The best next note

Work in progress on audio-interactive computer music

Version 1.0

Teun de Lange

20-11-2013

Table of contents

0 Introduction

The speed and accuracy of modern computers could - to my opinion - be used to create really interactive music in which live audio input triggers a meaningful musical response of a computer, thus creating an inspiring environment for free improvisation and intuitive musical experiments. The *Jazzperiments Jam* application - which I developed over the years - has most modules and ingredients to achieve this. Experiments show it works in 'experimental' contexts. It is time now to improve and optimize the system for 'common' use ... and if this 'works' ... to create a completely new musical experience.

As there are many different settings and processes which could be adapted, 'tuned' and tested, a trial and error approach is a waste of time and doesn't offer any guarantee that the best solution is found. A proper strategy based on musicological and technical theory is needed. The objective of this paper is to propose this. Hopefully it also leads to a fruitful exchange of ideas among likeminded experimental music enthusiasts.

The first chapter describes developments up to now and readers who know the *Jazzpertimens Jam* project can skip it.
The second chapter describes the theory behind the plans and readers who just want to know 'what's up' can skip this too.
Requests for the opinion of the reader are printed in blue.
Thanks for your opinion, remarks and criticism on black and blue parts of this paper.

1 Interactive music

1.1 Concept and existing examples

As an art form music is interactive by nature. Like dance and theatre, almost always more than one artist/musician is involved in a performance of a piece, requiring interaction between all participants. To perform a piece they have to act according to a scenario/score and react to actions of others simultaneously.

Electronics and computers have been used in music over the past century. However this technology was almost exclusively aimed at creating new instruments or adapting the sound of existing ones. The musicians always stayed in control. Up to now, music in which an electronic instrument or computer acts as a - more or less - autonomous 'performer' is quite rare and highly experimental. Even in those cases the interaction with the computer is often triggered by electronic signals from motion controllers or gestures picked up by cameras, not by the music itself, although 'audio-interaction' is the way music normally works.

There are exceptions:
Almost 30 years ago trombonist George Lewis (not the clarinetist from New Orleans) built his *Voyager* program which was used in many experimental jazz concerts, creating unique examples of sound based interactive music.
More recently Robert Rowe built his *Cypher* system in Max MSP to create beautiful interactive music, like for example his composition *Cigar Smoke*.

I've spent quite some time to find more composers/musicians and their pieces but didn't find any other examples of audio-interactive music. Please inform me, if you know other pieces.

1.2 *Jazzperiments Jam*

Jazzperiments Jam is an interactive music application. Its main objective is to generate an interesting musical response on audio input and to offer an inspiring environment for improvisation and intuitive musical experiments.
The system recognizes notes in the input sound using advanced Fourier analysis and plays back notes which are transposed to notes which belong to a matching scale or chord.
The program was originally developed for interactive (jazz) music, but was also used as the 'engine' for several interactive sound art projects. The applications range from an automatic bass accompaniment for a singer to an interactive sound installation in a willow forest in an open air museum.
Information and application: http://www.jazzperiments.com.

1.2.1 Origin and development

About 10 years ago I was playing clarinet in a jazz band when my colleague-sax player decided to connect a mic via a preamp to some guitar effects like 'chorus' and 'overdrive'. Of course I tried it too. The results of our experiments were loud, sometimes interesting, but very rarely attractive. My conclusion was, that it was useless to apply 'effects' in this way, as long as these 'effects' did not 'understand' the music played. Being a programmer I decided to take up the challenge of building an interactive sound application with a basic 'understanding' of music. This was the starting point of my experiments.

The first version of *Jazzperiments* was built in *Synthedit,* a C-application and -library. This tool is designed to build synthesizer simulations and includes a crude Fourier module which measures the frequency of audio input, also known as 'pitch tracking'.

In 2007 I was invited for a residency project at STEIM in Amsterdam and had the privilege to work and play together with Michel Waisvisz, Sabine Vogel and Alex Nowitz. Michel Waisvisz inspired me to enable the use of live samples (at least as an addition to MIDI), similarly to his use of live music samples in the famous *Hands*.

In 2008 and 2009 *Jazzperiments Jam* was completely rebuilt in Java. With the help of Karl Helgason, the developer of the Java Sound engine called Gervill, a highly optimized Fourier module and automatic live sample recorder were included. The performance of the system is 'state of the art' now and it runs on every pc and Apple Mac computer.

In order to improve the accuracy of 'pitch tracking' and to avoid 'avoid notes', a function was built in to filter out all notes which didn't match the scale set (or recognized). A list (addendum 5.1) of 'all' scales was included to choose from. Notes which are recognized are played back with selected octave transpositions and smaller intervals, random picked from the selected lists (addenda 5.2 and 5.3).

In the following years the *Jazzperiments Jam* application was adapted and used for interactive sound art projects (*Time Canvas* in MuHKA, *According to nature* at Verbeke Foundation, *Big Bang* in Opera de Lille, *According to Cage* in ZKC Middelburg). Artists, among whom Lynn Cassiers and Miguel Sosa have used *Jazzperiments Jam* intuitively and with remarkable results.
Beautiful sample: Aria of Lynn Cassiers: http://www.jazzperiments.com/mp3/ARIA_5-9-2012.mp3

1.2.2 *Jazzperiments Jam* now

At the moment *Jazzperiments Jam* is easily and intuitively useable for soundscape-like artistic projects. Samples are played back with a transposed pitch related to the real time input, which makes the output of *Jazzperiments Jam* sound surprising and familiar at the same time. The possibility to automatically record samples of variable sizes without the need to click any button or key is very important to create installations in which the user only has to make a sound to trigger a musical response.

However in 'common' musical settings (like jazz jam sessions) *Jazzperiments Jam* doesn't translate all information available in a useful manner yet. E.g. the option to switch automatically to the right scale based on the pitch of the first note played after a pause, 'looks' nice, but is hardly ever used in an effective way. Just leaving such a required pauses to 'manage' the program obstructs intuitive improvisation.

1.3 Further development

The objective for further development of *Jazzperiments Jam*, is to optimize the application for live improvised interactive music. To be more specific: any musical input should trigger an according or at least meaningful response or accompaniment.

2 Theory and practice

2.2 Why computers don't listen

As mentioned in the introduction, interactive computer music often relies on fancy interfaces like gesture controls and cameras. The most likely (often built-in-for-free) interface - the microphone - is hardly ever used. Still, it is clear that using audio input would allow for singers or acoustic instruments to interact with computers and would make the whole experience of playing music together with computers more natural.

However, there are two main reasons why the use of a microphone as an interface for interactive music makes things really difficult ... why computers just don't listen very well: feedback and latency.
Feedback is the reaction of a system on signals which it produces itself. This can be both physical (audio) or logical signals.
Latency is the delay caused by a system processing information needed before it can react.

Both feedback and latency are ever present in real life, but our brains all have had a lifelong training to cope with it. Our brains have learned to use feedback as a tool and can ignore latency in an incredible way, but because of this it is hard to image the problems computers have with it.

Specially latency has to be studied in detail and in an open minded way to find practical solutions for the problems it poses in any interactive music.

2.3 Latencies (plural)

Latency is unavoidable in all signal processing systems whether electronic, mechanical or pneumatic. Church organs are infamous because of their inherent latency. Yet, this didn't keep composers like Bach from creating some of the best music ever on/for this instrument.

In interactive music, it is possible to identify different kinds of latency. Let's take the following setup as an example:
A singer stands in front of a microphone and sings a note to a computer which will retransmit this note as a midi piano sample. How long will it take before the singer hears the computer playing his note?
In the next paragraphs is described what happens along the way with an estimation of the latency building up.

2.3.1 Audio input latency

Sound needs 3 ms per meter to travel from the mouth of the singer to the microphone.
Live: 3 ms.

2.3.2 Electric input latency

The signal needs 3 ns per meter to travel through the cable from the microphone to the amplifier and some extra nanoseconds to get to the computer.
Live: 0 ms.

2.3.3 Minimal sample latency

The computer calculates the pitch with Fourier analysis. At least one wavelength of sound is required to make these calculations. As a consequence lower notes cause more latency. Where a 440 Hz note needs 2 ms to be recognized, a 40 Hz bass note needs 25 ms.
Live: 10 ms.

2.3.4 Attack latency

A very important but often ignored kind of latency is caused by the instrument playing the note. Every percussive instrument - including a piano - plays a note which sounds higher shortly after the attack (due to a greater amount of high harmonics produced when the 'hammer' touches the 'string'). Wind instrument -

including the human voice - by contrast produce a sound which is initially lower (due to air pressure building up) then the note aimed for. Therefore, for almost any acoustic instrument including the human voice it is impossible to measure the correct frequency before at least 20 ms have passed since the note was first picked up.
As this latency is inherent to the instrument(s) producing the input sound, it can never be reduced or avoided. And to make things worse: our brains are perfectly capable of ignoring this kind of latency: we interpret notes as in pitch even when they were initially quite false.
Live: 20 ms.

2.3.5 Signal processing latency

Modern computers can reduce the time needed to process the information to a matter of microseconds depending of the processor and the algorithm used. Up until quite recently signal processing and calculations contributed a great deal to latency, but this is no longer the case.
Live: 1 ms.

2.3.6 Audio synthesis latency

Once the computer 'knows' which note to play, it has to find the sample needed, put it into a buffer of the sound processor and play it. The optimal size of the buffer is rather different for speech (Skype etc.) and music. It is necessary to install the right drivers (ASIO) and set the right buffer size to avoid latencies upto 300 ms (!) and bring it down to approximately 10 ms.
Live: 12 ms.

2.3.7 Electric output latency

The signal needs approximately 3 ns per meter to travel through the cable from the amplifier to the speakers.
Live: 0 ms.

2.3.8 Audio output latency

The sound takes 3 ms per meter to travel from the speaker to the ears of the listener.
Live: 6 ms.

2.3.10 Total latency

In total in a live performance the latency now is approximately 48 ms. And the specification above shows that most of this latency can't be reduced by technical improvements. It is for instance clear that it is impossible to ever solve latency problems by increasing processing power.

In musical terms a latency of 48 ms is, at a tempo of 100 bpm (beats per minute) less than 1/32: a very short note, but still a note which can't be ignored.

Traditional 'guitar effects' like reverb, chorus or overdrive also involve latency, but there is a huge difference, because in those cases the signal isn't measured/interpreted, it is only transformed and played back (from source to ear) in about 12 ms (1/128 at 100 bmp).

2.4 Coping with latency

2.4.1 Always too late

Waiting for some sound to be made and then starting the process to find out what to do next, isn't a good strategy to enable interactive computer music. However if a system is prepared to play the 'right' note as a reaction on a signal (key-input, beat of metronome, etc,) the latency can be reduced to acceptable delays.

Really interactive music requires that the computer 'knows' what is played before it reacts and therefore has to cope with the latency involved in analyzing the signal (volume and pitch). And coping with latency is necessary - for humans and computers alike - in all kinds of music, if more than one player is participating. If a musician only reacts on audio signals he receives, he will always be far too late to play along. He has to follow the beat and play the anticipated note - in sync - at the right time. Anticipation is the neurological solution to latency and it is the logical solution to handling latency in interactive music systems. Anticipating or estimating the notes to be played can be achieved in many different ways, as will be discussed in chapter 2.6.

2.5.2 Measuring tempo and handling feedback with pitch tracking

Although the latency involved is hard to handle, accurately measuring the pitch of notes has some great benefits, especially in live interactive performances.
When the system 'knows' the pitch of notes played, the number of notes and therefor the tempo of a phrase is also known. Measuring the tempo of the music in this way is faster and far more accurate than when only the variations in volume are taken into account.
Avoiding feedback also becomes much easier: if the computer 'hears' the note it just played itself, it can easily ignore it as 'external' input. In systems which do not process, but just transform the input, feedback is a real problem.

2.5.3 On the beat

Accurately measuring the tempo of music played, opens a way to reduce the impression of latency. With a latency of 48 ms (approximately 1/32) a listener/player will always have the impression that the system is annoyingly lagging behind, even if it reacts as fast as possible. However if the system reacts somewhat later, but on the beat, this impression of latency is reduced. Especially if the system can follow variations in the tempo of the music played (as is possible with *Jazzperiments Jam*), the reactions aren't perceived as too slow.

2.6 Anticipating/predicting the next note

As it is completely impossible to measure the pitch of notes played fast enough to play along immediately, it is necessary to anticipate/predict/estimate which note will be the best to be played at the following beat. The sources of information to estimate this note are the settings made by the user and the 'history' of the notes played before. The 'historic' information can be approached by means of musicological theory or pattern matching (and with a combination of both).

Musicological

2.6.1 The song is set

If the score of the piece which will be played, is imported as information before the session starts, during the session the next note can be predicted with a 100% certainty. The tempo measurement can be used to keep track of the tempo of the players. However in this setup spontaneous influence on the music is impossible. The system would be something like *Band in a Box*, which exists already of course (and works great).

2.6.2 The song is recognized

Existing applications, like *SoundHound,* can recognize songs while they are played. If the score of the songs could be downloaded immediately, this would make it possible to predict the next note with certainty and still allow a player to change the song along the way and for instance play medleys with the computer. A project like this could be great for (a cooperation between) the makers of *SoundHound* and *Band in a Box* and might result in a kind of sing-whatever-you-want karaoke system. However, for improvised music this approach doesn't work, because there is no existing song to be recognized.

2.6.3 The scale is set

If a system knows which scale it has to use for a session/phrase, it is possible to exclude all notes which do not belong to this scale from the output of the system. This is already possible with *Jazzperiments Jam* and when a player improvises with a certain 'standard' song in mind (without necessarily playing this song - just using its chord pattern as a kind of guideline) the results are quite nice, but for complex songs with a variety of chords form different scales, this approach is useless. Free improvisation is impossible too.

2.6.4 The scale is recognized

By analyzing the history of notes played before, it is possible to recognized the scale and/or the chord played. Based on this information it is possible the avoid playing a note which doesn't belong to the scale or chord played and would be dissonant, resulting in too much 'tension'. This offers a way to avoid playing the wrong note, in jazz idiom known as 'avoid notes'. Every jazz musician knows that you can play almost anything during improvised soloing ... except the 'avoid notes'.
However to recognize a scale accurately about 8 different notes have to be measured. For chords the same applies, as with fewer measurements a chord still can belong to different scales. Accurately recognizing a scale therefor takes quite some time.
However even based on as few as 1 measurement some predictions can be made. Assuming that changing the scale of music played happens only when a new phrase or movement starts, these changes are relatively rare and avoid notes can be predicted quite accurately.

2.6.5 Arpeggios and chord progression

Short notes are often part of arpeggios and chords are often played in more or less fixed 'progressions', both in classical music and in jazz. The next note to be played can sometimes be estimated quite easily as it is part of a musicological ordered list of notes.
Because this is only possible when the scale is set or recognized accurately, this approach has the same limitations as the musicological options mentioned before.

Pattern matching

2.6.6 Repeated notes

If the last note recognized is the same as a note played before during a session, the following 'historic' note would probably sound nice again. This is even more likely if previous notes also match, for instance if notes are part of a repeated arpeggio or - on a larger scale - chord progression. Repeating patterns of notes occur in any kind of music and therefor pattern matching is very useful to estimate notes to be played.
A big advantage of this approach is, that the intervals between the historic notes are preserved in the new output and in this way the harmonic relationship between the notes is maintained. Pattern matching 'recognizes' (parts of) scales and chords and implicitly involves musicological methods.

Of course pattern matching requires that notes which are played during a session are stored for processing.
It is possible to keep track of all notes recognized (including recorded notes played by other participants during a session) or just the notes played by the system itself. It is hard to predict how this choice will influence the result, but it will be very interesting to experiment with it.

Of course, not all historic notes are equally important for an estimation of the notes to be played. The importance of historic notes could be translated to a 'weighting' to be observed in the estimation.

2.6.7 'Age' of notes

For an estimation of the best note played next, the notes played in the last phrase (typically 4 bars) are more significant than notes played earlier.
A distinct pause in the music could be a signal to reduce the weighting of the notes played before the pause as it is likely that new scales and/or chords will be involved after it.

2.6.8 Length of notes

If the system has to estimate the notes best played next, the last note recognized is probably the most important clue for the estimation. However if this was a short note (like in arpeggios) the previous notes could be important too for a prediction. In many cases the notes played for at least one beat are the most significant for the underlying melody.
In the history of notes played, the notes which are shorter than a beat could be ignored (get a low weighting) as these are probably part of the ornamentation of the music, not the 'main' melody.

2.6.9 Part of repeating pattern

If a repeating pattern is recognized the weighting of the note corresponding with the note to be played next could be increased. The more the weighting of this note is increased, the more the system will 'stick' with the previously played pattern/phrase.

2.6.10 Ignoring low tension intervals

If the interval between two notes is exactly one or more octaves, it is often difficult to hear the difference especially if the notes are played on different instruments. Also fifth intervals (50% of an octave) show little tension with the ground note.
It is possible to ignore octaves and (maybe) fifth intervals when comparing notes for pattern matching. In this way the chance of finding matching notes can be increased dramatically.

2.6.11 Over all conclusion

Pattern matching probably is the best basic technique to estimate the notes to be played next, as repeating patterns are frequent in almost any music. Musicological considerations can be used to improve the pattern matching processes. The fact that the pattern matching algorithm is highly flexible is also an advantage of this approach.

2.7 Risk management

An objective of this project is to build a system which can act as a 'player' in free improvisation. This means that at any moment other 'free' participants can decide to play notes which couldn't be predicted in any way. It is therefore necessary to anticipate the risk that the system doesn't 'know' what to play next ... just like human musicians during improvisation.
Options for 'risk management' are:

2.7.1 Silence

It takes a very long time before silence is perceived as an error: not playing anything, is hardly ever wrong. So the system should be programmed to play nothing, if the input doesn't contain useful information or is 'risky' in some respect.
For instance: if the note to be played was never played before in a session which lasts already more than 32 bars: it shouldn't be played.
Simple principles like this are already used in the *Jazzperiments Jam* application and are probably the main reason why people interact so intuitively with it.

2.7.2 Bigger intervals

Playing a note which is close to the 'right' note can sound very dissonant/wrong if it is just half a note too high or too low. The bigger the interval between the right note and the wrong note, the smaller the change that the notes involved are perceived as dissonant.
If a next note will be played based on rather limited information (for instance if the previous note is played only once before in the session) the chance of it being dissonant can be reduced by playing a note which is an octave (save) or a fifth (more daring, but often interesting) higher or lower than the note originally estimated.
The choice of a lower or higher note can be based on 'historic' patterns being ascending or descending.

2.7.3 Repetition

Every jazz musician knows that during improvisation wrong/strange notes can be made to sound corrected/agreeable/intentional if they are repeated. At the moment it isn't planned to use this in the algorithm, but it is good to keep it in mind.

2.7.4 Shorter notes

When it isn't certain that the notes to be played are the right ones, the best option is to play short notes (1/8 or less). Of course, also this 'solution' is well known in jazz. At the moment it isn't planned to use this in the algorithm, but it is good to keep it in mind.

2.8 Chance operations

In experimental music chance operations and random notes are commonly used. In some ways, this can be considered a legacy of pioneers like John Cage. However, in interactive music without pitch tracking (but including interactivity based on camera input or dynamic input devices) so little tonal information is available, that resorting to random notes is the only option.
When pitch tracking is available and notes played can be recognized, more than enough tonal information is available to produce a response with only intentional/calculated notes.
In *Jazzperiments Jam* random functions are only used to select notes which can be considered to be equally useful. One of the objectives of this project is reduce or eliminate this. When statistical analysis of a history of notes played is added, the chance that different notes are equally useful is negligible.

Avoiding the use of random functions in interactive music is very important to allow interactive improvisation. If an interactive system doesn't 'know' what to play, silence is the best option; throwing in a random note certainly isn't. Acquiring as much information as possible (and coping with the latency problems involved) allows the system to be 'patient' enough to wait until it can play a note which fits in the 'conversation' and doesn't disturb or interrupt it.

3 And the best next note is...

3.1 Strategy

The system will only play notes, when the microphone picks up sounds (of any sort).
If a pattern of notes is recognized, the system will follow that pattern and thus support repetition of phrases and themes.
If no pattern is recognized, the system chooses notes with a harmonic interval with the note measured last of at least a fifth, to avoid too much tension.

3.2 Flow

```
0     Any sound playing louder than the minimum level?
No:
      Silence
      Exit
Yes:
      Go to 1
1 Last note recognized is found in history?
      No:
            Go to 2
      Yes:
            1.1 Preceding notes matching in pattern?
                  No:
                        1.2 Was the historic note => 1 beat long?
                              No:
                                    Go to 3
                              Yes:
                                    Play same note next in history plus or minus* a 5th on the next beat.
                                    Exit
                  Yes:
                        Play same note next in history on the next beat.
                        Exit
2 Is the last note recognized =>1 beat long?
      No:
            Go to 3
      Yes
            Play recognized note plus or minus* a 5th on the next beat.
            Exit
3 Next note is unpredictable. Did the system play any note during last 2 beats?
      No:
            Repeat the note the system played last.
            Exit
      Yes:
            Silence.
            Exit
```

***** Depending of the pitch of previous notes played and in this way fitting in an ascending or descending melody.

3.3 Weighting

All notes recognized during a session are stored with their length and pitch. This 'history' can be analyzed to modify and improve the flow described in the previous paragraph. The analysis and calculations can be performed in a separate process (to avoid extra latency) and the results can be stored along with the history as a weighting of the notes.
During the process to estimate the next note to be played by the system, notes with the highest weighting are more likely to be used.

Criteria for the weighting of the notes are:

3.3.1 'Age'
The weighting of notes played longer ago in the session is less than of more recent notes. This implies a gradual decrease of the weighting in time.

3.3.2 Length
The weighting of short notes (< 1/8) is smaller than longer notes as longer notes are more significant for the underlying melody. In the flow described in the previous paragraph the conditions 1.2 and 2 (note => 1 beat) can be made more nuanced.

3.3.3 Pauses
If a pause occurs, the weighting of notes before the pause can be reduced, depending of the length of the pause. Notes played before a long pause (> 1"), can in this way be considered to belong to another part/movement and to be unrelated to the following part(s).

3.3.4 Recorded notes
Jazzperiments Jam allows players to record samples during a session. The notes played in this recording could get extra weighting, because they are intentionally added to the piece and are repeated in the samples (be it at different pitches).

4 What's next
4.1 Planning

Further development isn't restricted to a limited timeframe, as new developments will lead to new questions en even more interesting developments.

However within 3 month the most promising approach should be selected and implemented and within 6 month, before 1 June 2014, a new test version of *Jazzperiments Jam* should be available for live performances and new experiences.

A blog of the discussion and experiments will be available on the website http://www.jazzperiments.com/nextnote.

5 Addenda

5.1 'All' scales

01	Augmented	32	Locrian Natural 2nd
02	Balinese	33	Lydian
03	Bebop Major	34	Lydian Augmented
04	Bebop Minor	35	Lydian Dominant
05	Blues	36	Lydian Minor
06	Chinese	37	Major
07	Chromatic	38	Marva (Indian)
08	Diminished	39	Melodic Minor (asc)
09	Dorian	40	Melodic Minor (desc)
10	Double Harmonic	41	Mixolydian
11	Egyptian	42	Mixolydian Augmented
12	Enigmatic	43	Natural Minor
13	Ethiopian	44	Neapolitan Major
14	Flamenco	45	Neapolitan Minor
15	Four Semitone	46	Oriental
16	Half-Whole Diminished	47	Pelog (Balinese)
17	Harmonic Major	48	Pentatonic Major
18	Harmonic Minor	49	Pentatonic Minor
19	Hindu	50	Persian
20	Hirajoshi (Japanese)	51	Phrygian
21	Hungarian Gypsy	52	Phrygian Major
22	Hungarian Major	53	Rock and Roll
23	Hungarian Minor	54	Romanian
24	In Sen (Japanese)	55	Spanish 8 Tone
25	Indian	56	Super Locrian
26	Iwato (Japanese)	57	Symmetrical
27	Japanese	58	Three Semitone
28	Javanese	59	Todi (Indian)
29	Leading Whole Tone	60	Ultra Locrian
30	Locrian	61	Whole Tone
31	Locrian Major	62	Whole-Half Diminished

5.2 Possible octave transpositions

Relative		*Special*		*Absolute*	
01	-2 octaves	05	Inverted	06	Great (C)
02	-1 octave			07	Small (c)
03	+0 octaves			08	1-lined '
04	+1 octave			09	2-lined ''
				10	3-lined '''

5.3 List of small intervals in order of possible tension

1	I (same note)
2	I or V
3	I or V or vii
4	I or V or vii or III
5	All scale notes