。従って、このような動作データに基づいて、演奏ガイドの表示が行われる場合には、図24の例えば*の位置に示されるように、音高「67」の鍵が `position[0]` の親指で押鍵されるときに（濃い灰色部分）、音高「67」の鍵の左隣りの音高「66」の鍵を `position[1]` の人差し指で押鍵する準備動作に入るべきことを、`status[1]` に設定される値「-1」と、`position[1]` に設定される音高「66」とで指定することができる（薄い灰色部分）。そして、親指が関係していることが認識されることによって、演奏ガイドの表示においては、現在の親指の押鍵に対して人差し指が指越えを行って次の押鍵を行う準備に入ることを表示させることが可能となる。これにより、ユーザは、押鍵時の演奏テクニックを習得することが可能となる。この場合、演奏情報としては、図22に例示されるように、演奏テクニックのための特別な情報を持たせる必要はなく、指番号 `Finger` の情報から、準備動作の情報を自動的に生成することが可能となる。

【0261】

図25は、演奏ガイドの表示例を示す図であり、右手の演奏ガイド表示を示している。鍵盤上の5個の丸い印は、右手の各指のポジションを示しており、左端の少し大きい丸印は親指を示し、右に向かって人差し指、中指、薬指、小指にそれぞれ対応する小さい丸印が示されている。左手の場合は、方向が逆になる。また、押鍵状態のときは、鍵盤が濃い色になる。

【0262】

図25(a)の例では、親指が押鍵状態で、その他の指は準備状態であることが示されている。このとき、図18のフローチャートで示した奥行き位置計算処理によって、図2

10

20

30

40

50

5 (a) の演奏ガイド例では、`z p o s` に値「1」が格納されることにより、全ての指を白鍵上に置く指示が表示される。

【0263】

図25 (b) の例も、親指が押鍵状態で、その他の指は準備状態であることが示されているが、図20のフローチャートで示した奥行き位置計算処理によって、図25 (b) の演奏ガイド例では、`z p o s` に値「2」が格納されることにより、人差し指、中指、薬指の3本を黒鍵上に置き、親指と小指を白鍵上に置く指示が表示される。そして、人差し指は親指よりも半音低い黒鍵上に置いて準備する指示が表示されるため、演奏者は、自然に親指越えをして人差し指をF # の黒鍵上に置いて準備する状態をとることができる。

【0264】

なお、指越えや指くぐりが発生したときに、それらを明示的に示すような表示がなされてもよい。例えば、画面上に矢印や特殊なマークを表示するようにしても良い。あるいは、対象となる指（例えば、親指や人差し指）を点滅させたり、別の色で表示する等しても良い。

【0265】

図26は、左手の押鍵ガイドがその演奏進行に伴って右手の押鍵の準備ガイドに衝突しうる楽譜例を示す図である。図26において、ト音記号が付与された上段のパートは右手パートを示し、ヘ音記号が付与された下段のパートは左手パートを示している。各音符の上に付与されている1から5の範囲の数字は、押鍵時の指番号を示しており、1は親指、2は人差し指、3は中指、4は薬指、5は小指で押鍵がされる音符であることを示している。図27は、図26の楽譜例に対応し、衝突回避制御を行わないときの衝突が発生する場合の右手パートに関する動作データの出力例（その1）を示す図である。図28は、図26の楽譜例に対応し、衝突回避制御を行うときの衝突が発生する場合の右手パートに関する動作データの出力例（その1）を示す図である。図27および図28において、図23および図24の場合と同様に、濃い灰色のデータ部分は押鍵動作が指定される部分、薄い灰色のデータ部分は押鍵の準備動作が指定される部分である。また、図27および図28において、左端の「No」列は各行の動作データのフレーム番号（図4の`frame [0]`、`frame [1]`、・・・、`frame [FrameCount - 1]`に対応）を示している。以下の説明では、例えばフレーム番号として0が付与されている行の動作データを、「第0フレームの動作データ」と記載する。

【0266】

図26において、第1小節から第4小節までは、右手パートのみで演奏が行われ、それに伴い、第0フレームから第28フレームまでの動作データが、それぞれの押鍵タイミングに先行するタイミングで出力され、その動作データに基づいて演奏ガイドの表示が行われてゆく。このとき、左手パートは、第1小節から第5小節までは休符となっているため、その動作データは、特には図示しないが、押鍵の準備動作を示し、それに基づく演奏ガイド表示がなされる。

【0267】

図26において、第5小節から第10小節までは、右手パートは休符状態に入る。この区間に対応して、もし衝突回避制御を行わないとしたときの右手パートの動作データは、図27の第29フレームの動作データとして示されるように、第11小節の先頭の2つの音符に対応する`Position`の値が「0」（親指）で音高が「60」の「ド」音および`Position`が「1」（人差し指）で音高が「62」の「レ」音の押鍵の準備動作を指示する動作データが出力される。この第29フレームの動作データは、`start`項目の時間値「7.54」と`term`項目の時間値「17.508」として示されるように、第5小節から第10小節までずっと同じ演奏ガイド状態を維持する。。

【0268】

一方、左手パートについては、図26において、第5小節から第9小節までは、押鍵動作が行われる。そして、第7小節の先頭音（音高が「62」の「レ」音）の発音に先行してその押鍵が指示されるタイミングで、左手パートの親指に最も近い指である親指自身に

10

20

30

40

50

対して設定される動作データが、図26上の第7小節の4番目の音符（音高が「67」の「ソ」音）の押鍵の準備動作を指示する。この結果、この左手パートの親指の演奏情報の音高が、右手パートの親指に最も近い指である親指自身に対して設定される演奏情報の音高「60」（「ド」音）よりも高くなる。この結果、右手パートに関して、図26の第7小節以降も、図27に示されるように、もし衝突回避制御が行われずに第29フレームの動作データにより演奏ガイド表示が継続された場合には、左手パートの押鍵または押鍵の準備動作の演奏ガイドにおける指の指定位置が、右手パートの演奏ガイドにおける指の指定位置に衝突してしまい、演奏ガイドがうまく表示できず、このため演奏者は演奏をうまく行えなくなる。

【0269】

そこで、本実施形態では、上述の状態では、図6のステップS606のパート同士の状態比較の処理における図17のステップS1704の判定がYESとなる。さらに、第7小節の先頭音に対する演奏ガイドの開始時点で、右手パートで押鍵の準備動作の指示がされている演奏情報のうち発音開始時刻が最も早いものは第11小節の先頭の音符のものであり、左手パートで押鍵または押鍵の準備動作の指示がされている指のうち発音開始時刻が最も早いものは第7小節の先頭の音符のものである。このため、図6のステップS605のパート毎の状態確認の処理における図9のフローチャートの処理において、 $iH = 0$ （右手パート）のときにはステップS903の判定はNOとなって（図6のステップS603参照）、ステップS905で押鍵動作フラグ変数 $pushKey[0]$ に値「0」がセットされ、 $iH = 1$ （左手パート）のときにはステップS903の判定はYESとなつて、ステップS905で $pushKey[1]$ に値「1」がセットされる。この結果、図17のステップS1705の判定がNO、ステップS1707の判定がNO、ステップS1709の判定がYESとなつて、ステップS1710で、右手パート用の鍵盤横方向状態補正值 $xOffset[0]$ に、ステップS1704で判定された音高の差に応じた量だけ右側方向への移動を指示するための補正值がセットされる。これにより、図6のステップS607の動作データ出力の処理における図18と図19のフローチャートにおいて、 $iH = 0$ （右手パート）のときに、ステップS1814で、押鍵または押鍵の準備動作を指示する動作データ $frame[i]$ の各指 $ifig$ の演奏位置（鍵盤番号） $position[ifig]$ が、右手パート用の鍵盤横方向状態補正值 $xOffset[0]$ の分だけ鍵盤右横方向に移動させられる。

【0270】

この結果、本実施形態では、衝突回避制御が行われる図28の動作データ例に基づいて動作が実行される。図27の衝突回避制御が行われなかった場合の動作データ例と図28の衝突回避制御が行われる動作データ例では、衝突の発生が検知されない第0フレームから第28フレームまでは、同じ動作データ群が出力される。そして、第29フレームの出力が開始された後、図26の第7小節の先頭音（音高が「62」の「レ」音）の発音に先行してその押鍵が指示されるタイミングで、上述のように衝突が検知されて、右手パートの鍵盤横方向状態が補正された第30フレームの動作データが出力される。すなわち、図28に示されるように、第29フレームの動作データでは、第11小節の先頭の2つの音符（音高が「60」の「ド」音および音高が「62」の「レ」音）に対応する $Position$ の値が「0」（親指）と「1」（人差し指）での押鍵の準備動作を指示する動作データが出力される。続いて、衝突が検知された第30フレームの動作データでは、上記2つの音符を順次右に2半音ずつずらした音高が「62」の「レ」音、音高が「64」の「ミ」音、さらにもう1音加えて音高が「66」の「ファ#」音（右手パートの第11小節の第3音目に対応）に対応する $Position$ の値が「0」（親指）と「1」（人差し指）と「2」（中指）での押鍵の準備動作を指示する動作データが出力される。

【0271】

以下、図26の第7小節において、左手パートが鍵盤右横方向に順次押鍵されてゆくに従って、上述の衝突に対する制御によって、左手パートの指が右手パートの指に重なる分だけ、右手パートの押鍵の準備動作を指示する各指の音高が鍵盤右横方向に移動させられ

10

20

30

40

50

る動作が順次実行されてゆく。この結果、左手パートの鍵盤右横方向への押鍵移動に従って、右手パートの押鍵の準備動作の演奏ガイドも鍵盤右横方向へ移動してゆく。その後、図26の第7小節から第8小節にかけて、左手パートが鍵盤左横方向に順次押鍵されてゆくに従って、上述の衝突に対する制御によって、左手パートの指が右手パートの指に重なる分だけ、右手パートの押鍵の準備動作を指示する各指の音高が鍵盤右横方向に移動させられる動作が順次実行されてゆく。この結果、左手パートの鍵盤左横方向への押鍵移動に従って、右手パートの押鍵の準備動作の演奏ガイドも鍵盤左横方向へ移動してゆく。以上の衝突回避制御の動作が、図26および図28の破線枠で囲まれた範囲で実行されることになる。

【0272】

10

最後に、左手パートと右手パートの衝突が検知されなくなる図28の第47フレームの動作データの出力タイミングで、衝突の回避制御をしないとした場合の図27の第30フレームの動作データと同じ動作データが出力される状態に戻る。そして、このように構成された図28の動作データに基づいて、図25に示したような演奏ガイドの表示を行うことができる。具体的には、図24の動作データから図25の演奏ガイド表示を生成したのと同様の手順で、図28の動作データに基づく演奏ガイドの表示を行うことができる。これにより、ユーザは、押鍵時の演奏テクニックを習得することが可能となる。

【0273】

図29は、同じ押鍵タイミングでの右手および左手の接近により両者の演奏ガイドが衝突しうる楽譜例を示す図である。図29において、図26の場合と同様に、ト音記号が付与された上段のパートは右手パートを示し、ヘ音記号が付与された下段のパートは左手パートを示している。図26の場合と同様に、各音符の上に付与されている1から5の範囲の数字は、押鍵時の指番号を示している。図30は、図29の楽譜例に対応し、衝突回避制御を行わないときの衝突が発生する場合の動作データの出力例(その2)を示す図であり、図30(a)が右手パート、図30(b)が左手パートである。図31は、図29の楽譜例に対応し、衝突回避制御を行うときの衝突が発生する場合の動作データの出力例(その2)を示す図であり、図31(a)が右手パート、図31(b)が左手パートである。図30および図31において、図23および図24の場合と同様に、濃い灰色のデータ部分は押鍵動作が指定される部分、薄い灰色のデータ部分は押鍵の準備動作が指定される部分である。また、図30および図31において、図27および図28の場合と同様に、左端の「No」列は各行の動作データのフレーム番号を示している。

20

30

【0274】

図29において、第2小節から第3小節にかけての破線枠で囲まれた部分の3つの押鍵タイミングに先行する動作データの出力タイミングにおいて、右手パートでは、音高「62」の「レ」音と音高「68」の「ソ#」音と音高「71」の「シ」音の押鍵が指示される。一方、左手パートでは、音高「52」の「ミ」音と音高「64」の「ミ」音の押鍵が指示される。この結果、右手パートの音高「62」の「レ」音と左手パートの音高「64」の「ミ」音で、左手パートの音高のほうが高くなって、右手パートと左手パートで衝突が発生する。

【0275】

40

この場合、もし衝突開始の制御が行われない場合は、例えば図30の動作データ群が出力される。すなわち、右手パートについては、図29の破線枠の部分に対応する図30(a)の第3フレームから第7フレームの動作データ群が出力され、左手パートについては、図29の破線枠の部分に対応する図30(b)の第1フレームから第5フレームの動作データ群が出力される。このとき、図6のステップS607の動作データ出力の処理における図19のステップS1822の奥行き計算の処理(図20のフローチャートの処理)により、右手パートの第3フレームから第7フレームの動作データ群の奥行き方向の位置Zposとして、値「2」が設定される(図30(a)の太実線枠部分)。Zpos=2の場合、人差し指、中指、薬指の3本の指を鍵盤上の奥側の黒鍵上に置くことが指示される。この結果、図30(a)において、右手の親指(Position=0)は音高「6

50

2」の「レ」音の白鍵を押鍵し、右手の人差し指 (Position = 1) は音高「68」の「ソ#」音の黒鍵を押鍵し、右手の薬指 (Position = 3) は音高「71」の「シ」音の白鍵をそれぞれ押鍵することが指示される。同じく、ステップ S1822 の奥行き計算の処理により、左手パートの第1フレームから第5フレームの動作データ群の奥行き方向の位置 Zpos として、値「1」が設定される (図30 (b) の太実線枠部分)。Zpos = 1 の場合、5本の指全てを鍵盤上の手前側の白鍵上に置くことが指示される。この結果、図30 (b) において、左手の小指 (Position = 4) は音高「52」の「ミ」音の白鍵を押鍵し、左手の親指 (Position = 0) は音高「64」の「ミ」音の白鍵を押鍵することが指示される。

【0276】

10

この結果、衝突回避制御を行わないとした場合の図30の動作データの例では、図29の破線枠に対応する右手パートに対応する図30 (a) の第3フレームから第7フレームの動作データと左手パートに対応する図30 (b) の第1フレームから第5フレームの動作データの出力タイミングにおいて、右手パートの親指に対する音高「62」の「レ」音の白鍵の押鍵指示と、左手パートの親指に対する音高「64」の「ミ」音の白鍵の押鍵指示とが、親指同士がクロスして衝突が発生してしまう。従って、演奏ガイドがうまく表示できず、このため演奏者は、うまく演奏を行うことができなくなってしまう。

【0277】

そこで、本実施形態では、以下に示す衝突回避制御によって、例えば図31 (a) (右手パート) および図31 (b) (左手パート) の動作データが生成され、その動作データに基づいて演奏ガイドが実行される。すなわち、上述の状態では、図6のステップ S606 のパート同士の状態比較の処理における図17のステップ S1704 の判定が YES となる。さらに、図29の破線枠の演奏ガイドの開始時点で、右手パートと左手パートでは同時に押鍵が発生するため、図6のステップ S605 のパート毎の状態確認の処理における図9のフローチャートの処理において、iH = 0 (右手パート) および iH = 1 (左手パート) とともにステップ S903 の判定は YES となって (図6のステップ S603 参照)、ステップ S904 で押鍵動作フラグ変数 pushKey[0] および pushKey[1] とともに値「1」がセットされる。この結果、図17のステップ S1705 の判定が YES となって、ステップ S1706 で、鍵盤奥方向 (黒鍵側) にある右手パートに対応する鍵盤奥行き方向状態補正值 zOffset[0] に、例えば正の値「2」がセットされ、手前方向 (白鍵側) にある左手パートに対応する鍵盤奥行き方向状態補正值 zOffset[1] には、現在の奥行き方向の位置が最低値「1」であるから、補正無しを示す値「0」がセットされる。この結果、図6のステップ S607 の動作データ出力の処理における図18と図19のフローチャートにおけるステップ S1823 の処理により、iH = 0 (右手パート) のときに、奥行き方向の位置 zpos = 2 (図30 (a) の太実線枠) に鍵盤奥行き方向状態補正值 zOffset[0] = 2 が加算され、iH = 0 (左手パート) のときに、奥行き方向の位置 zpos = 1 (図30 (b) の太実線枠) に鍵盤奥行き方向状態補正值 zOffset[1] = 0 が加算される。

【0278】

以上の制御処理により、右手パートについては、図29の破線枠の部分に対応する図31 (a) の第3フレームから第7フレームの動作データ群において、太実線枠として示されるように、奥行き方向の位置 Zpos として、値「4」が設定される。また、左手パートについては、図29の破線枠の部分に対応する図31 (b) の第1フレームから第5フレームの動作データ群において、太実線枠として示されるように、奥行き方向の位置 Zpos として、値「1」が設定される。Zpos = 4 の場合、5本の指全てを鍵盤上の奥側の黒鍵上に置くことが指示される。この結果、右手パート全体が鍵盤奥の黒鍵側に移動させられ、左手パート全体は鍵盤手前の白鍵側に置かれるため、両手間の衝突が回避されることになる。すなわち、衝突回避制御が行われた場合の図31の動作データの例では、図29の破線枠に対応する右手パートに対応する図31 (a) の第3フレームから第7フレームの動作データと左手パートに対応する図31 (b) の第1フレームから第5フレーム

20

30

40

50

の動作データの出力タイミングにおいて、右手パートの親指に対する音高「6 2」の「レ」音の白鍵の押鍵指示と、左手パートの親指に対する音高「6 4」の「ミ」音の白鍵の押鍵指示において、親指同士のクロスによる衝突を回避でき、演奏者は、うまく演奏を行うことが可能となる。そして、このように構成された図3 1の動作データに基づいて、図2 5に示したような演奏ガイドの表示を行うことができる。具体的には、図2 4の動作データから図2 5の演奏ガイド表示を生成したのと同様の手順で、図3 1の動作データに基づく演奏ガイドの表示を行うことができる。これにより、ユーザは、押鍵時の演奏テクニックを習得することが可能となる。

【0 2 7 9】

以上説明したように、本実施形態によれば、演奏情報と運指情報のみから、押鍵のための情報にとどまらない、演奏動作を遂行する上でのテクニックに相当する情報を生成することが可能となるため、それにより運指とともに演奏動作上のテクニックも演奏ガイドすることが可能となる。また、準備に必要な箇所についてのユーザへの注意を喚起する情報を自動で生成できるなどの、楽器演奏の練習についての有用な情報の提供が可能になる。さらに、演奏情報に、熟達者による付加情報をつける必要がなくなるというメリットも生まれる。

【0 2 8 0】

さらに本実施形態では、左手と右手のパートで練習を行うための演奏ガイドを実行するときに、楽譜上、左手パートと右手パートの間で、演奏される指同士が衝突しそうになったり、あるいは次に演奏する準備を行うべき指を置く場所と演奏中の指が衝突しそうな

場合に、それらの状態を検知して補正することが可能となる。

以上の各実施形態に関して、更に以下の付記を開示する。

【0 2 8 1】

(付記1)

楽曲を構成する楽音夫々の発音タイミング情報と、音高情報と、演奏に用いる手及び指を示す情報と、を含む演奏情報を記憶する記憶部から読み出す読み出し手段と、

当該読み出された演奏情報それぞれに基づいて、前記右手及び左手の指夫々が複数の演奏操作子のいずれの位置にあるかを示す指位置を推定する位置推定手段と、

前記位置推定手段により推定された前記右手及び左手の指位置同士の関係が予め定められた条件を満たすか否かを判別する判別手段と、

前記判別手段にて条件を満たすと判別された場合に、前記位置推定手段にて推定された前記右手及び左手の位置のうち、少なくとも一方の手の指位置を補正する位置補正手段と、

前記読み出された演奏情報及び前記各手の指位置に基づいて、演奏操作ガイド情報を生成するガイド情報生成手段と、

を備えることを特徴とする演奏操作ガイド装置。

(付記2)

前記判定手段は、前記指位置推定手段により推定された右手の各指位置と左手の各指位置のうち、前記左手の親指位置が、前記右手の親指位置より右側にあるか否かを判別することを特徴とする付記1に記載の演奏操作ガイド装置。

(付記3)

前記指位置補正手段は、

前記読み出された演奏情報に含まれる発音タイミング情報に基づき、前記演奏装置を演奏している指が前記右手及び左手のいずれに存在しているかを判別する演奏指判別手段をさらに有する、

ことを特徴とする付記1または2のいずれかに記載の演奏操作ガイド装置。

(付記4)

前記指位置補正手段は、

前記演奏指判別手段により、前記右手に前記演奏装置を演奏している指が存在している

と判別された場合は、前記左手の指位置を、当該左手方向に補正する一方、前記左手に前記演奏装置を演奏している指が存在していると判別された場合は、前記右手の指位置を、当該右手方向に補正する、

ことを特徴とする付記 3 に記載の演奏操作ガイド装置。

(付記 5)

前記指位置補正手段は、

前記演奏指判別手段により、前記右手及び左手の両方に前記演奏装置を演奏している指が存在していると判別された場合は、前記右手及び左手の少なくとも一方の指位置を、当該右手方向及び左手方向以外の方向に補正する、

ことを特徴とする付記 3 または 4 のいずれかに記載の演奏操作ガイド装置。

10

(付記 6)

付記 1 から 5 のいずれかに記載の演奏操作ガイド装置と、

演奏者に演奏操作を行わせる演奏操作子と、

前記演奏操作子における演奏操作に応じて楽音を生成して出力する楽音出力部とを備える電子楽器。

(付記 7)

演奏操作ガイド装置が、

楽曲を構成する楽音夫々の発音タイミング情報と、音高情報と、演奏に用いる手及び指を示す情報と、を含む演奏情報を記憶する記憶部から読み出し、

当該読み出された演奏情報それぞれに基づいて、前記右手及び左手の指夫々が複数の演奏操作子のいずれの位置にあるかを示す指位置を推定し、

20

前記推定された前記右手及び左手の指位置同士の関係が予め定められた条件を満たすか否か判別し、

前記条件を満たすと判別された場合に、前記推定された右手及び左手の位置のうち、少なくとも一方の手の指位置を補正し、

前記読み出された演奏情報及び前記各手の指位置に基づいて、演奏操作ガイド情報を生成する、演奏操作ガイド方法。

(付記 8)

楽曲を構成する楽音夫々の発音タイミング情報と、音高情報と、演奏に用いる手及び指を示す情報と、を含む演奏情報を記憶する記憶部から読み出す読み出しステップ、

30

当該読み出された演奏情報それぞれに基づいて、前記右手及び左手の指夫々が複数の演奏操作子のいずれの位置にあるかを示す指位置を推定する位置推定ステップと、

前記推定された前記右手及び左手の指位置同士の関係が予め定められた条件を満たすか否か判別する判別ステップと、

前記条件を満たすと判別された場合に、前記推定された右手及び左手の位置のうち、少なくとも一方の手の指位置を補正する位置補正ステップと、

前記読み出された演奏情報及び前記各手の指位置に基づいて、演奏操作ガイド情報を生成するガイド情報生成ステップと、

をコンピュータに実行させる演奏操作ガイドプログラム。

【符号の説明】

40

【 0 2 8 2 】

1 0 0 演奏操作ガイド装置

1 0 1 演奏情報読み出し手段

1 0 2 操作状態推定手段

1 0 3 パート同士の状態比較手段

1 0 4 演奏操作ガイド手段

1 0 5 記憶部

1 0 6 第 1 の演奏操作状態

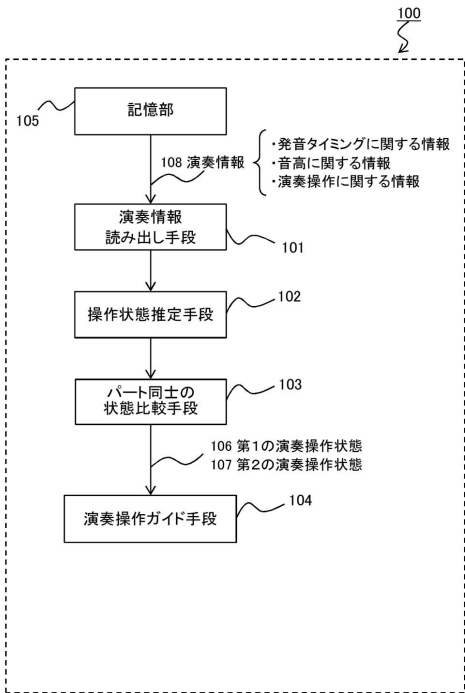
1 0 7 第 2 の演奏操作状態

1 0 8 演奏情報

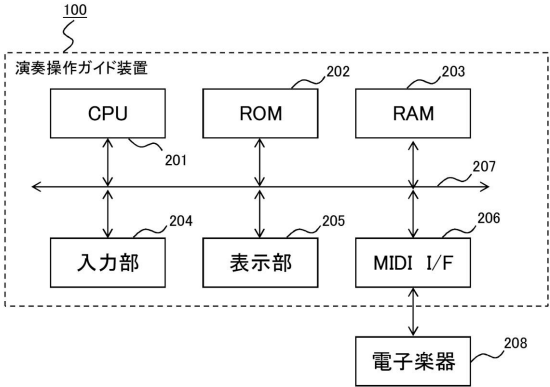
50

- 201CPU
- 202ROM
- 203RAM
- 204入力部
- 205表示部
- 206MIDI I/F (MIDIインタフェース部)
- 207バス
- 208電子楽器

【図1】



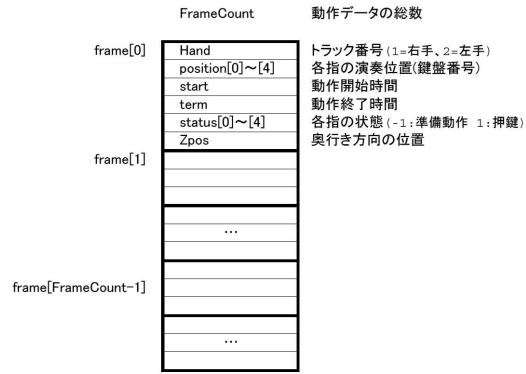
【図2】



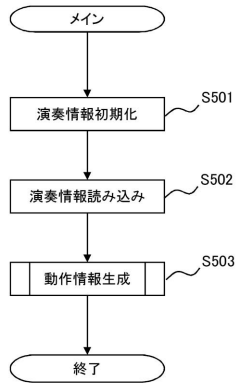
【図3】

Note[0]	Index	インデックス番号
	Time	発音開始時刻
	Gate	発音継続時間
	Pitch	音高
	Hand	トラック番号(0=右手、1=左手)
	Finger	指番号
Note[1]		
	...	
Note[N-1]		
	...	

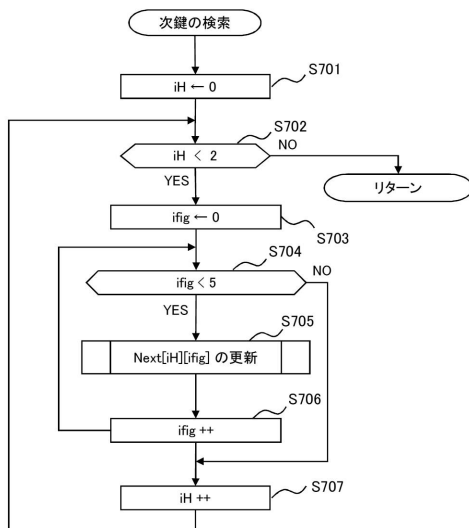
【図 4】



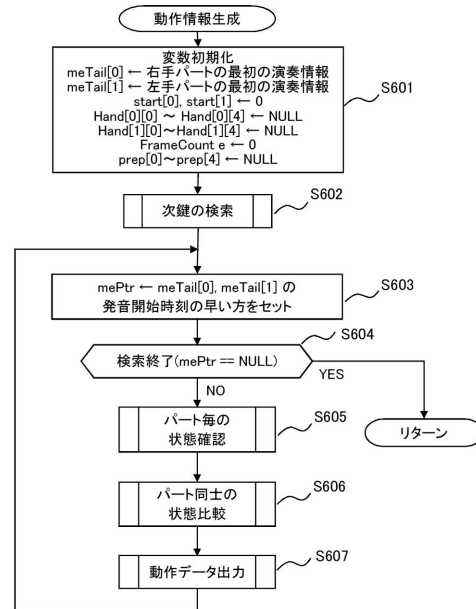
【図 5】



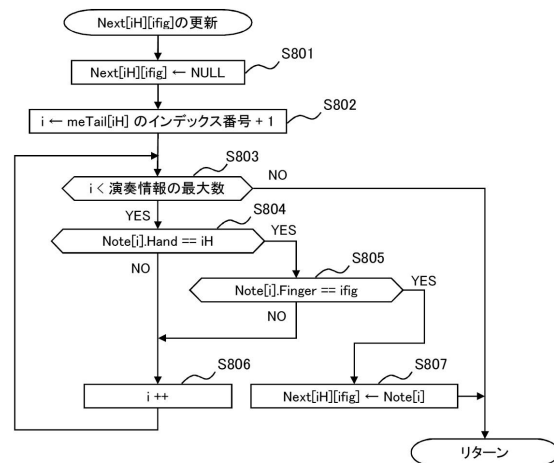
【図 7】



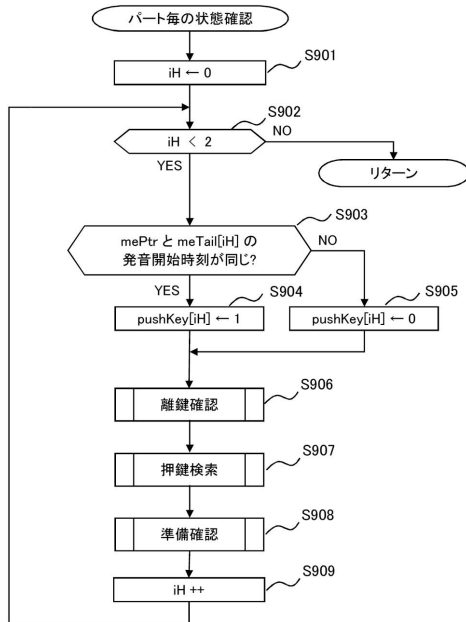
【図 6】



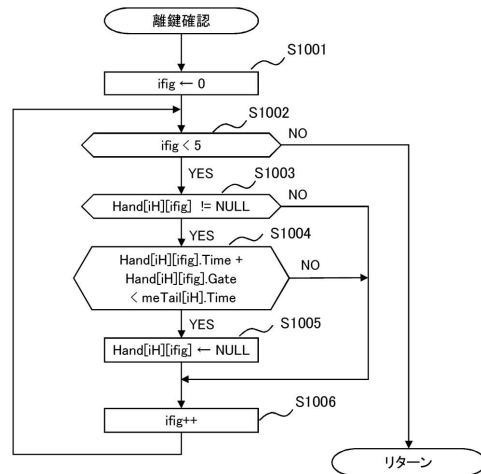
【図 8】



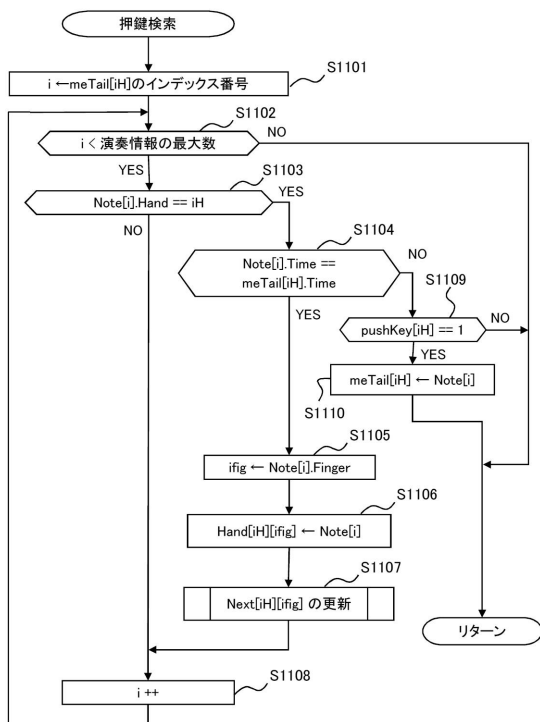
【図 9】



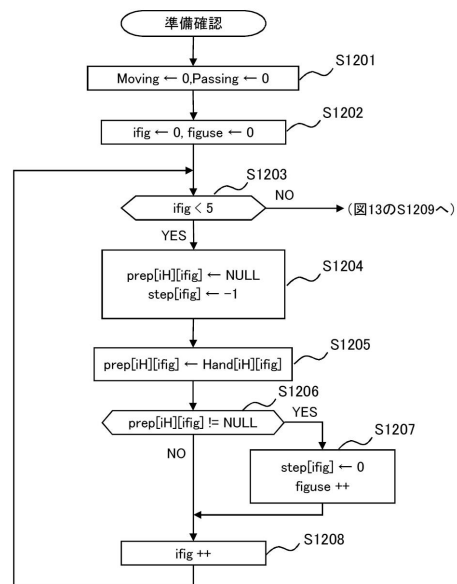
【図 10】



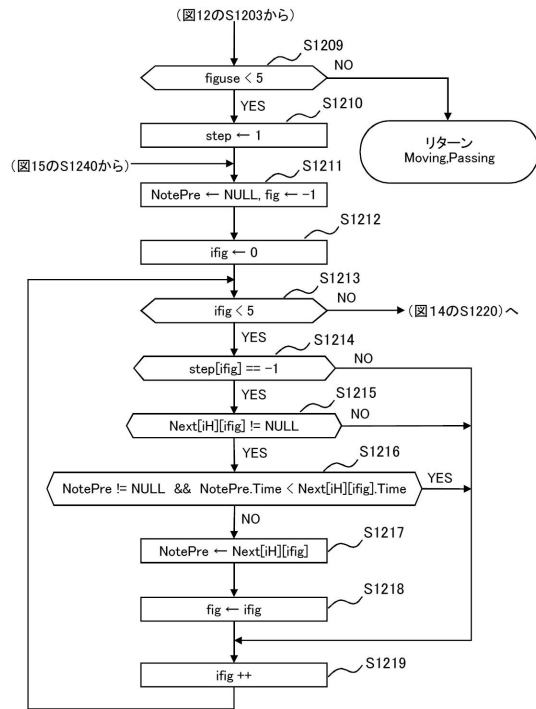
【図 11】



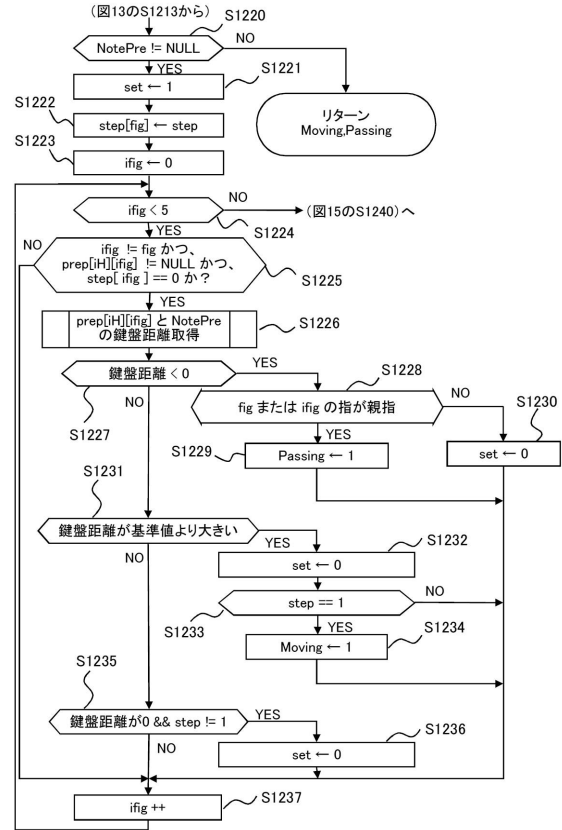
【図 12】



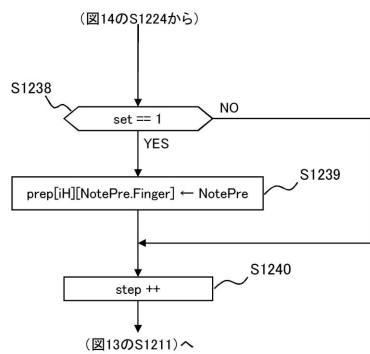
【図 13】



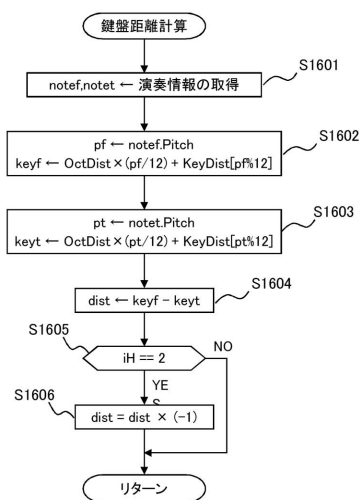
【図 14】



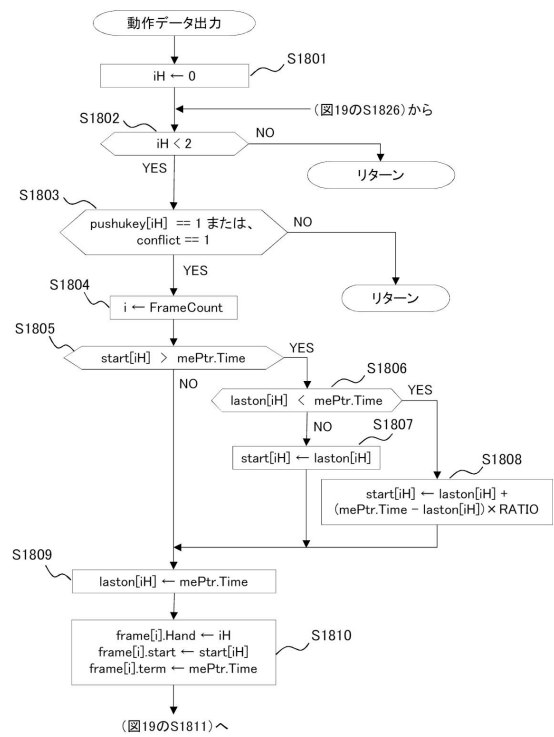
【図 15】



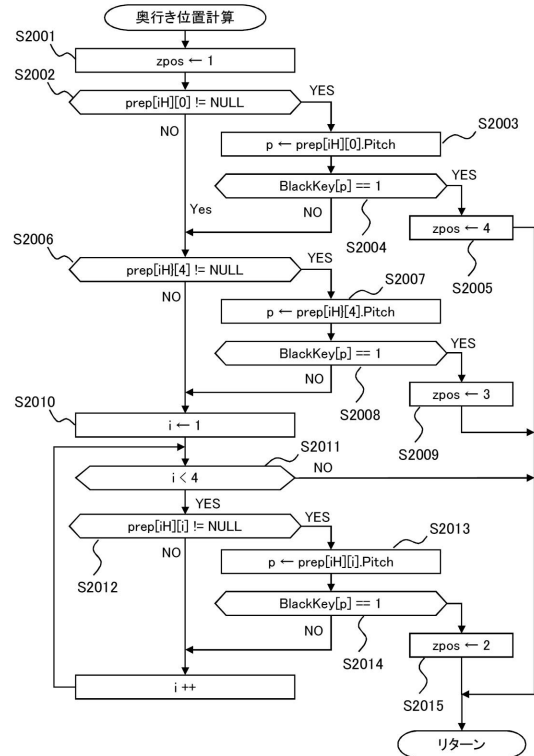
【図 16】



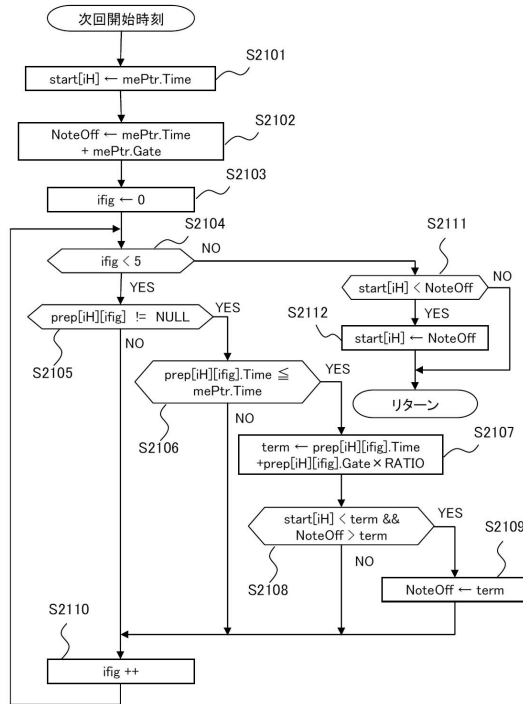
【 図 1 8 】



【 図 2 0 】



【図 2 1】



【図 2 2】

index	Time	Gate	Pitch	Hand	Finger
0	0.004	0.507	74	1	5
1	0.512	0.233	67	1	1
2	0.76	0.232	69	1	2
3	1.008	0.229	71	1	3
4	1.253	0.231	72	1	4
5	1.501	0.509	74	1	5
6	2.012	0.195	67	1	1
7	2.506	0.192	67	1	1
8	3.004	0.509	76	1	3
9	3.515	0.233	72	1	1
10	3.763	0.232	74	1	2
11	4.011	0.229	76	1	3
12	4.256	0.231	78	1	4
13	4.504	0.509	79	1	5
14	5.015	0.195	67	1	1
15	5.51	0.192	67	1	1
16	6.008	0.509	72	1	3
17	6.519	0.233	74	1	4
18	6.767	0.232	72	1	3
19	7.015	0.229	71	1	2
20	7.26	0.231	69	1	1
21	7.508	0.509	71	1	3
22	8.019	0.233	72	1	4
23	8.267	0.232	71	1	3
24	8.515	0.229	69	1	2
25	8.76	0.231	67	1	1
26	9.008	0.509	66	1	2
27	9.519	0.233	67	1	1
28	9.767	0.232	69	1	2
29	10.015	0.229	71	1	3
30	10.26	0.231	67	1	1
31	10.444	0.063	71	1	3
32	10.508	1.475	69	1	2

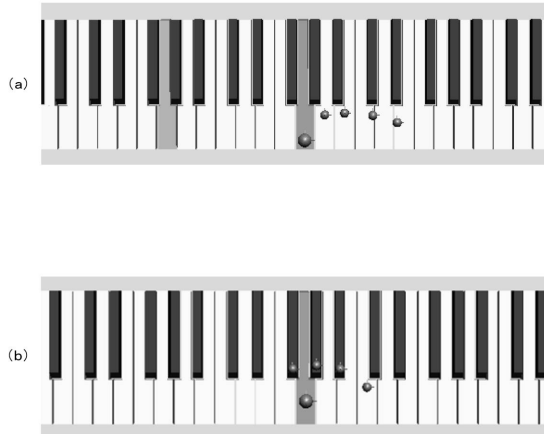
【図 2 3】

Position					start	term	Status					Zpos
0	1	2	3	4			0	1	2	3	4	
-1	-1	-1	-1	74	0	0.004	0	0	0	0	1	1
67	-1	-1	-1	-1	0.308	0.512	1	0	0	0	0	1
-1	69	-1	-1	-1	0.651	0.76	0	1	0	0	0	1
-1	-1	71	-1	-1	0.899	1.008	0	0	1	0	0	1
-1	-1	-1	72	-1	1.145	1.253	0	0	0	1	0	1
-1	-1	-1	-1	74	1.391	1.501	0	0	0	0	1	1
67	-1	-1	-1	-1	1.806	2.012	1	0	0	0	0	1
67	-1	-1	-1	-1	2.43	2.506	1	0	0	0	0	1
-1	-1	76	-1	-1	2.621	3.004	0	0	1	0	0	1
72	-1	-1	-1	-1	3.309	3.515	1	0	0	0	0	1
-1	74	-1	-1	-1	3.654	3.763	0	1	0	0	0	1
-1	-1	76	-1	-1	3.902	4.011	0	0	1	0	0	1
-1	-1	-1	78	-1	4.148	4.256	0	0	0	1	0	2
-1	-1	-1	-1	79	4.394	4.504	0	0	0	0	1	1
67	-1	-1	-1	-1	4.809	5.015	1	0	0	0	0	1
67	-1	-1	-1	-1	5.434	5.51	1	0	0	0	0	1
-1	-1	72	-1	-1	5.625	6.008	0	0	1	0	0	1
-1	-1	-1	74	-1	6.313	6.519	0	0	0	1	0	1
-1	-1	72	-1	-1	6.658	6.767	0	0	1	0	0	1
-1	71	-1	-1	-1	6.906	7.015	0	1	0	0	0	1
69	-1	-1	-1	-1	7.152	7.26	1	0	0	0	0	1
-1	-1	71	-1	-1	7.398	7.508	0	0	1	0	0	1
-1	-1	-1	72	-1	7.813	8.019	0	0	0	1	0	1
-1	-1	71	-1	-1	8.158	8.267	0	0	1	0	0	1
-1	69	-1	-1	-1	8.406	8.515	0	1	0	0	0	1
* 67	-1	-1	-1	-1	8.652	8.76	1	0	0	0	0	1
-1	66	-1	-1	-1	8.898	9.008	0	1	0	0	0	2
67	-1	-1	-1	-1	9.313	9.519	1	0	0	0	0	1
-1	69	-1	-1	-1	9.658	9.767	0	1	0	0	0	1
-1	-1	71	-1	-1	9.906	10.015	0	0	1	0	0	1
67	-1	-1	-1	-1	10.152	10.26	1	0	0	0	0	1
-1	-1	71	-1	-1	10.398	10.444	0	0	1	0	0	1
-1	69	-1	-1	-1	10.481	10.508	0	1	0	0	0	1

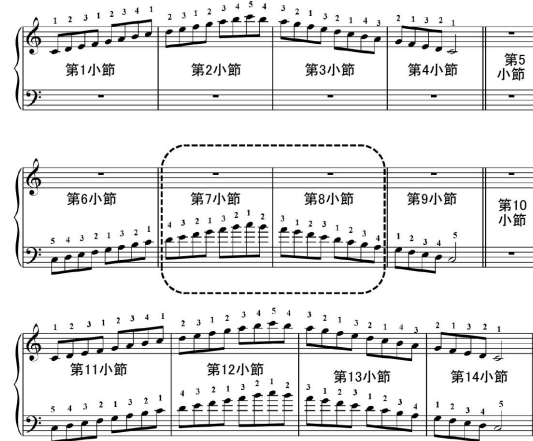
【図 2 4】

Position					start	term	Status					Zpos
0	1	2	3	4			0	1	2	3	4	
67	69	-1	-1	74	0	0.004	-1	-1	0	0	1	1
67	69	71	-1	-1	0.308	0.512	1	-1	-1	0	0	1
-1	69	71	72	-1	0.651	0.76	0	1	-1	-1	0	1
-1	-1	71	72	74	0.899	1.008	0	0	1	-1	-1	1
67	-1	-1	72	74	1.145	1.253	-1	0	0	1	-1	1
67	-1	-1	-1	74	1.391	1.501	-1	0	0	0	1	1
67	-1	-1	-1	-1	1.806	2.012	1	0	0	0	0	1
67	-1	-1	-1	-1	2.43	2.506	1	0	0	0	0	1
72	74	76	-1	-1	2.621	3.004	-1	-1	1	0	0	1
72	74	76	-1	-1	3.309	3.515	1	-1	-1	0	0	1
-1	74	76	78	-1	3.654	3.763	0	1	-1	-1	0	2
-1	-1	76	78	79	3.902	4.011	0	0	1	-1	-1	2
-1	-1	-1	78	79	4.148	4.256	0	0	0	1	-1	2
-1	-1	-1	-1	79	4.394	4.504	0	0	0	0	1	1
67	-1	72	-1	-1	4.809	5.015	1	0	-1	0	0	1
67	-1	72	74	-1	5.434	5.51	1	0	-1	-1	0	1
-1	-1	72	74	-1	5.625	6.008	0	0	1	-1	0	1
-1	71	72	74	-1	6.313	6.519	0	-1	-1	1	0	1
69	71	72	-1	-1	6.658	6.767	-1	-1	-1	1	0	1
69	71	-1	-1	-1	6.906	7.015	-1	1	0	0	0	1
69	-1	71	72	-1	7.152	7.26	1	0	-1	-1	0	1
-1	-1	71	72	-1	7.398	7.508	0	0	1	-1	0	1
-1	69	71	72	-1	7.813	8.019	0	-1	-1	1	0	1
67	69	71	-1	-1	8.158	8.267	-1	-1	1	0	0	1
* 67	69	-1	-1	-1	8.406	8.515	-1	1	0	0	0	1
67	66	-1	-1	-1	8.652	8.76	1	-1	0	0	0	2
67	66	-1	-1	-1	8.898	9.008	-1	1	0	0	0	1
67	69	71	-1	-1	9.313	9.519	1	-1	-1	0	0	1
67	69	71	-1	-1	9.658	9.767	-1	1	-1	0	0	1
67	-1	71	-1	-1	9.906	10.015	-1	0	-1	0	0	1
67	69	71	-1	-1	10.152	10.26	1	-1	-1	0	0	1
-1	69	71	-1	-1	10.398	10.444	0	-1	1	0	0	1
67	69	-1	-1	-1	10.481	10.508	-1	1	0	0	0	1

【図 25】



【図 26】



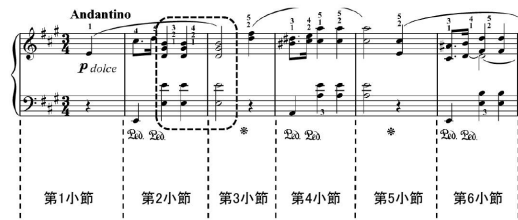
【図 27】

No	Position					start	term	Status					Zpos	
	0	1	2	3	4			0	1	2	3	4		
0	60	62	64	-1	-1	0	0	0	1	-1	-1	0	0	1
1	-1	62	64	-1	-1	0.135	0.25	0	1	-1	0	0	0	1
2	65	-1	64	-1	-1	0.385	0.5	-1	0	1	1	0	0	1
3	65	67	69	-1	-1	0.635	0.75	1	-1	-1	0	0	0	1
4	-1	67	69	71	-1	0.885	1	0	1	-1	-1	0	0	1
5	-1	-1	69	71	-1	1.135	1.25	0	0	1	-1	0	0	1
6	72	-1	-1	71	-1	1.385	1.5	-1	0	0	1	0	0	1
7	72	74	76	-1	-1	1.635	1.75	1	-1	-1	0	0	0	1
8	-1	74	76	-1	-1	1.885	2	0	1	-1	0	0	0	1
9	77	-1	76	-1	-1	2.135	2.25	-1	0	1	0	0	0	1
10	77	79	81	-1	-1	2.385	2.5	1	-1	-1	0	0	0	1
11	-1	79	81	83	-1	2.635	2.75	0	1	-1	-1	-1	0	1
12	-1	-1	81	83	84	2.885	3	0	0	0	-1	-1	-1	1
13	-1	-1	-1	83	84	3.135	3.25	0	0	0	0	0	-1	1
14	-1	-1	81	83	84	3.385	3.5	0	0	-1	-1	-1	1	1
15	-1	79	81	83	-1	3.635	3.75	0	-1	-1	-1	1	0	1
16	77	79	81	-1	-1	3.885	4	-1	-1	1	0	0	0	1
17	77	79	-1	-1	-1	4.135	4.25	-1	1	0	0	0	0	1
18	77	-1	76	-1	-1	4.385	4.5	1	0	-1	0	0	0	1
19	72	74	76	-1	-1	4.635	4.75	-1	-1	1	0	0	0	1
20	72	74	-1	-1	-1	4.885	5	-1	1	0	0	0	0	1
21	72	-1	-1	71	-1	5.135	5.25	1	0	0	-1	0	0	1
22	-1	67	69	71	-1	5.385	5.5	0	-1	-1	1	0	0	1
23	65	67	69	-1	-1	5.635	5.75	-1	-1	1	0	0	0	1
24	65	67	-1	-1	-1	5.885	6	-1	1	0	0	0	0	1
25	65	-1	64	-1	-1	6.135	6.25	1	0	-1	0	0	0	1
26	60	62	64	-1	-1	6.385	6.5	-1	-1	1	0	0	0	1
27	60	62	-1	-1	-1	6.635	6.75	-1	1	0	0	0	0	1
28	60	62	-1	-1	-1	6.885	7	1	-1	0	0	0	0	1
29	60	62	-1	-1	-1	7.54	17.508	-1	-1	0	0	0	0	1
30	60	62	64	-1	-1	17.508	20	1	-1	-1	0	0	0	1
31	-1	62	64	-1	-1	20.135	20.25	0	1	-1	0	0	0	1
32	65	-1	64	-1	-1	20.385	20.5	-1	0	1	0	0	0	1
33	65	67	69	-1	-1	20.635	20.75	1	-1	-1	0	0	0	1
34	-1	67	69	71	-1	20.885	21	0	1	-1	-1	0	0	1
35	-1	-1	69	71	-1	21.135	21.25	0	0	1	-1	0	0	1

【図 28】

No	Position				start	term	Status				Zpos		
	0	1	2	3			4	0	1	2		3	4
0	60	62	64	-1	-1	0	0	1	-1	-1	0	0	1
1	-1	62	64	-1	-1	0.135	0.25	0	1	-1	0	0	1
2	65	-1	64	-1	-1	0.385	0.5	-1	0	1	0	0	1
3	65	67	69	-1	-1	0.635	0.75	-1	-1	-1	0	0	1
4	-1	67	69	71	-1	0.885	1	0	1	-1	-1	0	1
5	-1	-1	69	71	-1	1.135	1.25	0	0	1	-1	0	1
6	72	-1	-1	71	-1	1.385	1.5	-1	0	0	1	0	1
7	72	74	76	-1	-1	1.635	1.75	1	-1	-1	0	0	1
8	-1	74	76	-1	-1	1.885	2	0	1	-1	0	0	1
9	77	76	-1	-1	-1	2.135	2.25	-1	0	1	0	0	1
10	77	79	81	-1	-1	2.385	2.5	1	-1	-1	0	0	1
11	-1	79	81	83	-1	2.635	2.75	0	1	-1	-1	0	1
12	-1	-1	81	83	84	2.885	3	0	0	1	-1	-1	1
13	-1	-1	-1	83	84	3.135	3.25	0	0	0	1	-1	1
14	-1	-1	81	83	84	3.385	3.5	0	0	-1	-1	1	1
15	-1	79	81	83	-1	3.635	3.75	0	-1	-1	1	0	1
16	77	79	81	-1	-1	3.885	4	-1	-1	1	0	0	1
17	77	79	-1	-1	-1	4.135	4.25	-1	1	0	0	0	1
18	77	-1	76	-1	-1	4.385	4.5	1	0	-1	0	0	1
19	72	74	76	-1	-1	4.635	4.75	-1	-1	1	0	0	1
20	72	74	-1	-1	-1	4.885	5	-1	1	0	0	0	1
21	72	-1	-1	71	-1	5.135	5.25	1	0	0	-1	0	1
22	-1	67	69	71	-1	5.385	5.5	0	-1	-1	1	0	1
23	65	67	69	-1	-1	5.635	5.75	-1	-1	1	0	0	1
24	65	67	-1	-1	-1	5.885	6	-1	1	0	0	0	1
25	65	-1	64	-1	-1	6.135	6.25	1	0	-1	0	0	1
26	60	62	64	-1	-1	6.385	6.5	-1	-1	1	0	0	1
27	60	62	-1	-1	-1	6.635	6.75	-1	1	0	0	0	1
28	60	62	-1	-1	-1	6.885	7	1	-1	0	0	0	1
29	60	62	-1	-1	-1	7.54	10.508	-1	-1	0	0	0	1
30	62	64	66	-1	-1	10.508	11.25	-1	-1	-1	0	0	1
31	62	64	66	-1	-1	11.475	11.5	-1	-1	-1	0	0	1
32	62	64	66	-1	-1	11.725	11.75	-1	-1	-1	0	0	1
33	68	70	72	-1	-1	11.975	12	-1	-1	-1	0	0	1
34	70	72	74	-1	-1	12.225	12.25	-1	-1	-1	0	0	1
35	70	72	74	-1	-1	12.475	12.5	-1	-1	-1	0	0	1
36	70	72	74	-1	-1	12.725	12.75	-1	-1	-1	0	0	1
37	76	78	80	-1	-1	12.975	13	-1	-1	-1	0	0	1
38	76	78	80	-1	-1	13.225	13.25	-1	-1	-1	0	0	1
39	76	78	80	-1	-1	13.475	13.5	-1	-1	-1	0	0	1
40	74	76	78	-1	-1	13.725	13.75	-1	-1	-1	0	0	1
41	70	72	74	-1	-1	13.975	14	-1	-1	-1	0	0	1
42	70	72	74	-1	-1	14.225	14.25	-1	-1	-1	0	0	1
43	68	70	72	-1	-1	14.475	14.5	-1	-1	-1	0	0	1
44	64	66	68	-1	-1	14.725	14.75	-1	-1	-1	0	0	1
45	64	66	68	-1	-1	14.975	15	-1	-1	-1	0	0	1
46	62	64	66	-1	-1	15.225	15.25	-1	-1	-1	0	0	1
47	60	62	64	-1	-1	15.475	20	-1	-1	-1	0	0	1
48	-1	62	64	-1	-1	20.135	20.25	0	1	-1	0	0	1
49	65	-1	64	-1	-1	20.385	20.5	-1	0	1	0	0	1

【図 29】



【図 30】

(a)

No	Position					start	term	Status					Zpos
	0	1	2	3	4			0	1	2	3	4	
0	64	-1	-1	-1	74	0	1.5	1	0	0	0	-1	4
1	-1	68	-1	73	74	1.95	2.25	0	-1	0	1	-1	3
2	-1	68	-1	-1	74	2.587	2.812	0	-1	0	0	1	2
3	62	68	-1	71	-1	2.924	3	1	1	0	1	0	2
4	62	68	-1	71	-1	3.45	3.69	-1	-1	0	-1	0	2
5	62	68	-1	71	-1	3.69	3.75	1	1	0	1	0	2
6	62	68	-1	71	-1	4.2	4.44	-1	-1	0	-1	0	2
7	62	68	-1	71	-1	4.44	4.5	1	1	0	1	0	2
8	72	74	-1	78	-1	5.4	6	-1	1	0	1	0	2
9	72	73	75	76	-1	6.45	6.75	1	-1	1	-1	0	2
10	73	73	-1	76	-1	7.087	7.312	-1	1	0	1	0	4
11	73	73	-1	-1	81	7.424	7.5	1	-1	0	0	1	4
12	73	73	-1	-1	81	7.95	8.19	1	-1	0	0	-1	4
13	-1	73	-1	-1	81	8.19	8.25	0	1	0	0	1	2
14	-1	73	-1	-1	81	8.7	8.94	0	-1	0	0	-1	3

(b)

No	Position					start	term	Status					Zpos
	0	1	2	3	4			0	1	2	3	4	
0	-1	-1	-1	-1	40	0	2.25	0	0	0	0	1	1
1	64	-1	-1	-1	52	2.7	3	1	0	0	0	1	1
2	64	-1	-1	-1	52	3.45	3.69	-1	0	0	0	-1	1
3	64	-1	-1	-1	52	3.69	3.75	1	0	0	0	1	1
4	64	-1	-1	-1	52	4.2	4.44	-1	0	0	0	-1	1
5	64	-1	-1	-1	52	4.44	4.5	1	0	0	0	1	1
6	-1	-1	-1	-1	45	5.4	6.75	0	0	0	0	1	1
7	64	-1	-1	57	-1	7.2	7.5	1	0	0	1	0	1
8	64	-1	-1	57	-1	7.95	8.19	-1	0	0	-1	0	1
9	64	-1	-1	57	-1	8.19	8.25	1	0	0	1	0	1
10	64	-1	-1	57	-1	8.7	8.94	-1	0	0	-1	0	1

【図 31】

(a)

No	Position					start	term	Status					Zpos
	0	1	2	3	4			0	1	2	3	4	
0	64	-1	-1	-1	74	0	1.5	1	0	0	0	-1	4
1	-1	68	-1	73	74	1.95	2.25	0	-1	0	1	-1	3
2	-1	68	-1	-1	74	2.587	2.812	0	-1	0	0	1	2
3	62	68	-1	71	-1	2.924	3	1	1	0	1	0	4
4	62	68	-1	71	-1	3.45	3.69	-1	-1	0	-1	0	4
5	62	68	-1	71	-1	3.69	3.75	1	1	0	1	0	4
6	62	68	-1	71	-1	4.2	4.44	-1	-1	0	-1	0	4
7	62	68	-1	71	-1	4.44	4.5	1	1	0	1	0	4
8	72	74	-1	78	-1	5.4	6	-1	1	0	1	0	2
9	72	73	75	76	-1	6.45	6.75	1	-1	1	-1	0	2
10	73	73	-1	76	-1	7.087	7.312	-1	1	0	1	0	4
11	73	73	-1	-1	81	7.424	7.5	1	-1	0	0	1	4
12	73	73	-1	-1	81	7.95	8.19	1	-1	0	0	-1	4
13	-1	73	-1	-1	81	8.19	8.25	0	1	0	0	1	2
14	-1	73	-1	-1	81	8.7	8.94	0	-1	0	0	-1	3

(b)

No	Position					start	term	Status					Zpos
	0	1	2	3	4			0	1	2	3	4	
0	-1	-1	-1	-1	40	0	2.25	0	0	0	0	1	1
1	64	-1	-1	-1	52	2.7	3	1	0	0	0	1	1
2	64	-1	-1	-1	52	3.45	3.69	-1	0	0	0	-1	1
3	64	-1	-1	-1	52	3.69	3.75	1	0	0	0	1	1
4	64	-1	-1	-1	52	4.2	4.44	-1	0	0	0	-1	1
5	64	-1	-1	-1	52	4.44	4.5	1	0	0	0	1	1
6	-1	-1	-1	-1	45	5.4	6.75	0	0	0	0	1	1
7	64	-1	-1	57	-1	7.2	7.5	1	0	0	1	0	1
8	64	-1	-1	57	-1	7.95	8.19	-1	0	0	-1	0	1
9	64	-1	-1	57	-1	8.19	8.25	1	0	0	1	0	1
10	64	-1	-1	57	-1	8.7	8.94	-1	0	0	-1	0	1

フロントページの続き

(58)調査した分野(Int.Cl. , D B 名)

G 1 0 G	1 / 0 2
G 1 0 H	1 / 0 0