(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 265 221 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
11.12.2002 Bulletin 2002/50

(51) Int Cl.[7]: **G10H 1/00**

(21) Application number: **01401485.6**

(22) Date of filing: **08.06.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(71) Applicant: **Sony France S.A.**
92110 Clichy (FR)

(72) Inventor: **Pachet, Francois**
75010 Paris (FR)

(74) Representative: **Bertrand, Didier et al**
c/o S.A. FEDIT-LORIOT & AUTRES
CONSEILS EN PROPRIETE INDUSTRIELLE
38, Avenue Hoche
75008 Paris (FR)

(54) Automatic music improvisation method and device

(57) The invention provides a method for automatically generating music that constitutes an improvisation, characterised in that it comprises the steps of:

- establishing a data base of music patterns (42),
- detecting current music data (12),
- generating with reference to said music patterns an improvisation as a continuation in real time of said current music data (4).

In this way, an improvisation is performed in a seamless manner as continuation of a musical piece played by an instrument.

The invention also provides the possibility of controlling the direction of improvisation through control data, e.g. in the form of notes.

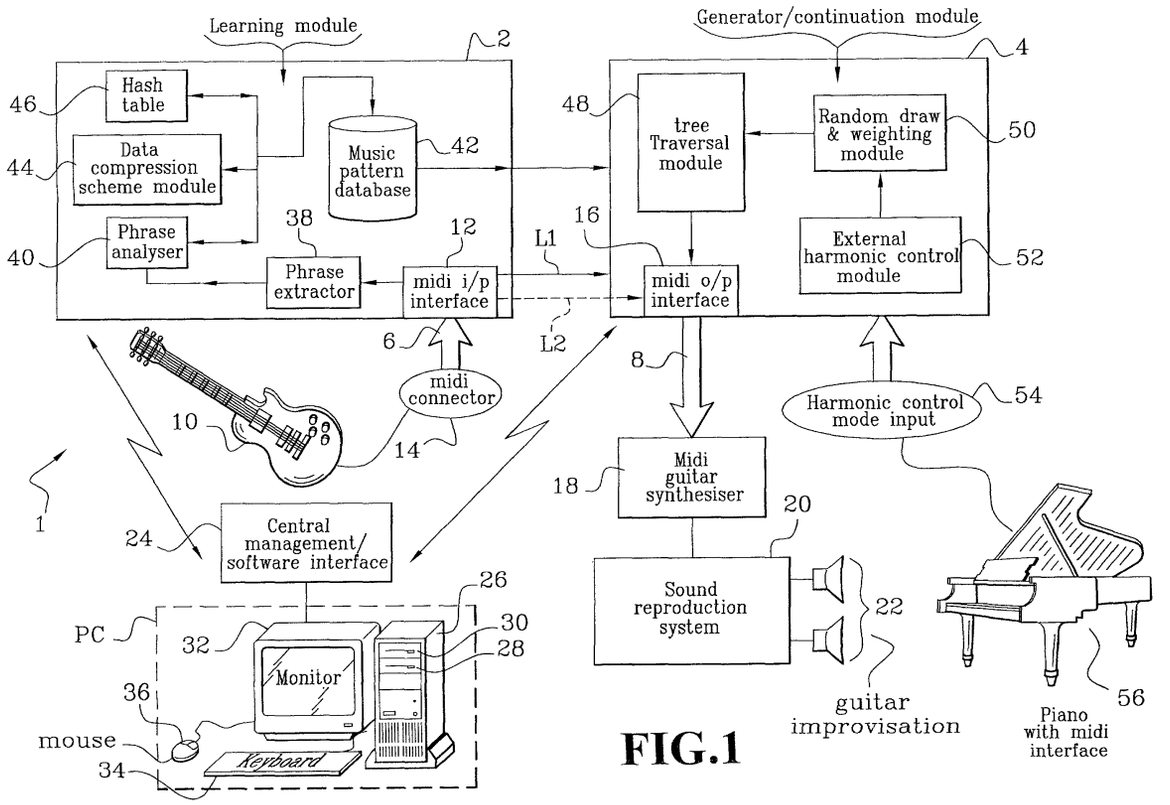


FIG.1

EP 1 265 221 A1

Description

[0001] The invention relates to a device and process for automatically improvising music such that it follows on seamlessly and in real time from music produced from an external source, e.g. a musical instrument being played live.

[0002] It can serve to simulate an improvising performing musician, capable for instance of completing a musical phrase started by a human musician, taking up instantly with an improvisation that takes into account the immediate musical context, style and other characteristics.

[0003] In this respect, the invention contrasts with prior computerised music composing systems, which can be classed into two types:

> i) systems which compose on demand autonomous musical pieces in the manner of a certain composer, performing artist or musical genre, but which do not adapt coherently to a live musical environment; and
> ii) "question-answer" type systems, i.e. in which a player inputs a music sequence and the systems replies with a complementary music sequence. The latter is influenced by the player input, but forms an independent musical item with a clear break point marking the switch from the player input (question) to the artificially generated response (answer).

[0004] Musical improvisation, especially in Jazz, is both a fascinating activity and a very frustrating one. Improvisation by a human musician requires an intimate relationship between musical thought and sensory-motor processes: the musician must listen, think, develop ideas and move his/her fingers very quickly. Speed and lack of time are crucial ingredients of improvisation; it is what makes it exciting. It is also what makes it frustrating: beginners as well as experienced improvisers are by definition limited by their technical abilities, and by the morphology of the instrument.

[0005] The invention can overcome this hurdle by creating meta instruments which address this issue explicitly: providing fast, efficient and enhanced means of generating interesting improvisation, in a real-world, real-time context.

[0006] Music improvisation has long been an object of study for computer scientists. Various systems and models of improvisation have been designed, using a wide variety of techniques. Most of the approaches have consisted in either building partial or fully-fledged improvisation systems (Biles, "Interactive GenJam; Integrating Real-Time Performance with a Genetic Algorithm, Proc. IMC 98, Ann Arbor, Michigan; Ramalho, et al, Simulating Creativity in Jazz Performance. Proc. of the National Conference in Artificial Intelligence, pp. 108-113, AAAI-94, Seattle, AAAI Press) or in proposing cognitive models of improvisation (Johnson-Laird, "Jazz Improvisation: A Theory at the Computational Level", Representing Musical Structures, P. Howell, R. West and I. Cross, eds, Academic Press, 291-325).

[0007] In all cases, no system has so far succeeded in creating fully convincing improvisations: the feeling of automation is never avoided. Moreover, most of the existing systems require a considerable amount of explicit, symbolic information to be given by a human to the system, such as human input for supervising learning, the underlying chord sequence, the tempo, the song structure, etc.

[0008] In view of the foregoing, the invention departs from these known approaches in several fundamental ways.

[0009] First, it provides for seamless integration in the playing mode of the musician, as opposed to traditional question/answer or fully automatic systems.

[0010] Second, it is based on agnostic learning, which does not require any a priori knowledge of the musician's style, harmonic grid, or tempo.

[0011] Third, it adapts quickly and without human intervention to unexpected changes in rhythm, harmony or style.

[0012] Finally, it can be made easily and intimately controllable by the musician - an extension of a musical instrument - rather than as an actual intelligent and autonomous musical agent. The resulting system achieves very good performance by basically replacing explicit, symbolic knowledge and autonomy by intimate control.

[0013] More particularly, a first object of the invention is to provide a method of automatically generating music that constitutes an improvisation, characterised in that it comprises the steps of:

- establishing a data base of music patterns,
- detecting current music data,
- generating with reference to the music patterns an improvisation as a continuation in real time of the current music data.

[0014] The music patterns can be musical phrases extracted from a stream of music data, each phrase constituting a respective music pattern.

[0015] In the preferred embodiment, the continuation is determined from potential root nodes of a tree structure expressing music patterns.

[0016] To this end, the embodiment can comprise the steps of:

- mapping in a hash table predetermined data, such as pitch, against a set of tree nodes containing the data,
- using the hash table to obtain a list of potential root nodes to start from for generating the continuation, and
- updating the hash table as a function of the creation of new nodes in the tree structure.

[0017] The tree structure can be constructed in accordance the Lempel-Ziv algorithm using a data compression scheme.

[0018] The generating step may comprise the steps of:

- extracting at least one traversal of the tree structure to obtain a set of possible continuations following on from the current music data, and
- choosing a music sequence continuation from the set by performing a random draw.

[0019] The above procedure can preferably also include the step of applying a weighting to nodes of the tree.

[0020] Advantageously, the invention can provide an improvisation control mode, comprising the steps of:

- receiving music control data, typically (although not necessarily) during the improvisation construction step, and
- selecting at least one pattern for the improvisation on the basis of the music control data.

[0021] The music control data can comprise a sequence of n notes, where n is an arbitrarily chosen number.

[0022] The continuation can be determined from potential root nodes of a tree structure expressing music patterns, wherein the selecting step comprises the step of attributing, to each of one or more nodes of the tree structure, a weight as a function of how a sequence associated to a given node matches with the control data.

[0023] The weight attributing step may comprise the steps of:

- determining a harmonic weighting function (Harmo_prob) expressed as:

$$\text{Harmo\_prob} \equiv (\text{notes}(X) \cap \text{Ctrl}) / |\text{notes}(X)|$$

  where notes(X) designates a set of pitches represented by a node X, and Ctrl designates the last n notes forming the control data, whereby:

  - if X is a note, then |X| = 1 and Harmo_prob(x) = 0 or 1, and
  - if X is a chord, then Harmo_prob(x) belongs to [0,1], and is maximal (1) when all the notes of X are in the set of external notes,

- using the harmonic weighting as at least a component of an overall weighting function (Weight(X)) to apply to nodes of the tree.

[0024] The overall weighting function, for a possible node X, can be defined as:

$$\text{Weight}(X) \equiv (1 - S)*\text{LZ\_prob}(X) + S*\text{Harmo\_prob}(X),$$

where LZ(prob(X) is the probability of X in the Tree, and S is a tuning parameter.

[0025] The method can further comprise the step of providing a jumping procedure comprising the steps of:

- determining whether, for a given input sub-sequence seq, none of the possible continuations have a non-zero Harmo_prob value,
- if none of the possible continuations have a non-zero Harmo_prob value, making a random draw weighted by S, as follows:

  If Weight(X) ≤ S, and
  If the value of the random draw is less than S, then

- effecting the jump by restarting the computation of the next node by taking the whole set of notes of the tree.

[0026] The step of establishing a data base can comprise the step of discriminating between chords and non-chordal note successions, chords being identified as notes separated from each other by a time interval shorter than a predetermined threshold.

[0027] The improvisation generating step is preferably halted upon detecting current music data. Conversely, the improvisation generating step is preferably started upon detecting an interruption in the current music data.

[0028] The music patterns forming the data base can originate from a source, e.g. music files, different from the source producing the current music data, e.g. a musical instrument.

[0029] The music patterns can also be extracted from the source producing the current music data, e.g. a musical instrument.

[0030] The music control data can be produced from a source, e.g. a musical instrument, different from the source, e.g. another musical instrument, producing the current music data.

[0031] According to a second aspect, the invention provides a device for automatically generating music that constitutes an improvisation, characterised in that it comprises:

- means for establishing a data base of music patterns,
- means for detecting current music data, and
- means for generating with reference to said music patterns an improvisation as a continuation in real time of the current music data.

[0032] The optional features presented supra in the context of the method according to the invention are ap-

plicable mutatis mutandis to the above device.

[0033] Namely, the above device may further comprise means for extracting musical phrases from a stream of music data, each phrase constituting a respective music pattern.

[0034] The improvisation generating means may comprise selection means for selecting music patterns from potential root nodes of a tree structure expressing music patterns.

[0035] Preferably, the device further comprises:

- hash table means for mapping predetermined data, such as pitch, in a hash table against a set of tree nodes containing the data,
- means for obtaining from the hash table a list of potential root nodes to start from for generating the continuation, and
- means for updating the hash table as a function of the creation of new nodes in the tree structure.

[0036] Further, the generating means may comprise:

- means for extracting at least one traversal of the tree structure to obtain a set of possible continuations following on from the current music data, and
- means for choosing a music sequence continuation from the set by performing a random draw.

[0037] The above device may further comprise means for applying a weighting to nodes of the tree.

[0038] The device may further be equipped with an improvisation control mode, comprising:

- means for receiving music control data (Ctrl), typically (although not necessarily) during the improvisation construction step, and
- means for selecting at least one pattern for the improvisation on the basis of the music control data.

[0039] The continuation may be determined from potential root nodes of a tree structure expressing music patterns, the selecting means preferably comprising means for attributing, to each of one or more nodes of the tree structure, a weight as a function of how a sequence associated to a given node matches with the control data (Ctrl).

[0040] Suitably, the device further comprises at least one of:

i) means for halting the improvisation generation upon detecting current music data; and
ii) means for starting the improvisation generating upon detecting an interruption in the current music data.

[0041] According to a third aspect, the invention provides a music improvisation system, characterised in that it comprises:

- a device as defined above,
- a first source of music data operatively connected to supply data to the data base,
- a second source of music data producing the current music data, e.g. a musical instrument.

[0042] The first source of audio data can be one of:

i) music file data, and ii) an output from a musical instrument; and the second source of audio data can be a musical instrument.

[0043] The first source and second source of audio data can equally be a same musical instrument or respective musical instruments.

[0044] According to a fourth aspect, the invention provides a music improvisation system, characterised in that it comprises:

- a device as above, equipped with an input for handling control data, and
- an improvisation control source operatively connected to the device for delivering the music control data during the improvisation.

[0045] The improvisation control source can be a musical instrument different from a musical instrument forming the second source of audio data.

[0046] According to a fifth aspect, the invention provides a system comprising:

- at least first and second devices as defined above,
- a first musical instrument and a second musical instrument different from the first musical instrument,

wherein

- the first musical instrument is operatively connected as a source of data for the data base of music patterns of the first device and as a source of current music data for the second device, whereby the second device generates an improvisation with a sound of the first musical instrument referring to a data base produced from the second instrument, and
- the second musical instrument is operatively connected as a source of data for the data base of music patterns of the second device and as a source of current music data for the first device, whereby the first device generates an improvisation with a sound of the second musical instrument referring to a data base produced from the first instrument.

[0047] According to a sixth aspect, the invention provides a software package comprising functional units for implementing the method according to the first object with a computer.

[0048] The invention and its advantages shall become more apparent from reading the following descrip-

tion of the preferred embodiments, given purely as non-limiting examples, with reference to the appended drawings in which:

- figure 1 is a general block diagram showing the functional elements of an improvisation system in accordance with a preferred embodiment of the invention,
- figure 2 is a simplified example of an LZ Tree structure used in a traversal Tree algorithmic structure within the system of figure 1,
- figure 3 is an example of different user controls offered by the embodiment through its on-screen graphic computer interface, and
- figure 4 is a diagram showing how several systems of figure 1 can be interconnected to implement a sharing mode.

[0049] A music improvisation system 1 according to a preferred embodiment of the invention is based on a combination of two modules: a learning module 2 and a generator/continuation module 4, both working in real time. The input 6 and the output 8 of the system are streams of Midi information. The system 1 is able to analyse and produce pitch, amplitude and rhythm information (onsets and duration).

[0050] The system accommodates several playing modes; it can adopt an arbitrary role and cooperate with any number of musicians.

[0051] In the standard playing mode, the system 1 is used by one musician, whose musical instrument, e.g. an electric guitar 10, has a Midi-compatible output connected to a Midi input interface 12 of the learning module 2 via a Midi connector box 14.

[0052] The output 8 of the system 1 is taken from a Midi output interface 16 to a Midi synthesiser 18 and then to a sound reproduction system 20. The latter plays through loudspeakers 22 either the audio output of the system 1 or the direct output from the instrument 10, depending whether the system or the instrument is playing.

[0053] The learning module 2 and the generator/continuation module 4 are under the overall control of a central management and software interface unit 24 for the system. This unit is functionally integrated with a personal computer (PC) comprising a main processing unit (base station) 26 equipped with a mother board, memory, support boards, CDrom and/or DVDrom drive 28, a diskette drive 30, as well as a hard disk, drivers and interfaces. The software interface 24 is user accessible via the PC's monitor 32, keyboard 34 and mouse 36. Optionally, further control inputs to the system 1 can be accessed from pedal switches and control buttons on the Midi connector box 14, or Midi gloves.

[0054] In the standard situation, the system acts as a "sequence continuator": the note stream of the musician's instrument 10 is systematically segmented into phrases by a phrase extractor 38, using a temporal threshold (200 milliseconds). Each phrase resulting from that segmentation is sent asynchronously from the phrase extractor 38 to a phrase analyser 40, which builds up a model of recurring patterns. In reaction to the played phrase, the system also generates a new phrase, which is built as a continuation of the input phrase, and not as an "answer" as in the case of the Biles reference supra.

[0055] To build the continuation, the learning module 2 systematically learns all melodic phrases played by the musician. The technique consists in building progressively a database of recurring patterns 42 detected in the input sequences produced by the phrase analyser 40. For this purpose, the learning module 2 uses a data compression scheme (implemented by module 44) adapted from the scheme described by Lempel-Ziv in the paper "Compression of individual sequences via a variable rate coding, IEEE Trans. Int. Computer Music Conf. (ICMC'91), pp.344-347. This scheme, referred to hereafter as the LZ scheme, has been shown to be well suited for capturing melodic patterns efficiently, as explained by Assayag & al. in the paper "Guessing the composer's mind: applying universal prediction to musical style, Proc. ICMC 99, Beijing, China, I.C.M.A., San Francisco, USA.

[0056] The technique consists in building a prefix Tree by a simple, linear analysis of each input sequence. Each time a sequence is input to the system 1, it is parsed from left to right and new prefixes encountered are systematically added to the Tree. The principle of the LZ Tree is described in detail in the Assayag et al paper.

[0057] The LZ scheme is itself well known in data compression, the algorithm itself dating back from the 80's. The idea of applying LZ to musical modelling has been proposed by various authors (Assayag, Dubnov, Delerue and others), but only in the context of off-line musical composition, or musical style classification.

[0058] By contrast, the present invention adapts this scheme to the context of real-time (*real tim*) music improvisation. To this end, the embodiment provides an efficient data structure based on 1) a standard Lempel Ziv Tree structure and 2) a hash table (dictionary) 46 that allows to directly access the nodes in the structures having a given data. This serves to ensure that learning can be done in real time (less than a few milliseconds). There is recorded in the Tree only information related to pitch and velocity (Midi). Rhythmic information is discarded in the present embodiment, although it can also be accommodated if required.

[0059] Learning can be done on the fly with the user playing from scratch. It can also be done from a library of predefined musical styles (for instance in the style of well known Jazz artists such as Pat Martino, John McLaughlin, John Coltrane, Charlie Parker, etc.).

[0060] Note that it the embodiment, the handling of the LZ algorithm is modified in several ways, covered in sections 1 to 3 infra.

1. Implementing an efficient Tree structure.

[0061] Because most usage of LZ for music is for batch processing (e.g. composition or classification), the present application calls for an efficient representation of the LZ Tree, both for Tree updating (learning) and traversal (generation). The Assayag et al. paper proposes a dual representation of the LZ Tree which is supposed to speed up the traversal, but which takes up much more space, by basically doubling the size of the Tree.

[0062] Very good results have been obtained by exploiting the basic Tree structure, augmented with a Hash table 46 mapping each data (here Midi pitch) to the set of Tree nodes containing this data. The Hash table 46 is updated each time a new node is created. When a Tree traversal must be effected from an input sequence starting with item *i*, the Hash table directly yields the list of potential root nodes to start from.

[0063] The generator/continuation module 4 of the system 1 is the real time continuation mechanism, which generates the music in reaction to an input sequence. The generation is performed using a traversal of Tree, through a traversal Tree module 48, to obtain the set of all possible continuations of the sequence. Then, the following item is chosen by a random draw, weighted by the probabilities of each possible continuation. This function is provided by a random draw and weighting module 50.

[0064] Music input to the generator/continuation module 4 is taken from the real time Midi source input interface 12 via an internal connection L1. In this way, the module 4 receives, at least, pitch and velocity information from the instrument 10.

[0065] In operation, the system 1 automatically detects musical phrases, by the means of a given time threshold applied to the phrase extractor 38. When a musical phrase is detected, it is sent to the phrase analyser 40 of the learning module 2 AND, through internal connection L1, to the generator/continuation module 4. The latter computes a possible continuation for the input phrase, of a given length (parameter), and outputs it to the sound reproduction system 20 (e.g. through a Midi scheduler). Usually, prefixes are only looked up from the root node. In order to ensure completeness, the embodiment introduces a modification in the access method of the Tree. Since a given prefix can be located arbitrarily in the Tree structure, a check is made for all its possible occurrences in any part of the Tree. The selection of the "next" node to take is based on the reunion of all possible continuations thus found, associated with the corresponding weights.

[0066] The Tree traversal is extended to ensure a complete search. By definition, an LZ Tree usually contains only partial information about patterns present in the input sequence, and is not complete (as opposed to classical Markov models). Consequently, a given pattern can be present at several locations in the Tree. For instance, in the Tree shown in figure 2, the pattern ABC is present twice (at nodes with an asterix):

[0067] To detect these two occurrences, the usual Tree traversal mechanism is augmented slightly by computing all possible continuations for a given input sequence. In the present case, with an input sequence ABC, the hash table 46 yields three nodes having data "A", so three "standard" traversals are performed, to get eventually only two sets of possible continuations (here, D, D and E). These continuations are aggregated and the next item is drawn according to the respective weights of each node. The system can handle over 35 000 LZ Tree nodes in real time (i.e. with a response time of less than 200 milliseconds), with a Java implementation on a personal computer PC running with a Pentium III microprocessor using a "MidiShare" (Pentium III is a registered trademark of Intel. Inc.).

[0068] The preferred embodiment makes use of the LZ (Lempel-Ziv) algorithm in its capacity to build a Tree of recurring patterns in an efficient way, but not its compressive capacity, as in usual implementations. More particularly, there is provided a more efficient use of this method, so that a real-time learning can be performed.

[0069] In data compression, the above-mentioned Tree is built and eventually used to build a compressed representation of a given sequence. The embodiment, however, does not utilise this compression, but merely uses the Tree of patterns for generating other sequences "in the style of" the previously inputted sequences.

[0070] The real-time learning can be performed by any method, as long as it can be done quickly (as in the case of Lempel-Ziv algorithm, which involves only one traversal of the input sequence), and as long as it can quickly produce a structure useable to complete the input sequence.

[0071] It is recalled that Lempel-Ziv parsing consists in building a Tree starting from the beginning of the sequence (root is on the left) to the right, and building each time a prefix Tree.

[0072] For instance, the sequence A B C A B C D A B C C D E produces a Tree as follows, where each prefix is indicated successively between commas (,):

A, B, C, AB, CD, ABC, CDE
A-B-C
B
C-D-E

[0073] The Tree does not contain all the patterns in the sequence. However, it does converge, for infinite sequences, to the entropy of the sequence, i.e. it has good properties in the long term.

[0074] Now, the Tree can be used for generating a sequence by choosing each time a node among possible continuations, according to its "weight", cf. unit 50. The weight is in fact the size of its sub-Tree, and corresponds, by construction, to its probability of occurrence.

2. Handling of chords.

[0075] The embodiment introduces a representation of chords so that the LZ Tree can handle single notes or chords indifferently. Chords are abstract entities: a Midi stream is by definition a stream of single notes, so chords as such do not exist in reality for a Midi stream. What shall be termed a chord in the present context is any stream of n notes whose tonal duration is less than a given threshold, e.g. 40 milliseconds. When such a stream is detected, a single chord object aggregating all the notes in the stream is created, instead of n single note objects.

[0076] To allow comparison of this chord object with other chord objects, each chord has a canonical representation that takes into account basic facts about tonal harmony:

- pitches are considered only modulo 12 to allow octave equivalence,
- pitches are sorted to consider streams of similar pitches with different ordering as similar.

3. Implicit management of harmony.

[0077] Harmony is a fundamental notion in most forms of music, Jazz being a particularly good example in this respect. Chord changes play an important role in deciding whether notes are "right" or not. It is important to note that while harmony detection is extremely simple to perform for a normally trained musician, it is extremely difficult for a system to express and represent explicitly harmony information, especially in real time. Accordingly, a design choice of the system according to the present embodiment is not to manage harmony explicitly. This choice is justified on three considerations :

i) because of the intimate control of the system, it is easy for the musician to correct the system in case it really goes too far out of tune, by simply playing a few notes (e.g. the third and fifth) and relaunching the system 1 in a new, correct, direction. To this end, the generator/continuation module 4 has a module 52 for external harmonic control, which cooperates with the random draw/weighting module 50. The external harmonic control module 52 presents an external input connector 54 for receiving the above mentioned notes. These "steering notes" effectively redirect the improvisation towards a chosen direction;

ii) because the system continuously learns, it eventually also learns the chord changes in the pattern base. For instance, playing tunes such as "So What" by Miles Davis (alternation of D minor and D# minor) creates in the long run patterns with this chord change;

iii) finally, and perhaps most importantly, the embodiment is designed to have a control mode through module 52 supra that actually allows the system to take into account external harmonic information without unduly complicating the data representation scheme.

[0078] The idea is to introduce a constraint facility in the generation phase. External information may be sent as additional input to the system via the harmonic mode control input 54. This information can be typically the last n notes (pitches) played by any external source (e. g. a piano with Midi interface 56) in a piano-guitar ensemble, where the guitar 10 is connected to Midi input interface 12, for instance. The value of n can be set arbitrarily to 8, to provide steering on the basis of the last eight notes. External input 54 is thus used to influence the generation process as follows. When a set of possible continuation nodes is computed with the LZ Tree, as described above, instead of choosing a node according to its weight (probability), the random draw and weighting module 50 is set to weight the nodes according to how they match the notes presented at the external input 54. For instance, it can be decided to give preference to nodes whose pitch is included in the set of external pitches, to favour branches of the Tree having common notes with the piano comping (?). In this case, the harmonic information is provided in real time by one of the musicians (in this case the pianist), without intervention of the user, and without having to explicitly enter the harmonic grid in the system. The system then effectively matches its improvisation to the thus-entered steering notes.

[0079] This matching is achieved by a harmonic weighting function designated "Harmo_prob" and defined as follows.

[0080] Consider a set of external notes, designated Ctrl, entered into the harmonic control module 52 through input 54. These notes Ctrl are taken to correspond to the last n notes entered at input 54, coming e. g. from a piano 56, while Midi input interface 12 is connected to a guitar 10 and the synthesiser 18 that is connected to the Midi output interface 16 is a guitar synthesiser.

[0081] Consider now the set of pitches represented by node X, designated notes(X). The harmonic weighting function for notes(X) can then be expressed as:

$$\text{Harmo\_prob} \equiv (\text{notes}(X) \cap \text{Ctrl}) / |\text{notes}(X)|$$

[0082] If X is a note (and not a chord), then |X| = 1 and Harmo_prob(x) = 0 or 1.

[0083] If X is a chord, then Harmo_prob(x) belongs to [0,1], and is maximal (1) when all the notes of X are in the set of external notes.

[0084] There is then defined a new function for choosing the next node in the LZ Tree. Consider 1) LZ_prob (X), the probability of X in the LZ Tree, and 2) Harmo_prob (X), the harmonic weighting function,

which assigns a weight to node X in the Tree, representing how close the node matches an external input. Both LZ_prob and Harmo_prob assign values in [0,1]. The aim is to achieve a compromise between these two weighting schemes. To introduce some flexibility, the system 1 adds a parameter S that allows tuning the total weighting scheme, so that the weight can take on a range of intermediate values between two extremes. When S = 0, the weighting scheme is equal to the standard probability-based weighting scheme. When S = 1, the weighting scheme is equivalent to the harmonic function.

[0085] The weight function is therefore defined as follows, where X is a possible node:

$$\text{Weight}(X) \equiv (1 - S)*\text{LZ\_prob}(X) + S*\text{Harmo\_prob}(X).$$

[0086] Finally, the system 1 introduces a "jumping procedure", which allows to avoid a drawback of the general approach. Indeed, it may be the case that for a given input sub-sequence seq, none of the possible continuations have a non-zero Harmo_prob value. In such a case, the system 1 introduces the possibility to "jump" back to the root of the LZ Tree, to allow the generated sequence to be closer to the external input. Of course, this jump should not be made too often, because the stylistic consistency represented by the LZ Tree would otherwise be broken. The system 1 therefore performs this jump by making a random draw weighted by S, as follows:

If Weight(X) ≤ S, and
If the value of the random draw is less than S

[0087] Then make a jump, that is restart the computation of the next node by taking the whole set of notes of the LZ Tree, rather than the natural continuation of seq.

[0088] The system 1 does not learn nor produce rhythmic information. In the context of Jazz improvisation, for instance, it was found that letting the continuation system produce rhythm by combining patterns is very awkward, and in fact limits its usability. Instead, the preferred embodiment generates phrases of eight notes. Velocity is enough to produce phrases that sound human to the point of being indistinguishable. The system 1 can actually also generate non-linear streams of notes by simply using the rhythmic pattern of the input sequence, and mapping it to the output sequence. In any case, no rhythmic information is learned.

[0089] In the embodiment, Midi files for improvisation are used as input sequences. To this purpose, there can be used known improvisers, which are long enough to let the system actually capture the recurring patterns.

[0090] Further, the learning scheme can be carried out several times to accelerate the learning process.

[0091] The embodiment described is limited, intentionally, in its musical knowledge. The limitations are of three kinds: 1) rhythmic, 2) harmonic, and 3) polyphonic. Interestingly, it has turned out that these limitations are actually strengths, because of the control mode. In some way, these limitations are compensated by control.

[0092] The system leads to several control issues. First, in view of providing an intimate control by the musician, the embodiment implements a set of basic controllers that are easy to trigger in real time. These are accessible through the software interface 24 in the form of on-screen pushbuttons and pull-down menus on the PC monitor 32, which can be activated through the computer keyboard 34 and/or mouse 36.

[0093] An example of a typical screen page graphic of the computer interface displayed on the monitor 32 is shown in figure 3.

[0094] Among the different controllable parameters are the following basic controls:

- "learn on/off" (tick box 60), to set the learning process on or off
- "continuation on/off" (tick box 62) to tell the system to produce continuations of input sequences or not, and
- "superposition on/off" (tick box 64), to tell the system whether it should stop its generation when a new phrase is detected, or not.

[0095] The last control is particularly useful. By default, the system stops playing when the user starts to play or resumes, to avoid superposition of improvisations. With a little bit of training, this mode can be used to produce a unified stream of notes, thereby producing an impression of seamlessness. In other words, the system 1 takes over with its improvisation immediately from the point where the musician (guitar 10) stops playing, and ceases instantly when the musician starts to play again. These controls are implemented with a foot controller of the Midi connector box 14 when enabled by the basic controls on screen (tick boxes). They can also be implemented with "Midi" gloves.

[0096] When the system 1 is silent owing to the presence of music output from the instrument 10, it continues to analyse that output as part of its continuing learning process, as explained above. An internal link L2 is active in this case to also send the music output of the instrument from the Midi input interface 12 to the Midi output interface 16, so as to allow the instrument to be heard through the Midi synthesiser 18, sound reproduction system 20 and speakers 22.

[0097] Additionally the software interface allows a set of parameters to be adjusted from the screen 32, such as:

- the number of notes to be generated by the system (as a multiplicative factor of the number of notes in the input sequence) (box 66), and

- the tempo of the generated sequence (as a multiplicative factor of the tempo of the incoming sequence) (box 68).

[0098] An interesting consequence of the design of the system 1 is that it leads to several remarkable new playing modes, including:

- Single autarcy. One musician plays with the system 1 after having fed the system with a database of improvisations by a famous musician, as Midi files. Successful results have been achieved in this way with a database of Midi choruses from Pat Martino;
- Multiple autarcy: each musician has his/her own system 1, with its own datatbase;
- Master/Slave: one musician (e.g. guitarist) uses the system in its basic form, another (e.g. pianist) provides the external data (steering notes) at the harmonic control mode input 54 to influence the generation of the music improvisation;
- Cumulative: all musicians share the same pattern database.
- Sharing: each musician plays with the pattern database of the other (e.g.; piano with guitar, etc.). This creates exciting new possibilities as a musician can experience playing with unusual patterns.

[0099] Figure 4 shows an example of a set-up for the sharing mode in the case of a guitar and piano duo (of course, other instruments outside this sharing mode can be present in the music ensemble). Here, each instrument in the sharing mode is non acoustic and composed a two functional parts : the played portion and a respective synthesiser. For the guitar, these portions are respectively the main guitar body 10 with its Midi output and a guitar synthesiser 18b. For the piano, they are respectively the main keyboard unit with its Midi output 56 and a piano synthesiser 18a.

[0100] Two improvisation systems 1a and 1b as described above are used. The elements shown in figure 4 in connection with these systems are designated with the same reference numerals as in figure 1, followed by an "a" or "b" depending on whether they depend from improvisation system 1a or 1b respectively.

[0101] One of the improvisation systems 1a has its Midi input interface 12a connected to the Midi output of the main guitar body 10 and its Midi output interface 16a connected to the input of the piano synthesiser 18a. The latter thus plays the improvisation of system 1a, through the sound reproduction system 20a and speakers 22a, based on the phrases taken from the guitar input.

[0102] The other improvisation system 1b has its Midi input interface 12b connected to the Midi output of the main keyboard unit 56 and its Midi output interface 16b connected to the Midi input of the guitar synthesiser 18b. The latter thus plays the improvisation of system 1b, through the sound reproduction system 20b and speakers 22b, based on the phrases taken from the piano input.

[0103] This inversion of synthesisers 18a and 18b is operative all while the improvisation is active. When a musician starts playing, the improvisation is automatically interrupted so that his/her instrument 10 or 56 takes over through its normally attributed synthesiser 18b or 18a respectively. This taking over is accomplished by adapting link L2 mentioned supra so that a first link L2a is established between Midi input interface 12a and Midi output interface 16b when the guitar 10 starts to play, and a second link L2b is established between Midi interface 12b and Midi output interface 16a when the piano 56 starts playing.

[0104] Naturally, this concept of connecting the inputs 6 and outputs 8 of the system to different instruments can extrapolated to any number n of improvisation systems, the choice of instruments involved being arbitrary.

[0105] From the foregoing, it is apparent that the embodiment allows to generate automatically musical melodies in a given style. It automatically learns the style from various sorts of musical inputs (real time, Midi files). The invention allows to generate musical melodies in a reactive way, according to musical stimulus, coming, e.g. from real time Midi signals. Several modes of control are proposed, to make the invention actually useable in a real live performance context.

[0106] The invention can be embodied in wide variety of forms with a large range of optional features. The implementation described is based largely on existing hardware elements (computer, Midi interfaces, etc.), with the main aspects contained in software based modules. These can be integrated in a complete or partial software package in the form of a suitable data carrier, such as DVD or CD disks, or diskettes that can be loaded through the appropriate drives 28, 30 of the PC.

[0107] Alternatively, the invention can be implemented as a complete stand-alone unit integrating all the necessary hardware and software to implement a complete system connectable to one or several instruments and having its own audio outputs, interfaces, controls etc.

[0108] Between these two extremes, a large number of software, firmware and hardware embodiments can be envisaged.

[0109] Finally, it is clear that music data protocols other than Midi can be envisaged. Likewise, the teachings of the invention accommodate for all sorts of music styles, categories, and all sorts of musical instruments, those mentioned with reference to the figures being mere examples.

Claims

1. A method of automatically generating music that constitutes an improvisation, **characterised in that** it comprises the steps of:

   - establishing a data base of music patterns (42),

- detecting current music data (12),
- generating with reference to said music patterns an improvisation as a continuation in real time of said current music data (4).

2. Method according to claim 1, further comprising the step of extracting musical phrases from a stream of music data (38, 40), each phrase constituting a respective music pattern.

3. Method according to claim 1 or 2, wherein said continuation is determined from potential root nodes of a Tree structure expressing music patterns.

4. Method according to claims 3, further comprising the steps of:

- mapping in a hash table (46) predetermined data, such as pitch, against a set of tree nodes containing said data,
- using said hash table to obtain a list of potential root nodes to start from for generating said continuation, and
- updating said hash table as a function of the creation of new nodes in said tree structure.

5. Method according to claim 3 or 4, wherein said tree structure is constructed in accordance the Lempel-Ziv algorithm using a data compression scheme (44).

6. Method according to any one of claims 3 to 5, wherein said generating step comprises the steps of:

- extracting at least one traversal of said tree structure (48) to obtain a set of possible continuations following on from said current music data, and
- choosing a music sequence continuation from said set by performing a random draw (50).

7. Method according to claim 6, further comprising the step of applying a weighting (52) to nodes of said tree.

8. Method according to any one of claims 1 to 7, further comprising the provision of an improvisation control mode (54, 52), comprising the steps of:

- receiving music control data during said improvisation construction step, and
- selecting at least one pattern for said improvisation on the basis of said music control data (Ctrl).

9. Method according to claim 8, wherein said music control data (Ctrl) comprise a sequence of n notes, where n is an arbitrarily chosen number.

10. Method according to any one of claims 8 or 9, with said continuation determined from potential root nodes of a Tree structure expressing music patterns, wherein said selecting step comprises the step (50) of attributing, to each of one or more nodes of said tree structure, a weight as a function of how a sequence associated to a given node matches with said control data (Ctrl).

11. Method according to claim 10, wherein said weight attributing step comprises the steps of:

- determining a harmonic weighting function (Harmo_prob) expressed as:

$$Harmo\_prob \equiv (notes(X) \cap Ctrl) / |notes(X)|$$

where notes(X) designates a set of pitches represented by a node X, and Ctrl designates the last n notes forming said control data, whereby:

- if X is a note, then |X| = 1 and Harmo_prob (x) = 0 or 1, and
- if X is a chord, then Harmo_prob(x) belongs to [0,1], and is maximal (1) when all the notes of X are in the set of external notes,

- using said harmonic weighting as at least a component of an overall weighting function (Weight(X)) to apply to nodes of said Tree.

12. Method according to claim 11, wherein said overall weighting function, for a possible node X, is defined as:

$$Weight(X) \equiv (1 - S)*LZ\_prob(X) + S*Harmo\_prob(X),$$

where LZ(prob(X) is the probability of X in said Tree, and S is a tuning parameter.

13. Method according to claim 11 or 12, further comprising the step of providing a jumping procedure comprising the steps of :

- determining whether, for a given input sub-sequence seq, none of the possible continuations have a non-zero Harmo_prob value,
- if none of the possible continuations have a non-zero Harmo_prob value, making a random draw weighted by S, as follows:

If Weight(X) ≤ S, and
If the value of the random draw is less than

S, then

- effecting said jump by restarting the computation of the next node by taking the whole set of notes of said Tree.

14. Method according to any one of claims 1 to 13, wherein said step of establishing a data base comprises the step of discriminating between chords and non-chordal note successions, chords being identified as notes separated from each other by a time interval shorter than a predetermined threshold.

15. Method according to any one of claims 1 to 14, wherein said improvisation generating step is halted upon detecting current music data.

16. Method according to any one of claims 1 to 15, wherein said improvisation generating step is started upon detecting an interruption in said current music data.

17. Method according to any one of claims 1 to 16, wherein said music patterns forming said data base originate from a source, e.g. music files, different from the source producing said current music data, e.g. a musical instrument (10).

18. Method according to any one of claims 1 to 17, wherein said music patterns are extracted from the source producing said current music data, e.g. a musical instrument.

19. Method according to any one of claims 8 to 18, wherein said music control data is produced from a source, e.g. a musical instrument (56), different from the source, e.g. another musical instrument, producing said current music data.

20. A Device for automatically generating music that constitutes an improvisation, **characterised in that** it comprises:

- means (2) for establishing a data base of music patterns,
- means (12)) for detecting current music data, and
- means (4) for generating with reference to said music patterns an improvisation as a continuation in real time of said current music data.

21. Device according to claim 20, further equipped with an improvisation control mode, comprising:

- means (54) for receiving music control data (Ctrl) during said improvisation construction step, and
- means (50) for selecting at least one pattern for said improvisation on the basis of said music control data.

22. A music improvisation system, **characterised in that** it comprises:

- a device according to claim 20 or 21,
- a first source of music data operatively connected to supply data to said data base, and
- a second source of music data (10) producing said current music data, e.g. a musical instrument.

23. System according to claim 22, wherein said first source of audio data is one of:

i) music file data, and ii) an output from a musical instrument (10); and wherein said second source of audio data is a musical instrument (10; 56).

24. System according to claim 23, wherein said first source and second source of audio data are a same musical instrument (10).

25. System according to claim 23, wherein said first source and said second source are respective musical instruments (10 and 56).

26. A music improvisation system, **characterised in that** it comprises:

- a device according to claim 21,
- an improvisation control source (56) operatively connected to said device for delivering said music control data (Ctrl) during said improvisation.

27. System according to claim 26, wherein said improvisation control source is a musical instrument (56) different from a musical instrument (10) forming said second source of audio data.

28. A system comprising:

- at least first and second devices (1a, 1b according to claim 20 or 21,
- a first musical instrument (10) and a second musical instrument (56) different from said first musical instrument,

wherein

- said first musical instrument is operatively connected as a source of data for said data base of music patterns of said first device and as a source of current music data for said second

device, whereby said second device generates an improvisation with a sound of said first musical instrument referring to a data base produced from said second instrument, and

- said second musical instrument is operatively connected as a source of data for said data base of music patterns of said second device and as a source of current music data for said first device, whereby said first device generates an improvisation with a sound of said second musical instrument referring to a data base produced from said first instrument.

29. A software package comprising functional units for implementing the method according to any one of claims 1 to 19 with a computer.

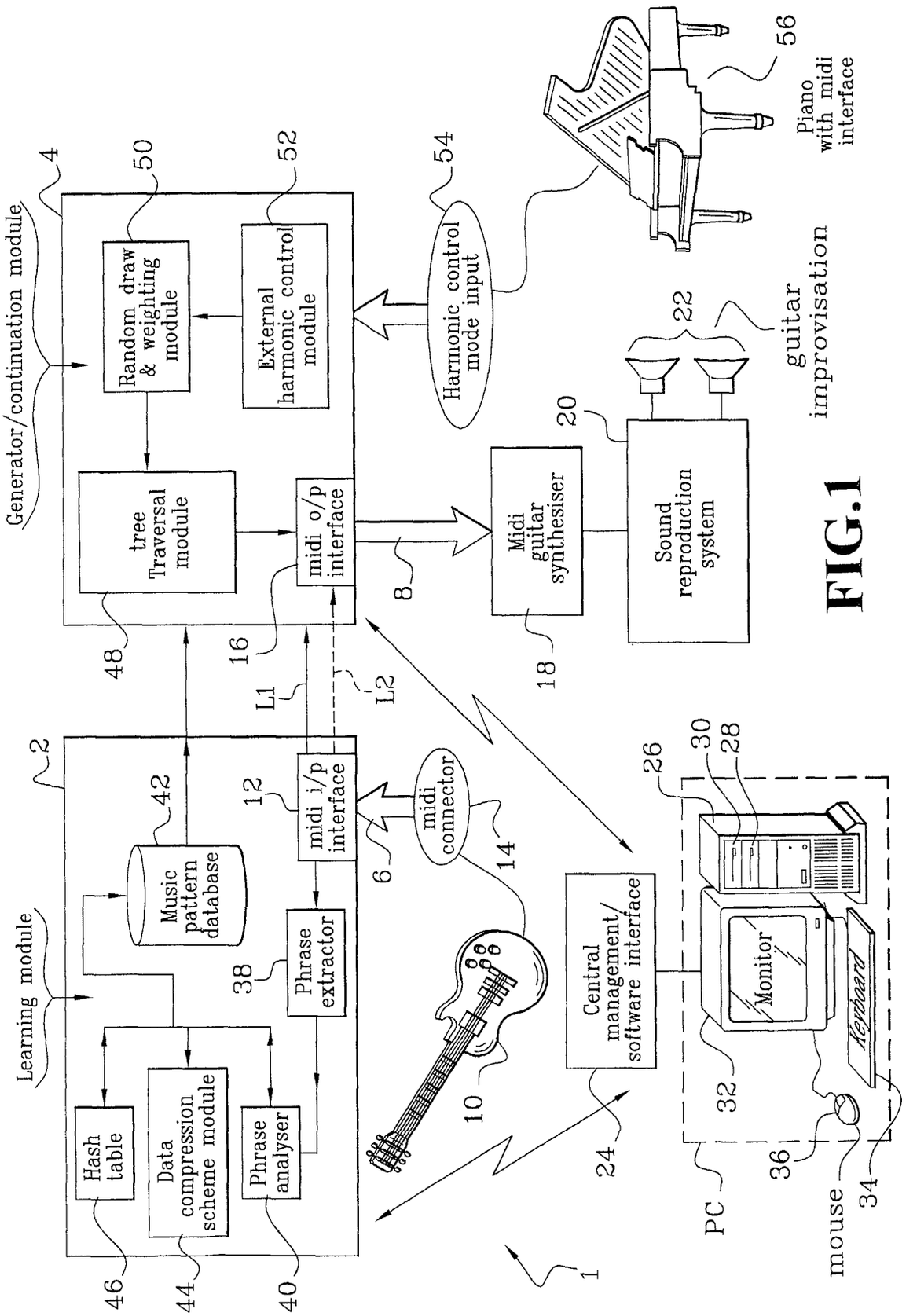

Generator/continuation module
4
50
Random draw & weighting module
52
External harmonic control module
54
Harmonic control mode input
56
Piano with midi interface
tree Traversal module
midi o/p interface
48
16
L1
L2
8
Midi guitar synthesiser
20
Sound reproduction system
22
guitar improvisation
18
2
Learning module
42
Music pattern database
12
midi i/p interface
midi connector
6
14
38
Phrase extractor
Hash table
Data compression scheme module
Phrase analyser
46
44
40
10
Central management/ software interface
24
26
30
28
Monitor
Keyboard
32
PC
36
mouse
34
1

FIG.1

FIG.2

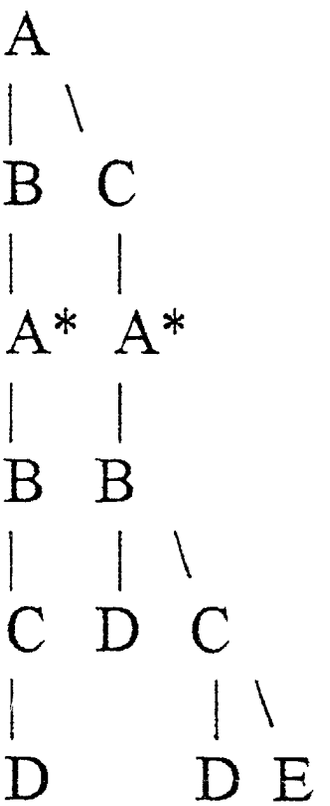
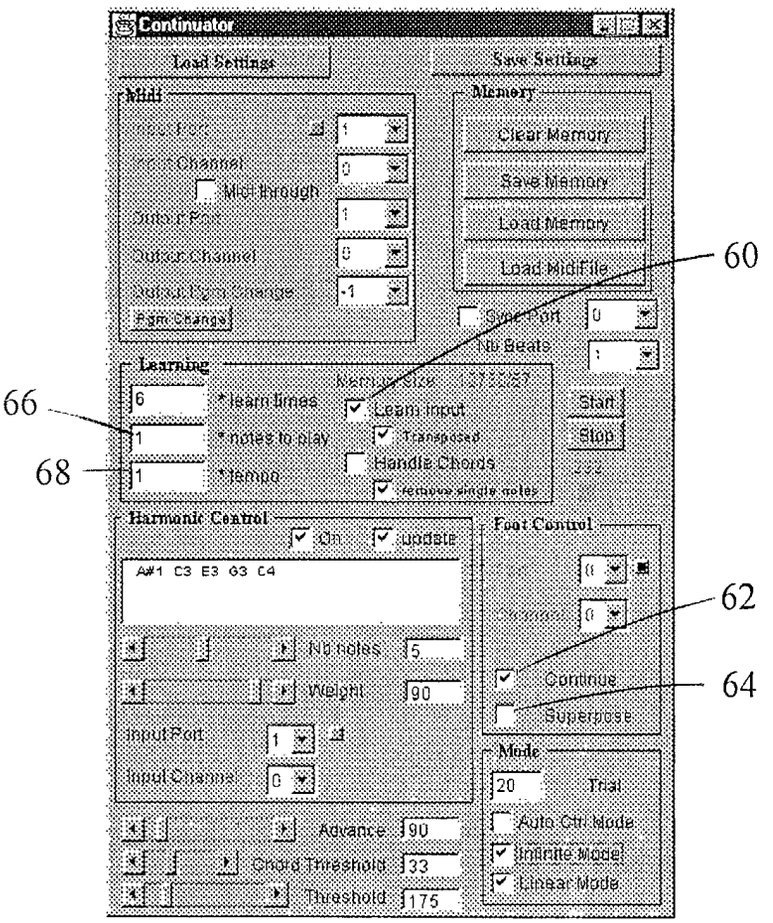


FIG.3


Continuator
Load Settings
Save Settings
Midi
Memory
Clear Memory
Save Memory
Load Memory
Load MidiFile
Midi Through
Learning
learn times
notes to play
tempo
Learn Input
Transposed
Handle Chords
Start
Stop
Harmonic Control
update
A#1 C3 E3 G3 C4
Nb notes
Weight
Input Port
Input Channel
Advance
Chord Threshold
Threshold
Foot Control
Continue
Superpose
Mode
Trial
Auto Ctrl Mode
Infinite Mode
Linear Mode
60
62
64
66
68

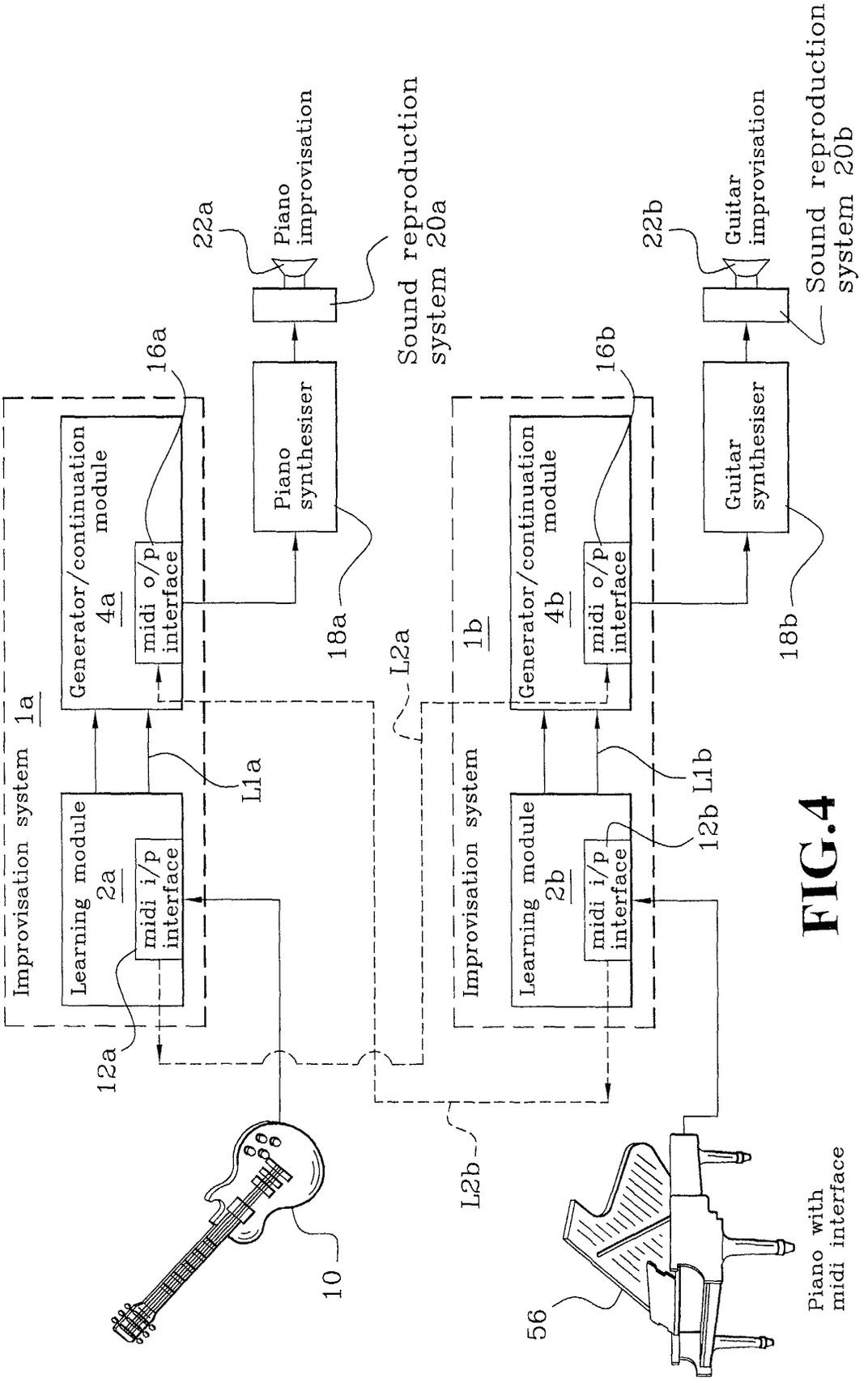

Improvisation system 1a
Learning module 2a
midi i/p interface
Generator/continuation module 4a
midi o/p interface
Piano synthesiser
Piano improvisation
Sound reproduction system 20a
Improvisation system 1b
Learning module 2b
midi i/p interface
Generator/continuation module 4b
midi o/p interface
Guitar synthesiser
Guitar improvisation
Sound reproduction system 20b
L1a
L2a
L1b
L2b
12a
12b
16a
16b
18a
18b
22a
22b
10
56
Piano with midi interface

FIG.4

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 40 1485

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	WO 99 46758 A (ADRIAANS ADZA BEHEER B V ;ADRIAANS PIETER WILLEM (NL)) 16 September 1999 (1999-09-16) * page 4, line 2 - page 5, line 8 * * page 9, line 5 - line 19 * * page 13, line 3 - line 27 * * page 14, line 33 - page 16, line 31 *	1-4,6,8, 17-20	G10H1/00
A		21-29	
X	US 5 736 666 A (GOODMAN RODNEY M ET AL) 7 April 1998 (1998-04-07) * column 5, line 1 - line 46 * * column 8, line 54 - column 10, line 36 * * column 11, line 26 - column 13, line 29 * * column 16, line 66 - column 17, line 51; figures 1-3 *	1-4, 8-10,14, 17-21,29	
A	WO 01 09874 A (MESTER SANDOR JR) 8 February 2001 (2001-02-08) * page 3, line 3 - line 28 * * page 5, line 16 - page 6, line 11 * * page 11, line 8 - line 30; figures 1,8 *	1,2,20	TECHNICAL FIELDS SEARCHED (Int.Cl.7) G10H
A	US 5 990 407 A (GANNON PETER) 23 November 1999 (1999-11-23) * column 1, line 38 - column 2, line 16 * * column 2, line 64 - column 3, line 10; figure 1 *	1-29	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	23 November 2001	Pulluard, R

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 01 40 1485

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

23-11-2001

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9946758 A	16-09-1999	NL 1008586 C1	14-09-1999
		AU 2962899 A	27-09-1999
		EP 1062656 A1	27-12-2000
		WO 9946758 A1	16-09-1999
		US 6313390 B1	06-11-2001
US 5736666 A	07-04-1998	AU 2341097 A	10-10-1997
		JP 2000507001 T	06-06-2000
		US 5883326 A	16-03-1999
		WO 9735299 A1	25-09-1997
WO 0109874 A	08-02-2001	HU 9902621 A2	28-09-2001
		AU 6308700 A	19-02-2001
		WO 0109874 A1	08-02-2001
US 5990407 A	23-11-1999	AU 731747 B2	05-04-2001
		AU 3431797 A	09-02-1998
		CA 2259369 A1	22-01-1998
		WO 9802867 A1	22-01-1998
		EP 0978117 A1	09-02-2000
		JP 2000514571 T	31-10-2000

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82