IMPROVISING COMPUTER MUSIC: AN APPROACH

Leonello Tarabella
computerART project of ISTI/CNR
via Moruzzi 1 - 56124 Pisa, Italy
leonello.tarabella@isti.cnr.it
http://tarabella.isti.cnr.it

ABSTRACT

I describe here my experience in interactive and improvised electro-acoustic music after having developed both hardware and software tools using which I compose and perform music. The research in the whole finds the roots in my active involvement in jazz music. This paper analyses the jazz approach to improvisation in terms of the involved components and emphasizes the importance of expressiveness and feeling in live performance. Then it proposes a schematic approach to improvising computer music, which gives musicians the same expressive approach, found in jazz, in the new electro-acoustic domain. Two different original gesture recognition devices and systems are described together with a real-time music language, or better, framework based on C-language for sound synthesis and event management. The role of mapping is proposed as crucial and peculiar of the new approach.

1. INTRODUCTION

In this paper I report my personal experience in interactive electro-acoustic music which, no doubt, has been derived and influenced by my active involvement in jazz music. In writing this paper I made use of parts of previously published articles of mine [1,2,3,4,5,6] - here properly revised and linked together - where I describe different aspect of my research activity dealing with the realization of gesture tracking devices and systems and with the development of a compositional frameworks based on C-language.

I first give a schematic vision of jazz music in terms of the various components involved and how they are dynamically connected while playing. Then I report the principle of operation of the gesture interfaces I developed in their more recent upgraded versions. I also describe the pCM (pureC Music) framework which gives the possibility to write a piece of music in terms of algorithmic-composition-based programs controlled by data streaming from external gesture interfaces. Finally I sketch out the importance of the role of mapping and propose a new schema where some of the previous components change *allocation* and new ones are introduced for improvising computer music. Some considerations must be done before starting.

In concerts executed with traditional instruments the visual aspect plays an important role for the emotional communication between the artists and the audience: dressing and behavior of performers on stage and spatial location of musical instruments directly controlled by the performers, guarantees the audience in the reality and completeness of a human-to-human artistic communication. Interactive electro-acoustic music proposes a complete new scenario not even for the new sound palettes introduced but, rather, for the new kind of relationships experienced by the audience between what-is-going-on-on-stage and the final musical result.

A traditional music instrument is a compact tool; the new electro-acoustic instrument is a system consisting of a spread out number of components: sensors and controllers, computer and sound generators, amplifiers and loudspeakers. How to link, that is how to map, information between the various parts of this exploded instrument is deeply correlated to new modalities of composing and performing in relationship with how the audience perceive and accept that.

2. JAZZ IMPROVISATION

The history of music, all over the genres, is plentiful of great characters that - besides composers - have been great improvisers too. However it's in jazz that the practice of improvising assumes distinctive and peculiar characteristics and reaches the higher levels of variety and sophistication. As well known, improvising in jazz basically consists of creating and performing *in real-time* a melody, which fits a chord progression. Actually, improvisation is a very challenging craft which requires the coordination of many factors at the same time such as chords and scales structures, form, rhythm, articulation, patterns, feeling... The Art-of-Improvising requires that a musician has deep knowledge of all that and he is skilful at both theoretical and practical levels.

Taking into account what Charlie Parker used to say (*learn everything about music and your instrument, then forget everything and play*!) and what Jerry Coker say in the first chapter (the Improviser's Basic Tool) of his famous book “Improvising Jazz” [7] about the roles of intellect, training and feeling, I here sketch out these various components and their relationships present in the mind and in the hands of a jazz musician.

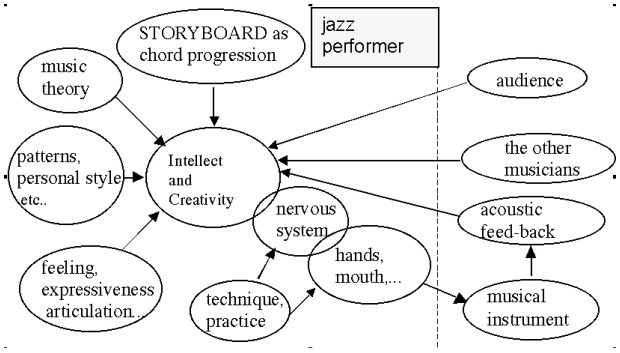


Figure 1. Components in jazz improvisation

Improvisation can be considered as the result of coordinating all these components at the same time. The solo is improvised on the basis of a chord progression taking into account the music theory of harmony, the reference to a personal database of patterns and style. Feeling and expressiveness must be added with the proper technical practice over the musical instrument.

It's a matter now to try to propose a similar situation where a musician can actively express his creativity in the different musical context offered by computer music. Here the chain of mechanic, electronic and electro-acoustic elements is longer and indirect in respect to the simple relationship between a musician and his instrument. So, it's no more possible the same direct expressive approach as in jazz music.

A new arrangement of the elements and a new way of using them, while introducing new ones regarding computer technology must be proposed. This will be discussed in paragraph n.6 after having described the results of my research regarding the realization of gesture tracking devices and systems and of a special-purpose music-language framework based on C-language. This allows algorithmic composition at both event and audio-signal levels and offers the right facilities to put at works the mapping paradigm, that is how to link information issued by gesture interface to sound synthesis.

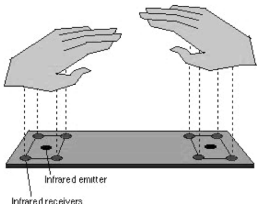
3. GESTURE TRACKING SYSTEMS

In computer generated music the basic element is *sound* in its very primary essence rather than the notes and the relationships between them at both horizontal (melody) and vertical (harmony). So, an ordinary 12-tone midi keyboard it's no more suitable and satisfactory for controlling digital music. For that the need to invent and realize new interactive media has been deeply felt in many research centers all over the world. In the computer music field a great variety of very sophisticated and complex gesture interfaces have been designed and realized using almost any kind of sensor [8,9,10,11,12,13,14].

During the '90s together with other researchers and collaborators of C.N.R. in Pisa, I realized a number of devices and systems based on the infrared (IR) and the real-time analysis of video captured images technologies: TwinTowers [2,3], Light Baton [15], UV-Stick [3], Imaginary Piano [3] and PAGe system [16,17].

3.1. TwinTowers and PalmDriver

These devices are based on IR technology and consist of groups of four elements arranged as the vertical edge of two parallelepipeds. The measurements of distance of the different zones of the hands' palm are performed by the amount of reflected light captured by the receivers and are quite accurate in respect to the irregularity and color of the hands' palms. The devices are stable and responsive; as a consequence, sounds generated by the computer evoke on the performer the sensation of *touching the sound*. This sort of psychological feedback greatly contributes to give expression to computer generated electro-acoustic music. After having presented the Twin Towers many times at technological and artistic level [18,19,20], we recently developed a new version consisting of 4 groups of 8 elements, which works as a standalone device properly equipped with a MIDI OUT port. The main feature of this new device named "PalmDriver" is modularity: the four modules can be spatially placed in many different planar configurations.



3.2. Handel

Image processing technology has been used for realizing the other systems. A CCD camera is connected to a video grabber card and the digital image to be analyzed consists of the reconstructed image by means of an algorithm which filters (that is, accepts) those pixels whose luminance is greater than a predefined threshold. Although this algorithm would be not applicable to a generality of images, it is precise enough to distinguish the luminance values of those pixels corresponding to the hands from the rest of the scene. Besides, in order to improve the robustness of the method, the performer dresses in black and has at his shoulders a black background.

This system named *Handel* and described with details in [6], gets information from shapes and positions of the hands. Handel is very flexible, fast and truly usable thanks to the high number of parameters put at disposal at the same time. In the same manner as it happens in the well-known BigEye application, it's possible to define sub-zones where to apply the analysis process. The sub-zones where to run the analysis can be dynamically defined. The whole system is based on ordinary devices such as an analog CCD video camera, a Capsure frame grabber PCMCIA card by IREZ able to convert images with 320x240 pixels at a rate of up to 30 frame/sec and a Macintosh PowerBook G3-500Mhz. The analysis algorithm has been optimized in order to follow this high sampling rate.

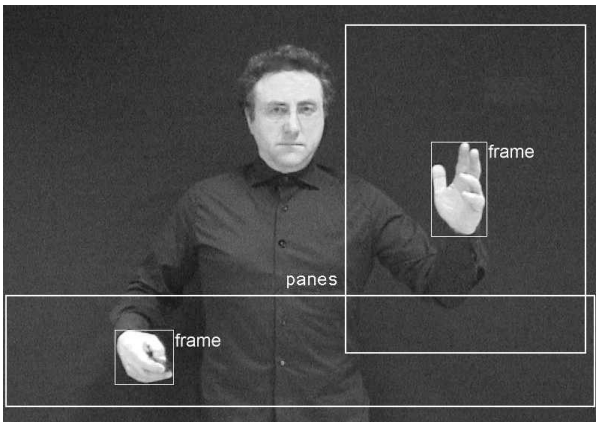


Figure 3. Typical situation in Handel

In the following I'll use this terminology: *panes,* i.e. the *defined* sub-area that can be placed everywhere in the capture video camera area with whatever dimensions; *frame,* i.e. the *detected* rectangle that delimits the shape of one hand considered as the reconstructed white spot in memory, therefore defined as *spot-hand.*

The algorithm that scans and analyses the hands' postures and movements is simple in principle but, at the same time, it allows a great variety of dynamic figurations truly important for the overall impact on the audience during the performance. Usually the hands assume postures that show the palm or the back in respect to the CCDcamera and the audience such as those reported in Fig.3.

Fingers can be kept closed together or kept in the fist position. Furthermore many combinations of finger-closed/finger-open (such as when counting) can be taken into consideration. With this class of postures the resulting frames are nearly squares so that values to be consider are those related to the mass and the position within the rectangle. The formulas and the operative code program are the well-known formulas for computing the center of mass. Frames dimensions are simply given by the difference of the coordinates between the extreme points of the spot-hands.

A second class of posture produce *flat* frames, i.e. where one dimension is considerably lower in respect to the other. This is the case where the forearm is placed horizontally and the open fingers point to the camera (mime an airplane flight with thumb and little finger as the wings) or in the posture used in the military salute.

Figure 4. Two positions of two different *flat* postures

With this class of posture the resulting frames are flat and then it makes sense to recognize the angle of rotation. This is computed used the well-known regression-line formulas. As summary, these are the information detected by the program.

- spot-hand **presence** (true/false).
- spot-hand **barycenter** (*x,y)* coordinates
- spot-hand **frame** dimensions (*base,height)*
- spot-hand **angle** of rotation

As a final remark, I want to highlight that it's not a matter of recognizing the shape of the hands but, rather, that of freely controlling size, position and rotation of the spot-hands, which, in turn, change the frames dimensions, and rotations. At the end the hands really control parametric values for giving expression to real time synthesized music. It's so possible to invent many and new postures and movements to be used in different musical compositions with any sort of free linkage with the theme and the poetics of the music. I mean that the great variety of shapes, postures and movements of the hands that can be invented by the composer/performer creativity, can be mapped into the *frame* classes so far described.

On the basis of Handel, the *Imaginary Piano* has been realized. Here a single rectangle - with a small thickness, as long as the horizontal dimension of the video capture area - has been defined at the height where the keyboard usually lies.

Figure 5. Imaginary Piano

When a finger, or a hand, crosses that strip downward, the systems reports proper information regarding the “key number” and a specific message is issued accordingly to “where” and "how fast" the rectangle has been crossed. Information is used for controlling algorithmic compositions rather than for playing scored music.

4. pureC MUSIC LANGUAGE

For composing and for performing interactive computer music I realized a framework based on pure C programming, that is *pure-C-Music* or *pCM* [4]. This programming framework gives the possibility to write a piece of music in terms of synthesis algorithms, score and management of data streaming from external interfaces. As a result a pCM composition consists of a C-project assembled with all the necessary libraries able to implement in realtime the typical synthesis and

processing elements such as oscillators, envelope shapers, filters, delays, reverbs, etc.

The composition itself is a C program, which mainly consists of the Score and Orchestra parts. The Object Oriented paradigm is mainly used for defining instruments in terms of class declaration then instanced as many times as wanted. Everything is compiled into machine code that runs at CPU speed.

`Storyboard()` is the default void function called by pCM that starts the program/composition.

```
void Storyboard() //called  by pCM
{
   AudioOpen
   MidiOpen
   Intro();      //The movements are called
   Movement1(); //in sequence and are
   Movement2();  //defined  by  the  composer
   as
   Finale();     //void functions
   MidiClose
   AudioClose
}
```

Each movement is defined as a void-function consisting of a two main parts: the Score that generates the events and the Orchestra, which generates the audio signal. The Score is the program part, which triggers and *feeds* the instruments by assigning proper values to common variables. Instruments are defined in terms of synthesis algorithms inside the Orchestra. The score-program is intended as an algorithm (from simple to complex) which also may include sequences of note-events as requested by scored music. This is an example of a movement:

```
void Intro();
{
   float
cmd,chn,val1,val2,valR,valL;
   oscillator oscL,oscR;
   Score
   {
      GetMidi(&cmd,&chn,&val1,&val2);
              //or any other external event
       ....instructions ;

       Orchestra  //synthesis  algorithm
       {
              valL = Osc(oscL,val1*...);
              valR = Osc(oscR,val2*...);
              outLR(valL,valR);// to DAC
       }
   }
}
```

All the necessary variables are defined following the C language syntax. Values are assigned to variables by instructions which make part of the program defined in the Score section, that is the composition, or by data coming from the external. The Orchestra uses variables for storing temporary values and for getting parametric values computed by the Score. Actually, the block {.....} which follows the **Orchestra** keyword, is a *hidden* loop which repeats the synthesis algorithm as many times as the number of locations of the audio buffer. A typical value of the audio buffer length (in number of locations) is 512, so that, since the Score too is a *hidden* loop, the inside program continuously repeats 44100/512 ≈ 86 times/second. A movement makes use of the functions belonging to the original library, which puts the pCM framework at work. This library consists of three different groups of functions named *toolkits*, each one devoted to specific tasks: the DSP toolkit, the Events toolkit and the Command toolkit.

- The **DSP toolkit** deals with synthesis and processing of sound. It groups elements such as oscillators, envelope shapers, filters, delay lines, reverbs, etc.. For example an oscillator is defined as follows:

```
oscillator myosc=newOsc(phase);
```

used in **Orchestra** in this way:

```
val = Osc(myosc,freq);
```

A delay line is created with

```
delayline mydelay=newDelay(duration);
```

used in **Orchestra** as follows:

```
PutDelay(mydelay,value);
.........
v = GetDelay(mydelay);
```

- The **Events toolkit** deals with the generation and the scheduling of events including timing and management of external events. Events are managed by the so-called *Scheduler mechanism*. A Scheduler is an element that gives the possibility to queue timed events in order to be taken into consideration later at the *right* time.

```
schedulermysched= newScheduler(maxNumEvents);
```

The `Event(schedname, dur,value)` function queues the event defined as duration-value couple, into the specified scheduler. This function is usually invoked at Score level and can also be affected by data coming from the external. Once the events are placed in the Scheduler queue, the instruction

```
if(nextEvent(schedname,&retval))
doSomething(retval);
```

is used for checking whether or not the time duration of the current event is finished. If yes, **nextEvent** returns *true* and *retval* has a valid value of the next event, which will be used in the instruction *doSomething* that usually trigs an instrument.

`GetMidi(&cmd,&chn,&val1,&val2);` is a boolean function which returns *false* if no midi message has been received; otherwise it returns *true* and the cmd, chn, val1, val2 variables report valid values.

- The **Commands toolkit** controls the activation of the computer built-in CD player; allows to directly record onto memory the audio signal and to store it onto disk as .aiff or .wave file; provides miscellaneous mathematical functions.

4.1. Instruments as objects

The **Orchestra**{synthesis algorithm} block computes the audio signal by processing the instructions which implements the instruments as defined by the composer using the DSP toolkit functions. The Object Oriented paradigm is mainly used for defining instruments so reaching a cleaner layout in writing the program-composition. An instrument is then defined as a class object and declared, that is, instanced in the movement as many times as required. The following is an example of a simple instrument based on the pluck element with some other additional elements which enrich its functionality:

```
class String:  public Object
{ protected: pluck       string;
             envelope   envks;
             lpfilter   lpfks;
             float        vks,val,pitch;
  public:    void         setup();
             void         trig(float frq);
             float        tick();
             ~String();
}
```

The String class is defined as public object, which includes both the protected section where the elements and the variables are defined and the public section where the methods are declared. Usually, in the class instrument declaration three are the methods declared, plus one for destroying the instanced objects. These methods do the following tasks: -set up everything necessary in order to the object works properly such as to create delay lines, envelopes, etc.; -activate (**trig**) the synthesis algorithm and, finally, -perform (**tick**) the synthesis algorithm which actually computes the signal. Then the methods are given.

```
void   String::setup() //setup for the
                       //instanced  object
{
  string=newPluck();
  strflt=newLPFilter();
  float e[]={3,0,0,1.2,.01,1.,1.5,2.,0,.0}
       //defines   breakpoints   for   the
envelope

  envks=newexpEnv(e);
}

void   String::trig(float frq)
{
       pitch=frq;
       trigPluck(string,pitch);
       trigEnv(envks);
}

float  String::tick()
{
  vks = Env(envks)*(Pluck(string,pitch);
  val = LPFilter(lpfks,vks,cutfrq);
  return val;
}
```

The pCM framework has been implemented first for Macintosh computers using the CodeWarrior C compiler by Metrowerks. What here reported refers to the last version I developed and use for composing and performing. With the introduction of the ASIO libraries available for MacOSX and Windows, I'm now developing a new version for both platforms I'm going to put on the Net as a freeware music language.

5. MAPPING

A traditional music instrument is a compact tool, which gathers together all the aspects (shape, ergonomics, mechanics and material) necessary for stating and determining timbre and for controlling pitch and nuances of sound. The physical structure of the instruments reflects both alphabet and syntax of reference for the music played, reflects the anatomic structure of the human body and forces it to specific postures and gesture. Compared to a traditional *compact* musical instrument the new instrument based on computer technology appears as an *exploded instrument* consisting of different elements: controller(s), audio-signal generator (the computer) and sound sources (loudspeakers) connected via different typologies of cables and signals.

There exist two main types of connections: the digital connection between controllers and computer and the analog connection between computer and loudspeakers. The digital connections are more crucial and interesting. Controllers, or gesture recognition devices, produce data-flows used by the computer for producing sound [21]. The problem now consists in how to link, or better, how to *map* information coming from controllers to programs, which generate complex musical structures, and/or to synthesis algorithms which generate sound signals.

From the point of view of the audience things are difficult to understand especially when original controllers based on different kinds of sensors (pressure, acceleration, heat, infra-red beams, ultrasound, etc.) or gesture recognition systems based on realtime analysis of video captured images, are used by the performer. From my experience, in particular regarding impressions and questions coming from the audience after my concerts, I argue that people usually can appreciate and understand that what is going on musically comes from the presence and the movements of the performer, but in general are unable to understand the complex cause-effect relationships and usually think the controller *is* the instrument. And usually they are completely unaware about mapping and the crucial role of the computer during the performance, as it generates events in accordance with predefined music/acoustic material combined with information from the controllers a performer is acting on. The simple one-to-one mapping rule valid for traditional instruments leaves room for a theoretically infinite range of mapping rules definable by the composer for a specific piece and even for each part of that piece. The mapping is a part of the composition and of the real-time performance.

This is a real novelty in computer music performance. The composer/performer sets up a software mechanism, which uses data coming from a controller to produce sound events: the performer *plays*…not precisely an instrument but rather a *meta-instrument with a particular configuration defined by software*. This approach opens a complete new and wide territory to explore for composition, and especially, for live performance. It is no longer a matter of playing an instrument in the traditional sense, but rather playing a specific piece of music in terms of activating and controlling during the live performance musical/acoustic material and algorithms prepared during the compositional phase [5].

In [13] Silviane Sapir wrote that mapping should be neither too simple nor too complex since in the first case the real power of the computer turns out to be not so well used; in the second case the audience is not able to understand what is happening and cannot appreciate completely the artistic content of the performance. I strongly agree with this observation and, further, I think the rule can and must be extended as follows: we experienced that if a complex mapping situation is reached after a growing-up complexity started using simple (close to one-to-one) mapping, the audience willingly accepts it even if highly complex to be understood. It's important however that the training phase has a *per sé* esthetical and musical meaning.

After one or two episodes such as that, it is possible to use the opposite path that is from a very complex mapping situation to a simple one. This will be accepted by the audience because in some way people are faithful that something will happen to *explain* (artistically speaking) what is going on. Often it happens that someone starts the guess-the-riddle game in his mind. And after a number of episodes such as those described, also with sharp changes from simple to complex and vice-versa, mapping proves to be of interest and well accepted by the audience.

6. IMPROVISING WITH COMPUTER TECHNOLOGY

On the basis of the analysis about jazz improvisation reported in paragraph 2 and after the description of the gesture tracking devices, the pCM language and the consideration about mapping, I give my point of view about the modalities of improvising music with computer technology.

The simple Performer->Instrument configuration found in jazz is now transformed into the longer chain of *elements* Performer > Controllers >ComputerSynth>Loudspeakers.

The novelty entailed by the new configuration is valuable specially when the computer is given an active role during the performance. It's a matter to consider a piece of music as the result of two precise and different moments: - first, when it is composed in terms of algorithms properly programmed (using pCM or Max, or Kyma, etc..) and, second, when it is executed and controlled using data issued by gesture tracking devices. The active role of the computer is possible when a piece of music is devised using the algorithmic composition approach. In this manner it's possible to describe complex musical-acoustic figurations that take shape at the moment of the performance under the active and expressive control of the performer.

The components and their relationships seen in Fig. 1 related to jazz improvisation is now proposed as regard to computer music improvisation taking into account the active role of the *sub-systems* **man** and **computer**.

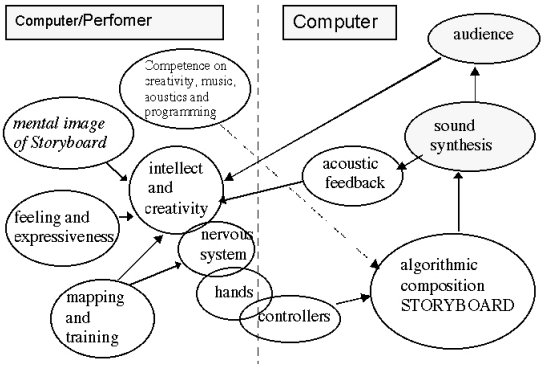


Figure 6. Components and their relationships in computer music improvisation

The component *Competence and creativity on music, acoustics and programming* is related to the moment during which the piece is defined and composed in terms of sound resources, layout and modalities of usage (mapping) properly implemented following the algorithmic composition paradigm.

In this approach, gesture tracking devices and pCM as a whole, must be considered as a void instrument where nothing is pre-established. A new piece of music must be invented in terms of timbre, rhythmic and structure music elements and in terms of new kinds of gesture and how to map it to sound. For that, how to map data issued by controllers to music assumes a crucial a truly new role in executing and improvising computer-generated music.

In jazz in strongly requested the effective practical training on the musical instrument for generating the notes. Here the effective production of sound (and/or notes) is delegated to the algorithms previously programmed. Since sound synthesis algorithms, musical structures, gesture and mapping rule are devised all together for each new piece of music, as a consequence, an accurate training phase regarding how to activate and control with expressiveness the algorithms, is still necessary. In fact, once a musical/acoustic situation has been programmed in terms of timbre, rhythm, *melody* (in a wide sense) and in terms of proper control gestures, it's time to practice gesture in order to reach the right level for a true artistic and expressive interpretation.

I spent no time in trying to define a specific gesture language. Well-described comments in the program listing work fine when rehearsing a piece a music even months later the last performance. At the end, when he performs computer music the way described so far, the performer assumes the same attitude of mind as when improvises jazz.

7. CONCLUSIONS

In an avant-garde concert executed with traditional musical instruments, a default for the audience is that musicians play musical instruments, i.e. that musicians use well-known mechanical "tools" for producing sound, in the same manner a speaker, or an actor, is expected to use his/her mouth. Attention is focused on the content. In a tape-electronic music concert, the artistic message is accepted as an opera prepared in studio in the same manner as a film or a video-clip, no matter how the composer reached the result.

But in a live computer music concert the visual component is of great importance when the new "exploded" instrument is used, just because attention of the audience is also focused on the relationships between gesture of the performer and the music they are listening to. And people want to understand the rules of the new game, besides tasting and appreciating the overall musical result. Then, it's important to plan a storyboard of different situations each one characterized by well-defined musical-acoustic micro-worlds inside of which well balanced "amounts" of simple and complex mapping arrangements between gesture and music should be used.

Following the approach I described in this article, the performance is much more similar to jazz rather than composed&written-scored-music. The difference is that here music is composed&written as a program which at the moment of the performance *actively* executes algorithms also taking into consideration data coming from the external. In this manner, even if composed&written, each performance is never equal to the previous ones.

The emphasis is toward truly *live* performances.

With both meanings: extemporary and vital!

8. AKNOWLEDGEMENTS

I want to thank all the people – teachers, collaborators and students – who in different ways contributed to the development of my work at both technological and artistic levels. Graziano Bertini and Gabriele Boschi who physically developed the electronics of the Twin Towers and Aldo Moretto who realized all the mechanical supports of all the devices I use in my concerts. Massimo Magrini who, as a very skilled and talented musician and expert in acoustics, electronics and informatics, helped me to choose the necessary hardware and to implement the low level crucial routines for grabbing images into memory and for data transmission via UDP protocol. Gianfranco Tommasi and Claudio Fasoli, my saxophone teachers.

9. REFERENCES

[1] Tarabella, L. (1995). *Improvvisazione Jazz e Performance Interattiva con l'uso dello strumento informatico*. In Proc. of the 2nd Int. Conference on Acoustics and Musical Research. Ferrara. Italy. Pedrielli Press.

[2] Tarabella, L., Magrini M., Scapellato G., (1997). *Devices for interactive computer music and computer graphics performances*: In Procs of IEEE First Workshop on Multimedia Signal Processing, Princeton, NJ, USA - IEEE cat.n.97TH8256

[3] Tarabella, L. Bertini, G. Sabbatini, T. (1997). *The Twin Towers: a Remote Sensing Device for Controlling live-interactive Computer Music* - Proceedings of 2nd International Workshop on Mechatronical Computer System for Perception and Action, SSSUP S.Anna, Pisa.

[4] Tarabella, L. (2003) *The pCM framework for realtime sound and music generation* in Proceedings of the XIV Colloquium on Musical Informatic (CIM2003) Firenze, Italy, May 8-9-10.

[5] Tarabella L, Bertini G., (2004). *About the Role of Mapping in gesture-controlled live computer music*. U.K.Will, Computer Music Modelling and Retrieval (CMMR2003). Revised Papers, Springer-Verlag, ISSN 0302-9743

[6] Tarabella, L. (2004) *Handel, a free-hands gesture recognition system*, In Proceedings of Computer Music Modeling and Retrieval'04, Alborg University Esbjerg, Denmark

[7] Coker J., (1964). Improvising Jazz, Prentice Hall, Inc,

[8] Wanderly,M. http://www.ircam.fr/equipes/analyse-synthese/wanderle/Gestes/Externe/

[9] Mulder, A. (1994*). Virtual musical instruments: Accessing the sound synthesis universe as a performer:* In Proc. of the First Brazilian Symposium on Computer Music.

[10] Paradiso, J. 1997. *Electronic Music: New Ways to Play*: IEEE Spectrum Computer Society Press. Dec. 1997. pp. 18-30.

[11] Povall, R. 1996. *Realtime control of audio and video through physical motion*: Steim's bigeye: In Proc. Journées d'Informatique Musicale.

[12] Rowe, R. 2001. *Machine Musicianship*. Cambridge: MIT Press. March 2001

[13] Sapir, S. 2002. *Gesture Control of Digital Audio Envirinments*. In Journal of New Music Research, 2002, Vol.31, No 2, pp.119,129, Swets & Zeitlinger

[14] O'Modhrain, M. S. 2000. *Playing by Feel: Incorporating Haptic Feedback into Computer-Based musical Instruments*. Ph.D. Dissertation, Stanford University Press.

[15] Bertini G., Carosi P., 1993, *Light Baton: A System for Conducting Computer Music Performance*, "Interface" (Journal of New Music Research), Lisse, Netherlands, Vol 22 N° 3, pp. 243-247.

[16] Cardini M., Tarabella L., 2002, *Wireless*, Premio Marconi 2002 per l'arte tecnologica , Circolo Artistico e Università di Bologna, Corte Isolani 7/a, Bologna,.

[17] Cardini M., 2003, *Segno elettronico contemporaneo*, International Conferenze on "Tecnologie e forme nell'arte e nella scienza", Università degli Studi di Salerno.

[18] Tarabella L., *Wireless*, Interctive Festival 1999, Columbia University, Merce Cunnigham Studio.

[19] Tarabella L., *Wireless*, (2002) 100th Anniversary Celebration of First Transatlantic Wireless Trasmission, Logan Airport, Boston.

[20] Tarabella L., *Suite for M*, (2002), International Conference on New Interfaces for Musical Expression, (NIME), Dublin, Concert May 25, 2002.

[21] Tarabella, L. Bertini, G., Boschi, G., 2001. *A data streaming based controller for real-time computer generated music*. In Procs of ISMA2001 (International Symposium on Musical Acoustics), Perugia, Italy. pp. 619-622.