Enhancing Sound Design with Conceptual Blending of Sound Descriptors

João M. Martins[1], Francisco C. Pereira[1], Eduardo Reck Miranda[2], and Amílcar Cardoso[1]

[1] AILab, Centro de Informática e Sistemas da Universidade de Coimbra (CISUC)
3030 Coimbra, Portugal
{jpmm, camara, amilcar}@dei.uc.pt
http://ailab.dei.uc.pt
[2] Computer Music Research
University of Plymouth
Plymouth, UK
eduardo.miranda@plymouth.ac.uk
http://neuromusic.soc.plymouth.ac.uk/ccmr.html

Abstract. This paper introduces a new method for sound synthesis using concept description of sounds. Sound descriptions are *blended* to form a new description, which inherits properties from the former entities as well as having an emergent structure of its own. Such blends are then synthesised to become a potentially new sound.
This work applies the system Divago, which is based on a general purpose computational model of Conceptual Blending.

1 Introduction

Computer sound synthesis has become very attractive for a wide range of musicians. Computers are highly programmable and personal computers can run software capable of synthesising sounds in real-time using a wide range of different techniques. Musicians often may not wish to use preset timbres but would rather prefer to create their own instruments. There are, however, a number of ways to implement synthesisers on a computer, and the choice of a suitable synthesis technique is crucial for effective results. Synthesis techniques may be classified into four categories: loose modelling, spectrum modelling, source modelling and time-based approaches.

Loose modelling techniques tend to provide synthesis parameters that bear little relation to the acoustic world. They are usually based entirely upon conceptual mathematical formulae. It is often difficult to predict the outcome and to explore the potential of a loose model. Frequency modulation (FM) is a typical example of loose modelling [1]. FM is a powerful technique and relatively easy to implement, but difficult to operate because the relationship between a timbre and its respective synthesis parameters is not intuitive; for example, increasing the value of a parameter may not necessarily increase the manifestation of its associated sound qualities.

Source modelling and spectrum modelling attempt to alleviate this problem by providing less obscure synthesis parameters; both support the incorporation of natural acoustic phenomena. The fundamental difference between source and spectrum modelling techniques is that the former tends to model a sound at its source, whilst the latter tends to model a sound at the basilar membrane of the human ear. The implementation of a source model (e.g., a physical model) is not straightforward. But once the model is implemented, the user is confronted with relatively intuitive parameters to operate it.

Spectrum modelling techniques have their origins in Fourier's Theorem and the additive synthesis technique. Fourier's Theorem states that any periodic waveform can be modelled as a sum of partials at various amplitude envelopes and time-varying frequencies. Additive synthesis is accepted as being perhaps the most powerful and flexible spectrum modelling method [2]. Musical timbres are composed of dozens of time-varying partials, including harmonic, non-harmonic and noise components. It would require dozens of oscillators, noise generators and envelopes to simulate musical timbres using the classic additive technique. The specification and control of the parameter values for these components are difficult and time-consuming.

Finally, time-based techniques approach synthesis from a time domain perspective. The parameters of time-based synthesis tend to describe sound evolution and transformation of time-related features; e.g. in terms of time lapses. Examples of time modelling techniques include granular synthesis [3] and sequential waveform composition [4]. But again, musicians tend to not use such techniques because it is difficult to determine the role of their parameters with respect to sound quality.

It is clear that some techniques are more intuitive to operate than others, but the most intuitive ones may not be the most appropriate for producing particular types of sounds. From the point of view of the user, however, the problem with sound synthesis is not so much with the intuition of the parameters of the various techniques, but with having the right tools to aid the creative design process.

Sound design is certainly a complex kind of intelligent behaviour. In attempting to solve a sound design problem, composers need to explore possible solutions by trying out possibilities and investigating their consequences. When synthesising sounds to be used in a composition, composers generally have their own ideas about the possibilities of organizing these sounds into a musical structure. In attempting to obtain the desired sound, the composer needs to explore a variety of possible solutions, trying out those possibilities within his or her personal aesthetic. It is the need to provide better support for this exploratory creative process that has motivated our research work.

Sound synthesis systems normally provide good graphic facilities to design the instruments; e.g. visual programming tools such as Max/MSP [5] and Reaktor [6]. However, such systems do not give support for the exploration of the potential of such instruments. The user is often confronted with a visual interface for setting the various synthesis parameters manually. In these cases, the sound design process normally involves non-systematic and lengthy trial and er-

ror practices. We believe that we can improve this scenario by providing Artificial Intelligence (AI) to sound design systems. One approach for doing so is to provide tools for the exploration of synthesis algorithms using high-level conceptual descriptions of sounds, as opposed to low-level parametric specifications.

Early attempts at high-level AI systems for sound design have been proposed by Rolland and Pachet [7], and also by Miranda [8], in his system called ARTIST (ARTificial Intelligence Sound Tool). ARTIST was intended to offer the ability to operate the system in terms of qualitative sound descriptors (e.g., adjectives in English) and intuitive operations rather than in terms of numerical values. The system featured a symbolic representation scheme devised to represent sounds in terms of their perceptual components and relations between them.

Other less AI-oriented attempts at the design of high-level interfaces for synthesisers include [9] [10] [11].

A thorough discussion on the pros and cons of these systems is beyond the scope of this paper. It suffices to say that the main limitation of the system developed by [7] is that it has been designed primarily as an interface for a commercial MIDI keyboard synthesiser manufactured in the mid of the 1990s. As for ARTIST, the robustness of the the knowledge base and inference engine has not been tested on cases combining different synthesis methods. Also, the system does not provide a straightforward solution for dealing with conflicting sound descriptors. Unfortunately neither of these systems have been further developed by the authors.

The present paper proposes a further development of the approach introduced in ARTIST, by taking on board a new computational model for conceptual blending, called Divago. We believe that the concept of conceptual blending is more flexible for representing and manipulating sound attributes than the frame-like representation used in ARTIST.

2 Overview of Divago

Divago is a system that is able to combine (i.e. *blend*) a pair of concepts into a single concept which has a structure of its own. In other words, a *blend* inherits characteristics from the original concepts, but may contain novel characteristics obtained from the process or from a third source (e.g. a rule base, an ontology, a *frame*). Thus it should have *emergent structure*. Divago is a rather large project therefore we will give a general overview of the aspects that are relevant for this paper, leaving out some of its foundations and specificities; readers can find this information elsewhere [12, 13].

2.1 Knowledge Representation

Divago allows several different kinds of knowledge representation (the sound synthesis jargon will be clarified in section 3):

– **Concept maps** describe factual knowledge about a concept. A concept map is essentially a semantic network in which all arcs are binary (i.e. they connect

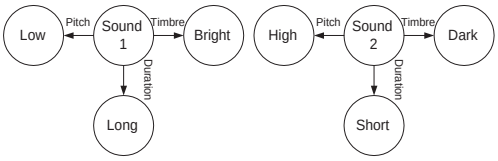

Fig. 1. Concept maps for two sound examples

exactly two different elements[1]). For example, the fact $pitch(sound\_1, low)$ is a characteristic of sound_1 and part of its concept map (see the arc between *sound*_1 and *low* in Fig. 1). In Fig. 3 two concept maps of a flute sound and guitar sound are presented as sources for the blend.

- **Rules** describe inferential knowledge about a concept or a domain. Rules are represented in first order logic format[2]. A possible rule could be "If X is a stringed instrument AND it is not a Piano THEN its partials are harmonic.".
- **Frames** describe abstract concepts or procedures. They can be instantiated by the concept maps (when this happens one says that "the frame has been integrated" or "the concept map accomplishes the frame"). They are formally equivalent to rules (their representation is similar). An example of a simple frame could be "wind instrument". If a concept map about a concept c instantiates this frame, then we can say that c is a "wind instrument" (i.e. it would have the generic characteristics expected for the sound of such an instrument, such as a *blow*, $attack - steady - decay$ sequence, etc.). Frames are extremely important in Divago and they can be seen as information moulds which can be used to *shape* new concepts.
- **Integrity constraints** are simple rules (with *false* consequent) that serve to identify inconsistencies (e.g. a sound cannot have a *crescendo* and *diminuendo* at the same time). These constraints, however, do not imply the elimination of the concepts that violate them, rather they are pressures against the generation of these concepts.

2.2 The Architecture

In Fig. 2, we show the architecture of Divago. The **Knowledge Base** comprises a set of concepts (each normally consisting of a concept map) and a *generic domain*, which has generic background knowledge (e.g. an *isa* hierarchy, a set of frames and integrity constraints). For the work presented here, the concepts of the Knowledge Base must be sound descriptions as in Fig. 1. In section 3.1, we will approach this issue in more detail.

The first step for the invention of a new concept (a new sound in the current context) is the selection of the input knowledge, in this case a pair of concepts (e.g. *sound*1 and *sound*2). Currently, this selection is either given by a user or randomly chosen. The **Mapper** then builds a structural alignment between

[1] In order to avoid ambiguity, we call each node of a concept map an *element*.

[2] A rule has the form $C_1 \vee C_2 \vee C_n \Leftarrow P_1 \wedge P_2 \wedge P_m$, for m premises and n conclusions.

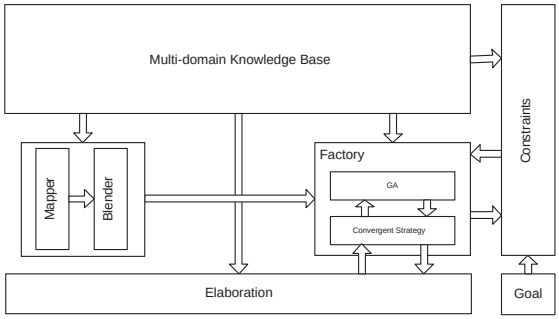

Fig. 2. The architecture of Divago

(the concept maps of) those two concepts. It then passes the resulting mapping to the **Blender**, which then proposes a set of conceptual combinations to be considered, each one corresponding to a *selective projection* from the inputs to the blend. A projection is meant to be the "image" (or the counterpart) in the blend of an element of the input concepts. For example, in the blend of *sound*1 and *sound*2 in Fig. 3, the element *sound*1 gets projected to *sound*2, *bright* gets projected to the blend untouched (*bright*) and *dark* to *bright* (such that, instead of *dark*, *sound*2 will be *bright*). In Fig. 3, we sketch a possible mapping as well as a combination of projections. Notice that not all the elements get projected - a *selective projection* (e.g. *low* in Sound1, *vector*3 in Sound2); an element from the inputs either gets projected to a copy of itself (e.g. *bright* from Sound1), to a copy of its mapping counterpart when it exists (e.g. *dark* from Sound2), or it is not projected at all. Obviously, the number of possible projections is vast for any two input concepts, thus the search space is extremely large. This search space is explored by the Factory module.

The **Factory** is based on a parallel search engine, a *genetic algorithm* (GA), which searches for the blend that best complies with the evaluation given by the Constraints module. Prior to sending each blend to this module, the Factory sends it to the Elaboration module, where it is subject to the application of domain or context-dependent knowledge (in the form of rules and frames found in the generic domain). The GA thus interacts both with the Constraints and Elaboration modules during the search.

The evaluation of a blend given by the **Constraints** module is based on an implementation of the eight Optimality Principles [12], which measure aspects such as *Topology maintenance*, *Frame integration* or *Goal satisfaction*. The **Elaboration** module essentially applies rule-based reasoning (e.g. the application of rules such as the one given in section 2.1). These rules are also part of the knowledge base.

After reaching a satisfactory solution or a specified number of iterations, the Factory stops the GA and returns the best solution achieved, also in the form of a concept map (and with new rules, frames or integrity constraints, in the rare cases in which these structures are also part of the input concepts and of the

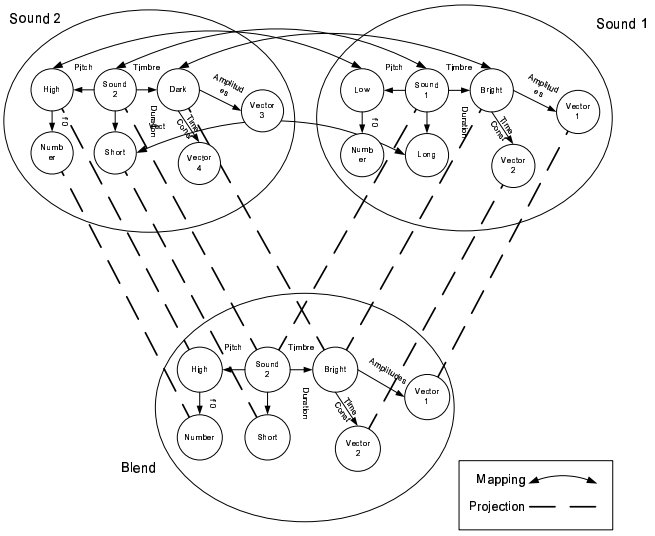

Fig. 3. A blend with its mappings and projections (for the sake of readability, we show only an excerpt).

blend). Thus, the input and output of Divago is expressed in the same syntax and with the same kind of knowledge structures as described in section 2.1.

In some cases, the output of Divago is also the input of an **Interpretation** module, which produces an interpretation of the new concept. In previous versions of this system, we made interpreters for 2D [14] and 3D images [15], as well as textual descriptions of the blend [16]. Of course, these several *modalities* were adapted to specific uses and therefore they are not guaranteed to work in different applications. For the present work, our Interpretation module will correspond to a *Synthesiser*, as will be explained in section 3.3.

3 A Case-Study System

This paper reports an extension of Divago to generate blends of sounds. More specifically, it consists of a knowledge base with sound descriptions, and frames, rules and integrity constraints that are more appropriate to the sound synthesis domain. On the output side, a synthesiser interpreter is being developed as explained in section 3.3.

As a first set of experiments, the concept maps from Fig. 1 were programmed in Divago, as well as integrity constraints for forbidding concurrence of sound states (e.g. a sound cannot be *long* and *short* at the same time). We also gave one frame to the system: $timbre\_transfer$, which expects the blend to have a *timbre* (and its associated vectors) from one input in the context of the other input. When used in the query, this frame values the transfer of *timbres* to a new context. In other words, sounds with the *new timbres* will be preferred by

the genetic algorithm. The blend generated (from a set of 30 runs) is described in Fig. 4.

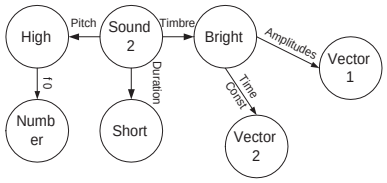

Fig. 4. A blend of sound1 and sound2 generated by Divago

This output thus needs an interpretation, a signal synthesis that results from an unambiguous reading of the blends. In the following subsection we will describe the concepts handled by Divago which will be interpreted by the synthesiser.

3.1 Sound Descriptions

Divago needs descriptions at the concept level, preferably in the form of concept maps. This implies the (always subjective) choice of a language and of abstract level primitives to describe sounds.

Four general characteristics are commonly used to describe sound [17]: *pitch*, *duration*, *timbre* and *loudness*.

All these attributes are subjective, each being dependant on more than one measurable physical characteristic, such as *pressure*, *frequency*, *spectrum*, *envelope*, and *duration*. From these, the least understood are spectrum and envelope. Spectrum is the space where the frequency content of a sound is pictured, and each frequency has a corresponding amplitude. Prominent peaks in spectrum are called partials. Envelope is the time variation of the amplitude (or energy) of sound.

Pitch is the attribute by which sounds can be ordered from low to high, most musical instruments have a defined pitch, except for some percussion instruments. Although pitch depends strongly on the frequency of the fundamental tone, it is also influenced by the intensity of the sound and its high frequency components (spectrum). Loudness is a perceptual measure of the intensity of sound. It depends mainly on sound pressure exerted on the timpani of the ear, but can also be influenced by the spectrum and duration of the sound. The American National Standards Institute (ANSI) defines timbre as "... that attribute of auditory sensation in terms of which a listener can judge that two sounds, similarly presented and having the same loudness and pitch, are different". This attribute identifies the sound source and is the most tricky attribute to quantify as it depends on many different characteristics. Another problem is that there is no uniform set of concepts to identify and classify timbre. Analogies with visual and tactile expressions are often created to suppress this lack of concepts in the

musical domain: sounds can be warm, dark, bright, sweet, metallic, etc. It is certain that timbre shows a strong dependence on spectral components and envelope characteristics. Statistical tests show that the transients of the attack and decay parts are critical for instrument identification [18]. Pollard and Janssen [19] designed a graphic representation method called *Tristimulus*, analogous to that used for mapping colours. In this method the relations between partials are mapped in two dimensions, marking the evolution of the partials in time on a graph, producing a visual representation of timbre. For more detailed description of the perceptual attributes of sound please refer to [20].

To control the previously described attributes we will use additive synthesis, as it is easy to understand and is the basis for more advanced techniques, such as Spectral Modelling. Sounds are synthesised by weighting and adding sinewaves with different frequencies [3] along the spectrum. These sinewaves model the partials of the original sound.

In later experiments we use guitar and flute sounds, dividing them into three distinct parts by inspection of the sound envelope, as shown in Fig.5, resulting in three concepts connected by the concept map: *attack*, *steady state*, and *decay*. We now explain these concepts as Sound1 and Sound2 taking the form of *decay* sounds.

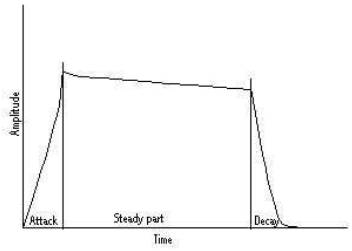

Fig. 5. Envelope model of a musical instrument sound

Attack is the initial part of the sound. It contains noise components from the physical interaction between the player and the musical instrument, as well as resonances from the body of the musical instrument that usually fade away quickly, and the raising partials from the original vibrating source.

Steady state is the signal part corresponding to sound driven by a player. In struck or plucked instruments, sounds do not possess steady parts, as they are not driven by a force or constant blow.

Decay is assumed to be the natural phenomenon of the attenuation of a sound when it is not fed with external energy. Here we keep the same parameters as in steady state, except for duration, which we substitute by exponential factors for each partial. The decay time is the time between the end of the steady state (or attack in case of instruments without this state) and the time point where the signal decays to $1/e$ of its initial value. In reproducing natural sounds it is

Partials		$1(f_0)$	2	3	4	5	6
Sound 1	weight	3.5	3.0	2.0	2.0	3.0	1.0
$f_0 = 110$Hz	Time const. (s)	0.5	0.3	0.5	2.0	2.0	2.0
Sound 2	weight	3.0	0.5	0.5	0	0	0
($f_0 = 240$Hz)	Time const. (s)	1/3	1/6	1/9	0	0	0

Table 1. The weights and time constants of the exponentials of the first six partials of sound examples 1 and 2

crucial to obtain different decays for each partial.The low partials of a string decay more slowly than the higher ones [21].

The experiments carried out with the sounds shown in Fig. 1 lead us to associate two vectors to the concept following the arc *timbre*: amplitudes and time constants of the partials. Although Sound 1 has a low pitch, which is given by its fundamental frequency of 110 Hz, the row vectors from Tab. 1 show that the high amplitudes and long decay times in the high frequencies account for a *bright* timbre. The insignificance of the high frequencies in Sound 2 accounts for its *dark* timbre.

3.2 Other Structures

Frames For the current setting, we use the same general-purpose frames that have been applied in other experiments [13, 22, 15]. The frames $aframe$ and $bframe$ imply the same relational structure as inputs 1 and 2 (resp.), i.e. a blend that integrates these frames will have the same relations as those inputs. The frames *aprojection* and *bprojection* imply the projection of the same elements of inputs 1 and 2 (resp.). In other words, when a blend integrates these frames, the nodes being used come from those inputs. Other frames from previous works could be used and will certainly be subject to experiments, however these four frames are the ones that are context independent and have proven to be fundamental in the construction of blends.

We have also built frames specific to the Sound domain. For the purposes of this paper, the frames for $timbre\_transfer$ as described above and used in the query, for two kinds of sound state sequences ($attack \rightarrow decay$ and $attack \rightarrow steady \rightarrow decay$) and for the two kinds of instruments were coded. Below, we show the coding of the frame *steady_sound*:

$$frame(steady\_sound :$$
$$steady\_sound \longleftarrow after(attack, steady) \wedge after(steady, decay)$$

More knowledge could be included, such as the conditions that should be present for a state to be considered: *attack*, *steady* and *decay*. In table 2, we show the frames that are currently available in the knowledge base of Divago.

Frame name	Conditions
aframe	The blend contains identical structure from input 1
aprojection	The blend contains the same elements of input 1
bframe	The blend contains identical structure from input 2
bprojection	The blend contains the same elements of input 2
timbre_transfer	The blend results from the transfer of the timbre of one input to the context of the other input
steady_sound	The blend follows the sequence of states $attack \rightarrow steady \rightarrow decay$
attack_decay_sound	The blend follows the sequence of states $attack \rightarrow decay$
wind_instrument	The blend contains the characteristics of a wind instrument
plucked_instrument	The blend contains the characteristics of a plucked instrument

Table 2. Some frames of the generic space

Integrity Constraints As the integrity constraints are essentially domain dependent, we add them as the system progresses in each new domain. For the experiments referred to in this paper, we only used three integrity constraints for avoiding sound state concurrence:

$$false \leftarrow duration(X, Y) \wedge duration(X, Z) \wedge Y \neq Z$$
$$false \leftarrow timbre(X, Y) \wedge timbre(Z, Y) \wedge X \neq Z$$
$$false \leftarrow pitch(X, Y) \wedge pitch(Z, Y) \wedge X \neq Z$$

Goals As for the rest of the knowledge structures in Divago, the language of goals allows the same possibilities any Prolog interpreter can offer, which implies that, when submitting a query to Divago, we can use simple pairs of relations and reference to frames or even entire logic programs. Nevertheless, experience has told us that using frames and simple relations is enough to make Divago give us satisfactory results. For example, for the result shown in Fig. 4, the query contained only $timbre\_transfer$.

3.3 Synthesiser

Spectral and Physical models are synthesis techniques that offer us good perspectives to interpret blended sounds [23], but we leave these for the near future, as we have been using the more simple technique of additive synthesis programmed in Matlab, to illustrate the system. At this point we have used the blend created by Divago, shown in Fig.4 to create a sound that maintains most of the characteristics of Sound2, yet having the timbre deriving from Sound1. The Matlab code and sound examples used to generate this examples can be found at our URL [24].

In other experiments we have blended guitar and flute sounds putting emphasis on the temporal division of sound addressed in Sec. 3.1, using *attack*, *steady* and *decay* in a map of concepts. The extraction of the split points between these regions shown in Fig. 5 is not trivial, and although there are some methods

described by Jensen [25] to extract them, we used only visual inspection. The amplitudes of the partials of both the steady state of the flute and the guitar decay, were extracted with an analysis tool developed at the Helsinki University of Technology [26], and have 19 components.

The resulting blend resembles a guitar sound, as it has the same pitch, the same decays and the same spectral distribution. The new feature about it is the existence of a steady state projected by the flute sound, lasting for approximately 2 seconds, with a sampling rate of 22050 Hz.

The blend signal is pictured on Fig. 6. It is important to notice that we are not only creating a new instrument somewhere between the former instruments, but we are also exploring the conceptual description and features of the sound.

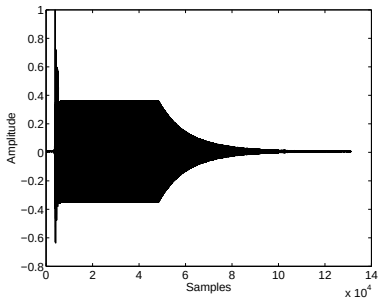

Fig. 6. The result of a guitar-flute blend

4 Conclusions and Further Work

Taking Divago as a basis for a sound synthesis system seems a promising idea as it has demonstrated versatility and creative capabilities in various domains. On one hand, we may test its potential in the domain of sound synthesis, and we may also develop a method for the generation of sounds based on abstract descriptions, designed from perceptual or cognitive perspectives of sound analysis. The only restrictions are that these descriptions should be unambiguous and correspond to Divagos syntax. Currently, we achieve this with a distinct semantics for each element of the concept maps used (e.g. “attack” is interpreted as the attack part of a sound signal). A further goal is the creation of an automatic interpreter to transpose the knowledge from the concept maps to the synthesiser module. Also, the use of Physical and Spectral Modelling remain future goals to explore the concepts associated with the sound production mechanisms.

References

1. Chowning, J., Bristow, D.: FM Theory and Applications: By Musicians for Musicians. Tokyo: Yamaha Music Foundation (1986)

2. Dodge, C., Jerse, T.: Computer Music. New York: Schirmer Books (1985)
3. Miranda, E.R.: Computer Sound Design: Synthesis Techniques and Programming. Oxford (UK): Focal Press (2002)
4. Chandra, A.: The linear change of waveform segments causing non-linear changes of timbral presence. Contemporary Music Review: Timbre Composition in Electroacoustic Music **10** (1994) 157–169
5. Cycling74. http://www.cycling74.com/ (Last visited 6 July 2004)
6. Sasso, L.: Native Instruments Reaktor 3 The ultimate hands-on guide for all Reaktor fans. Bremen (Germany): Wizoo (2001)
7. Rolland, P.Y., Pachet, F.: Representation de connaissances sur la programmation de synthetiseurs. In: Recherches et Applications en Informatique Musicale. Volume 1998. Hermes (Collection Informatique Musicale)
8. Miranda, E.R.: An artificial intelligence approach to sound design. Computer Music Journal **19** (1995) 59–75
9. Ethington, R., Punch, B.: Seawave: A system for musical timbre description. Computer Music Journal **18** (1994) 30–39
10. Garton, B.: The elthar program. Perspectives of New Music **27** (1989) 6–41
11. Schmidt, B.L.: Natural language interfaces and their application to music systems. In: Proc. of the 5th Audio Eng. Soc. International Conference. (1987) 198–206
12. Pereira, F.C., Cardoso, A.: Optimality principles for conceptual blending: A first computational approach. AISB Journal **1** (2003)
13. Pereira, F.C., Cardoso, A.: The horse-bird creature generation experiment. AISB Journal **1** (2003)
14. Pereira, F.C., Cardoso, A.: The boat-house visual blending experience. In: Proc. to the 2nd Workshop on Creative Systems, ECAI'02 (2002)
15. Ribeiro, P., Pereira, F.C., Marques, B., Leitao, B., Cardoso, A.: A model for creativity in creature generation. In: Proc. of the 4^{th} Conference on Games Development (GAME ON'03), EuroSIS / University of Wolverhampton (2003)
16. Pereira, F.C., Gervás, P.: Natural language generation from concept blends. In: AISB'03 Symposium on AI and Creativity in Arts and Science, SSAISB (2003)
17. Rossing, T.: The Science of Sound. Addison-Wesley (1990)
18. Berger, K.W.: Some factors in the recognition of timbre. J. Acoust. Soc. of America **36** (1964)
19. Pollard, H., Jansson, E.: A tristimulus method for the specification of musical timbre. Acustica **51** (1982)
20. Cook, P.: Music, Cognition, and Computerized Sound. The MIT Press (1999)
21. Fletcher, N., Rossing, T.: The Physics of musical instruments. Springer-Verlag New York Inc. (1991)
22. Pereira, F.C.: Experiments with free concept generation in Divago. In Cardoso, A., Bento, C., Gero, J., eds.: Proc. of the 3rd Workshop on Creative Systems, IJCAI-03 (2003)
23. Smith, J.O.: Viewpoints on the history of digital synthesis. In: Proc. Int. Computer Music Conf. (ICMC-91). (1991) 1–10
24. Martins, J.M. http://eden.dei.uc.pt/~jpmm/CC04/ (Last visited 6 July 2004)
25. Jensen, K.: Envelope model of isolated musical sounds. In: 2nd COST G-6 Workshop on Digital Audio Effects (DAFx99). (1999)
26. Välimäki, V., Tolonen, T.: Development and calibration of a guitar synthesizer. J. Audio Eng. Soc. **46** (1998) 766–778