

A Flexible System for Creating Music while Interacting with the Computer

Zeljko Obrenovic

Laboratory for Multimedia Communications
School of Business Administration, University of Belgrade
Jove Ilica 154, 11000 Belgrade, Serbia and Montenegro

Tel: +381-11-3201686, Fax: +381-11-461221

obren@fon.bg.ac.yu

ABSTRACT

Music is a very important part of our lives. People enjoy listening to the music, and many of us find a special pleasure in creating the music. Computers further extended many aspects of our musical experience. Listening to, recording, and creating music is now easier and more accessible to various users. On the other hand, various computing applications exploit the music in order to better support the interaction with users. However, listening to music is generally a passive experience. Although we may change many parameters, the music we listen to generally does not reflect our response, or does so very roughly.

In this paper we present a flexible framework that enables active creation of instrumental music based of the implicit dynamics and content of human-computer interaction. Our approach is application independent, and it provides a mapping of musical features to the abstraction of user interaction. This mapping is based on analysis of the dynamic and content of the human-computer interaction. In contrast to the most existing interactive music composition tools, which require explicit interaction with the system, we have provided a more flexible solution that implicitly maps user interaction parameters to the various musical features.

Categories and Subject Descriptors

H.5.1 [Multimedia Information Systems] ; Audio input/output; H.5.5 [Sound and Music Computing]: Signal analysis, synthesis, and processing; H.5.3 [Group and Organization Interfaces]: Synchronous interaction.

General Terms

Measurement, Design, Experimentation, Human Factors.

Keywords

Interactive music composition tools.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACM Multimedia '05, November 6–11, 2005, Singapore.

Copyright 2005 ACM 1-58113-000-0/00/0004...\$5.00.

1. INTRODUCTION

Music is a very important part of our lives [1]. People enjoy listening to the music, and many of us find a special pleasure in creating the music. Computers have further extended many aspects of our musical experience. Listening to, recording, and creating music is now easier and more accessible to various users. Various computing applications exploit the music in order to better support the interaction with users [2]. Computer games address this issue particularly careful. Some other applications, for example, use the music in order to create an audio context, which can help the users to remember facts [3].

However, *listening to music is generally a passive experience*. Although we may change many parameters, the music we listen to generally does not reflect our response, or does so very roughly. On the other hand, music composition tools still require lots of our attention and efforts. In this paper we present a flexible framework that enables active creation of instrumental music based of the implicit analysis of dynamics and content of human-computer interaction. Our approach is application independent, and it provides a mapping of musical features to the abstraction of user interaction. This mapping is based on analysis of the dynamic and content of the human-computer interaction. In contrast to the most existing interactive music composition tools, which require explicit interaction with the system, we have provided a more flexible solution that *implicitly* maps user interaction parameters to the various musical features.

In next section we discuss existing interactive music creation solutions. Then we present the basic idea of our approach, where we describe architecture, as well as the implementation. After that, we discuss several applications of our solution.

2. MUSIC AND INTERACTION

There has been a tremendous amount of work in the field of computer music, and various types of systems have been developed. In this section we will focus on interactive musical compositional tools, especially on those that, compared with traditional instrument model, offer higher-level interaction. However, if you want to get more complete picture, Joel Chadabe's *Electric Sound* [4], and Curtis Road's *Computer Music Tutorial* [5] are recommended readings.

Having in mind type and activity of interaction between the user and the music system, we can roughly classify interactive musical systems in [6]:

- *The instrument model systems* - rely on constant musical input from the user, similarly to normal instrument. Examples include various application that use devices such as MIDI keyboards.
- *The radio model systems* - operate mostly independently from the user, and the most important interaction is predetermined on a number of discrete alternatives, and all output is pre-composed. Interaction is measured in terms of minutes and hours.
- *High-level interactive music composition systems*, are somewhere in between the instrument and the radio model. They require constant, although much less active, interaction and offer high level musical interaction primitives, such as describing large-scale shape of the piece [7].
- *Implicit interactive music composition systems*, explore how the music can be created by implicitly analyzing users' everyday interaction with the environment, e.g. music is a *byproduct* of this interaction.

The basic idea of our approach is to map abstraction of user interaction onto higher-level musical parameters, so in next two sections, we will describe in more details some examples of high-level and implicit interactive computing composition tools.

2.1 High-level interactive computing composition tools

High-level interactive computing composition tools primarily address the problem of making music composition and playing accessible to children and user who do not have rich musical background [7,8].

For example, the Tod Machover's Hyperinstruments group at the MIT Media Lab (see www.media.mit.edu/hyperins/index.html) focuses on building sophisticated interactive musical instruments for non-professional musicians, students, and music lovers in general. Starting in 1992, this group began to explore ways of extending the concept of hyperinstruments to include amateur users. Earlier projects included Alexander Rigopoulos' Seed Music [9], a real-time interactive system that allowed users to improvise by controlling high-level parameters in the music and, Digital Theremins (1994-95), a series of interactive systems which facilitated music-making for amateurs by using electric field sensing to measure physical gestures.

Among many of their projects, they have recently developed the Hyperscore graphical computer-assisted composition system for users with limited or no musical training, which takes freehand drawing as input, letting users literally sketch their pieces [7]. The fundamental idea of Hyperscore is that anyone can perform two key creative activities without musical training: compose short melodies and describe the large-scale shape of a piece. Hyperscore was also used for the Toy Symphony, a large project involving children, orchestras, and technology. During the course of the project, children have worked with the software to compose pieces for string orchestra, some of which local professional orchestras or string quintets then performed in concert.

2.2 Implicit interactive computing composition tools

High-level interactive computing composition tools offer more expressive interfaces, but they still require explicit interaction and attention of the user. On the other hand, some researchers have been exploring how the music can be created by implicitly analyzing users' everyday interaction with the environment, e.g. how to create music as a byproduct of user everyday interaction.

For example, Healey and Picard developed a system that aids in music selection by incorporating physiological variables that might indicate the user's present mood. They developed a wearable computer that perceives and responds to the wearer's affective state, offering a new kind of perceptual interface. The complete system can capture patterns from many kinds of user behavior [10].

The Viktoria Institute's Future Applications Lab, and the Interactive Institute's PLAY Studi from Sweden developed the Sonic City, a form of interactive music instrument using the city as an interface. Sonic City enables users to create a real-time personalized music by walking through and interacting with urban environments. Paths are considered as *musical compositions* and mobility as a *large scale musical gesture*. They have realized a wearable prototype that senses the user's context and actions when walking through the city, maps this information to the audio processing of live urban sounds in real time, and outputs the resulting music through headphones [11].

However, most of these solutions are hard-coded, limiting a number of ways how they can be used outside the original context, or they sometimes require specialized equipment.

3. A Flexible System for Creating Music while Interacting with the Computer

The basic idea of our approach is to create a flexible framework that enables real-time creation of instrumental music based of the content and dynamics of human-computer interaction. In contrast to the most of existing interactive music composition tools, we have provided a more flexible solution that *implicitly* maps user interaction parameters to the various musical features. Figure 1 illustrates this idea. The key components in our systems are:

- The music composer,
- Various interaction sensors.

The music composer provides a high-level control interface for creation of the music. Interaction sensors monitor user interaction at various levels, and send data to the music composer. We have introduced three types of interaction sensors:

- *High-level interaction sensors*, which analyse the content of user interaction, such as analyzers of content of the Web page which the user currently navigates,
- *Low-level interaction sensors*, which analyse user action at primitive level, such as sensors of mouse movements and typing speed, and
- *Environment and user sensors*, which monitor interaction environment and the users, such as a sensors of movement or presence of the user.

In the following sections we will describe the implementation of these components in more details.

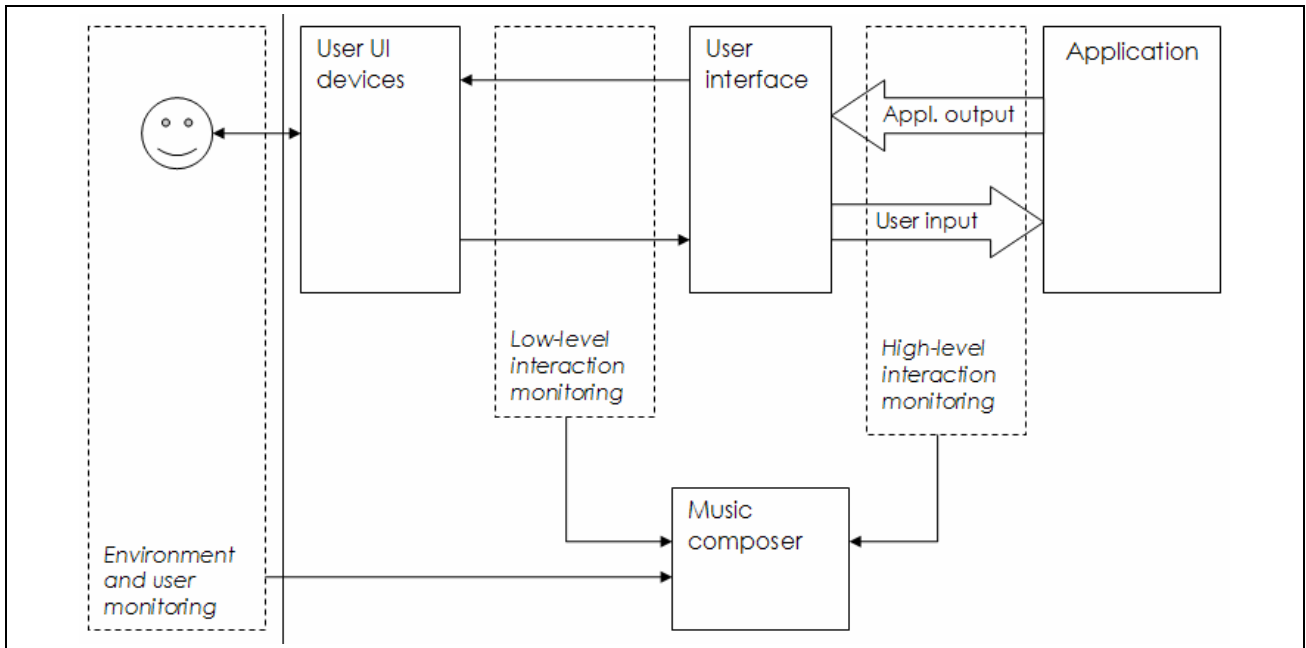


Figure 1. Basic idea of proposed system, composed of the music composer, and various interaction sensors.

3.1 Implementation of music composer and interaction sensors

We have developed a prototype of the described system mostly on Java Platform, but we have used some other developing environment for some of the sensors. The music composer implemented as a Java application.

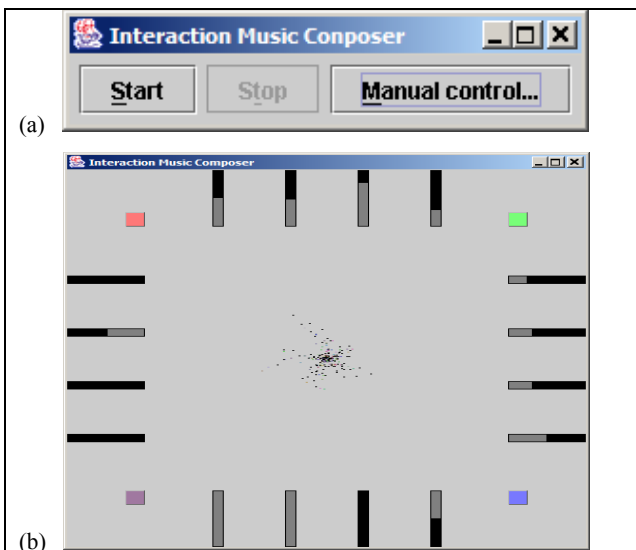


Figure 2. User interface of the Music Composer.

User interface of the Music Composer (Figure 2a) is quite simple, allowing the user to start it, stop it or to take manual control the music. Manual control dialog is not the part of the Music Composer, but it can optionally be provided by Music Composer plug-in. Figure 2b show this dialog for Sharle plug-in, used in our system.

Figure 3 presents the basic components of the music composer. Special modules in the music composer receive and collect these packages, process them and send it to the composer adapter. We have developed several high-level interaction sensors, including various *sensors integrated in Web proxies* that observe user interaction with the World Wide Web (WWW) analyzing the content of Web pages, the size of the page, number of pictures, number of links, and other elements, or analyse of the history of interaction with concrete page. More on these sensors will be described in the section 4.1. For low-level interaction sensors, we have currently provided implementation based on *Windows hooks mechanism*, which enables Windows applications to receive all the events generated in the system, including keyboard and mouse events. More on these sensors will be described in the section 4.2. We have also realized two environment and user sensor: *acceleration sensors*, that detects changes in the acceleration, and can be used to measure movement changes and slope, and *psychological sensors*, including EEG sensor. More on these sensors we give in sections 4.3, 4.5, and 4.6.

Communication between interaction sensors and the music composer is achieved by connection-less User Datagram Packages (UDPs). The sensor are configured with the Internet address and the port of the Music composer. One sensor can send data to more than one Music composer, while one Music composer can receive data form many sensors. Interaction is unidirectional, where interaction sensors send data to the music composer. As music does not have to be exact, but just directed by various interaction parameters, this simple way of communication, has shown to be adequate. Each interaction sensor send UDP packages with a string composed of three parts: a sensor identifier, a parameter name, and a parameter value. For example, a string "web_proxy; page-size; 32566" states that a web proxy sensor detected that a user has just loaded a Web page 32566 bytes big.

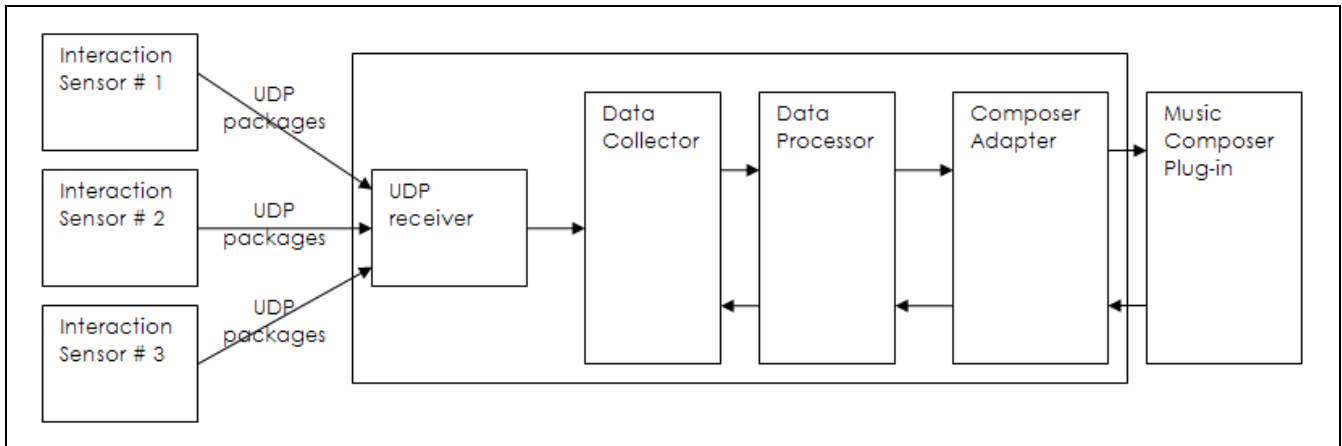


Figure 3. The basic components of the music composer, implemented as a Java application.

The Music Composer plug in, is actually a component that produces the music, offering the interface for shaping the music. We enabled flexible definition of mapping between interaction sensors values and music composer interface. Figure (4) shows a XML fragment of the configuration file which defines mapping of the Web proxy handler page-size parameter to the tempo and rhythm intensity of composition module.

This mapping states that the page size up to 50000 bytes will be linearly transformed into a value from 0.0 to 1.0, while page sizes higher than this will all produce maximal value of 1.0. In case of the tempo, it will change the rhythm linearly between its maximum or minimum, so that pages higher than 50000 bytes will have maximal tempo values, while all the other sizes will produce linearly lower values of the tempo. Rhythm density is changed relatively to its previous value, by adding page-size derived value with 0.1 and increasing the tempo by it.

```

<?xml version="1.0" ?>
- <map>
- <source id="web_proxy">
- <mapping>
  <source-event name="page-size" min="0"
    max="50000" />
  <musical-parameter name="tempo" function="linear"
    scale="0.3" type="absolute" />
  <musical-parameter name="rhythm-density"
    function="linear" scale="1.0" type="relative" />
</mapping>
</source>
</map>

```

Figure 4. Configuring the mapping among interaction parameters and the music composition module.

3.2 Music composer plug-in

Current version enables flexible integration of various music composition components. For each component, it is necessary to develop an adapter module that maps values from 0.0 to 1.0 into values that the module accepts. Our mapping module will ensure that the adapter receives only values from 0.0 to 1.0. Currently, we have used an open-source implementation of Sharle from MIT [6, 12]. This music composition module enables high-level manipulation of the music. The output is based solely on user-controlled parameters and low level rules embedded within the

generation engine. The generator itself is platform independent using MIDI and sound file output to produce music. Table 1 represents parameters that this module accepts.

Table 1. Parameters of the Sharle music composition module.

Cohesion	<i>Maximum repetition at full length, maximum variation at shortest length</i>
Key	<i>Changes the key (value modulo 12).</i>
Scale	<i>Six distinct settings. a major, a more consonant major, a more consonant minor, a minor, a pentatonic, and a more minor pentatonic scales.</i>
Tempo	<i>The shorter it is, the faster the tempo.</i>
Rhythm Consistency	<i>The longer this control is, the more regular the rhythm becomes.</i>
Rhythm Density	<i>The shorter this control is, the sparser the rhythm.</i>
Rhythm Length	<i>The longer this is, the longer the rhythmic period.</i>
Consonance	<i>The shorter this is, the more dissonant the current layer becomes.</i>
Pitch Direction	<i>The longer this is, the more the notes of the current layer will go up in pitch.</i>
Instrument	<i>Changes the instrument. The results depends on the instruments available.</i>
Active	<i>Zero turns the current layer off. One turns it back on.</i>
Rhythm Style	<i>(0) a steady beat, (1) between the beats of the normal rhythm, (2) a rhythm of a different period, and (3) no transformation.</i>
Rhythm Style Argument	<i>Various effects on the beat.</i>
Change / Rate of Mutation	<i>The longer this is, the more the controls will mutate themselves.</i>
Pitch Center	<i>This higher this is, the higher the pitches of the current layer.</i>

Besides working on integration of several other high-level interactive composition tools, we are also working on the design of the MP3 DJ music chooser, which would use interaction data to choose the next song, based on metadata about the song.

4. Applications

In this sections we will present some of the applications of our framework. Here we will present basic ideas, and discuss how creating music in proposed application can improve human

computer interaction. We will present five applications in more details. Table 2 gives a comparative overview of these applications, describing in short terms the main source of data, activity which the user do in order to affect the music, elementary and received data received from the source, and some example of music parameter that were affected with these data.

Table 2. Interaction Music Application.

Application	Source	Activity	Elementary data	Derived data
Browse the Music	Proxy server	Browsing the Web	Web page content Web page metadata	
Type the music	Windows hook mechanism	Typing and pointing in any application	Keyboard events Mouse events	User activity Keyboard patterns Mouse movement patterns
Drive the music	XY accelerator sensor	Using a computer on the unstable platform, such as a car.	X acceleration Y acceleration	Acceleration Drive stability
Mind the Music	External EEG device with drivers	EEG brain waves	EEG waves EEG spectrum	Hemisphere activity Stereo mapping
Join the Music	Other HCI Music application	Any of the master, none of the clients	The master routes all his events to the client	

4.1 Browse the Music

In this application, we wanted to change the musical parameters based on the user interaction on the World Wide Web (WWW). This application is primarily based on monitoring high-level interaction, e.g. the content and metadata about pages that the user visits. Figure 5 shows basic architecture of this application.

4.1.1 Music proxy handler

We have used several open-source Java proxy servers, such as Paw (pro-active webfilter) [13] and Muffin WWW filtering system [14], with proactive filtering, which allow plug-in integration of custom filters and handlers. These filters and handlers can monitor and even modify all the data requests and response parameters. Figure 6 gives a simplified Java code of one of these handlers. This code sends a size of current page to our music composer. This is not a typical example, but we included it here due to simplicity and to illustrate basic principles, as all other sensors are more complex to be included in the text. We implemented several of these filters and handlers to send UDP packages to Interaction Music Composer with based on:

- Analysis of the content of the page, data are sent when some keywords are found.
- Analysis of the size of the page, number of pictures, number of links, and other elements
- Analysis of the history of interaction with concrete page.

We have also made some custom modules on the side of our Web sites, adding a service that analyses the Web log and tells us how many people have visited some page. In this way it is possible to, for example, produce lively and louder sounds for more visited sites, adding important social cues to user interaction [15]. We have also used, the Amazon Alexa Web Information Service [16]. This service offers a platform for creating Web solutions and services based on Alexa's vast information about web sites, accessible with a web services API, including rank, traffic data, categories, related links and so on. It is currently free available during the beta period.

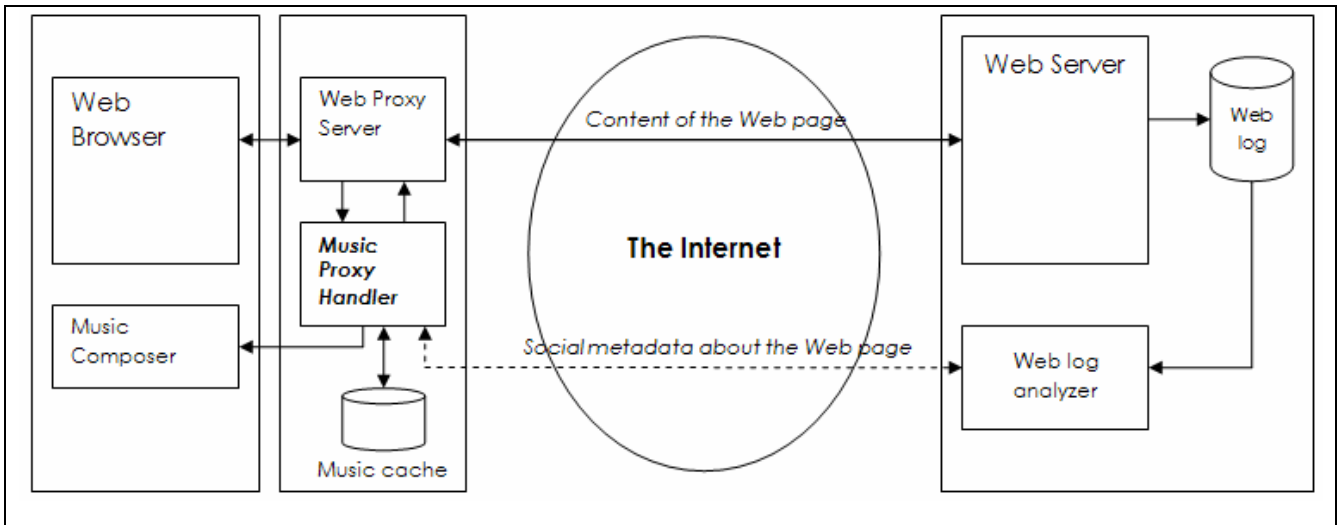


Figure 5. Basic architecture of 'Browse the Music' application.

```

package myhandlers;
....
import sunlabs.brazil.server.*;
import sunlabs.brazil.util.http.*;
import sunlabs.brazil.filter.*;

public class MusicFileSizeHandler implements Filter {
...
    public byte[] filter(Request request, MimeHeaders headers,
        byte[] content) {
        try {
            long duration = System.currentTimeMillis()
                - request.startMillis;
            sendMessage( request, content.length );
            ...
        } catch (Exception e) { ... }
        return (content);
    }

    public void sendMessage( Request request, int size ) {
        String message = "web_proxy;page-size;" + size;

        try {
            DatagramSocket socket = new DatagramSocket();
            byte[] buf = new byte[256];

            DatagramPacket packet = new DatagramPacket(
                message.getBytes(), // content
                message.getBytes().length, // content size
                address, // address of the client
                4445); // port
            socket.send(packet);
            socket.close();
        } catch (Exception e) { ... }
    }
...
}

```

Figure 6. Simplified Java code of our interaction sensor

implemented as a Paw proxy server handler.

4.1.2 Music Cache

The Music Cache is a special mode of this application. The basic idea of musical cache is to remember the music or instrumentation parameters played to the user during his/her last visit, and to play that melody when the user visits the same page again. Currently, we are recording just the music parameters, not the actual music produced. The main motivation for introduction of the musical cache is to create an auditory context, which could help the users remember things, such as navigation structure of some site or Web page. For example, as a part of the DARPA Augmented Cognition project, Tan et al explored building and evaluating computer system interfaces that make information memorable [17]. They implemented a multimodal prototype system, the Infocockpit (for "Information Cockpit") [3]. They reported a user study demonstrating a significant (56%) increase in memory for information presented with the Infocockpit system as compared to a standard desktop system. However, experimental evaluation of this hypothesis in the Music Cache will be the subject of future work.

4.2 Type the Music

Windows Hook mechanisms, allows applications to register for any of hundreds of events in the Windows operating systems, and to receive notification each time when some of these events occurs. It is also possible to monitor the entire user interaction, for example, by receiving events about keyboard and mouse actions. In this way, it is possible to get rough measure about user's activity.

We created a simple Visual C++ interaction sensor that monitors keyboard and mouse activity, sending UDP packages to the Interaction Music Composer. It is possible to send data about average number of keyboard hits and mouse movements. It is also possible to send an event when some special key is hit. For example, if the user hits the F1 key, which is usually associated to help, music could change. We experimented with various types of mapping between these events and musical parameters.

4.3 Drive the Music

For this application we have used a low cost Analog Devices' ADLX202 device. This device has two orthogonally positioned acceleration sensors. 2-axis accelerometers with a measurement range of ± 2 g can measure both dynamic acceleration (e.g., vibration) and static acceleration (e.g., gravity). The outputs are digital signals whose duty cycles are proportional to the acceleration in each of the 2 sensitive axes. These outputs may be measured directly with a microprocessor counter, requiring no A/D converter or glue logic. The device can be connected with the PC with RS232 serial connection. We have implemented the sensor as a Java application, using Java Comm package to communicate with the device (Figure 7).

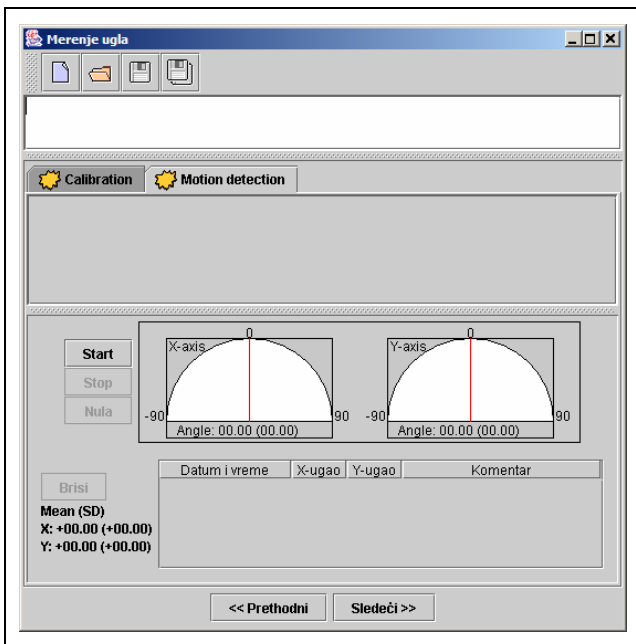


Figure 7. User interface for calibration and testing of ADLX202 device.

We used this sensor with a laptop in a vehicle in order to produce the music based on the parameter about the drive. We have two input channels, one sensor positioned in the direction of vehicle movement, roughly indicating speeding up and slowing down, and orthogonally positioned sensors that could give us data about going left or right. First channel was used to direct the rhythm and tempo of the music: if there is more accelerating the ride is good, and the music can be more dynamic. Going left or right was used in the opposite way, as it usually indicates that the road is not easy for driving.

4.4 Join the Music

IMC does not have to receive only the data from one user, but from many users interacting on different computers. For example, in computer classroom or Internet café, the ambient music could be directed with summary interaction of many people who are working on computers in a same room. Sensors on client computers could be configured to send data to the Music Composer at the server machine (Figure 8a).

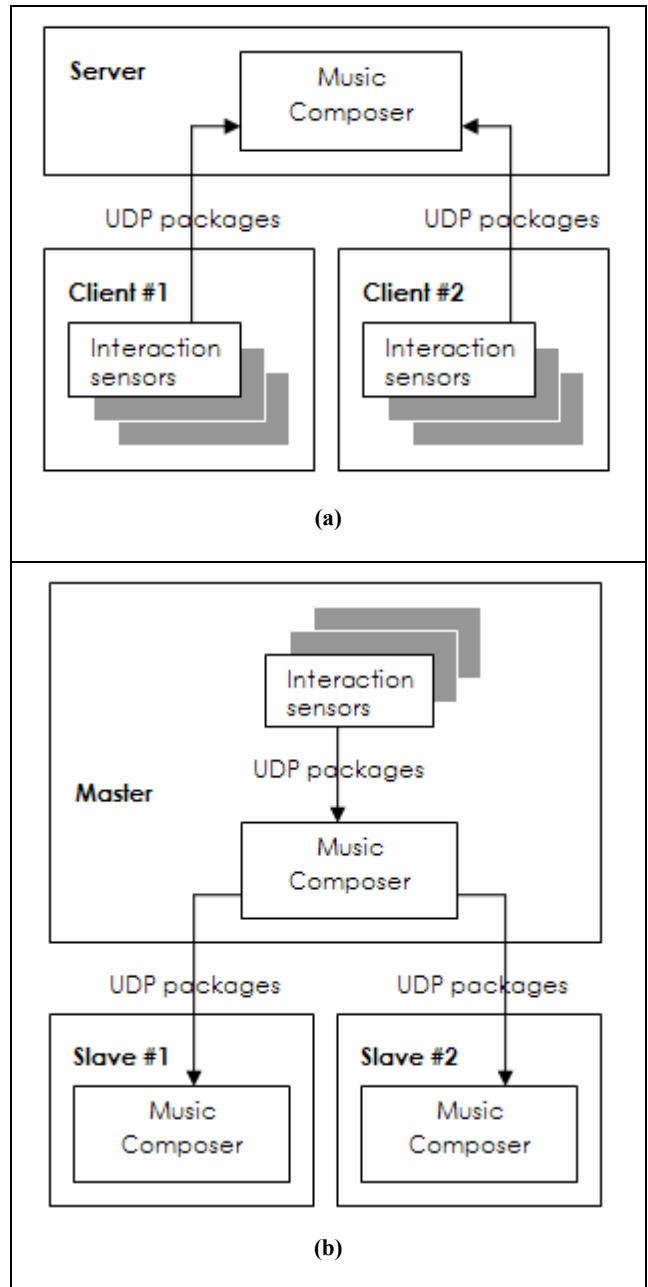


Figure 8. Two modes of synchronized music creation. In first mode (a) sensors from various users on different machines can send their data to through the network to one music composer. In second mode (b), one music composer copies its data and sends it to other music composers.

We also enabled that IMCs can communicate with other IMCs. For example, one of the IMCs would be the master, sending his parameters to all IMCs who have registered (Figure 8b). The music of these IMCs would not be the same, but it would be directed in the same way, e.g. the global structure would be the same. For example, you could listen to the music produces by the interaction of one of your friends.

4.5 Mind the Music

The basic idea of this approach is to map EEG scores [18] to the music composition parameter. Electrical brain activity and music have lots in common.

- Firstly, electrical brain activity is rhythmic, e.g. it goes from left to the right with a frequency of about 50 Hz
- Various activities activate different areas of the brain, so it is possible to roughly see the level of activity of the user's brain.

Figure 9 shows the structure of the EEG music system. Sensing hardware detects EEG electrical waves, and converts them into a digital form. Various software modules then further process raw digital data. At the first level, we calculate various simple derived values such as a mean value, or power spectrum based on the FFT analysis. At the second level, these raw and derived data are combined based on custom scripts to produce various EEG scores, such as the activity of hemispheres, which are then sent to the music composer to change the music settings. This sensor is realized in Visual C++, using standard Windows API, as well as FFTW software library for FFT analysis [19].

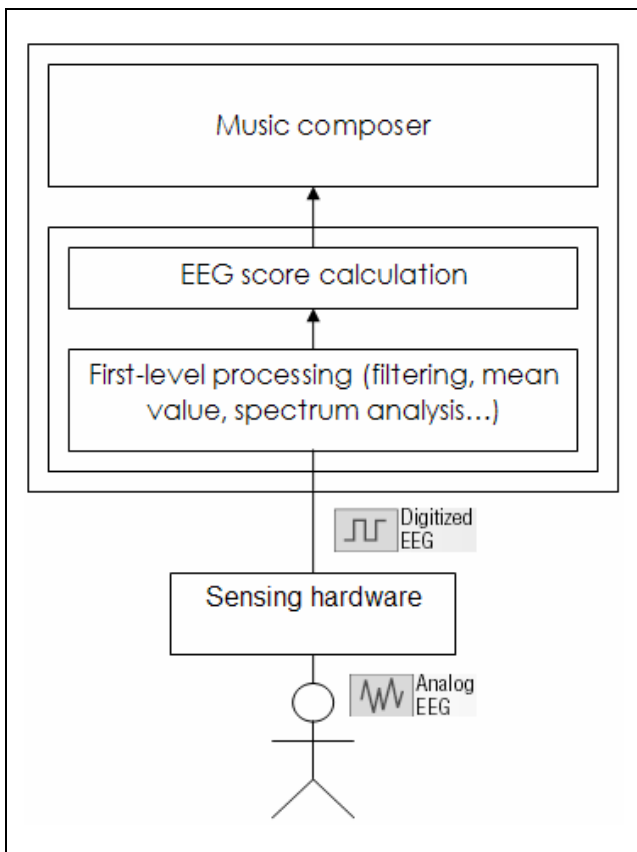


Figure 9. Simplified structure of the EEG music system.

5. CONCLUSIONS

In this paper we have presented a flexible framework that enables active creation of instrumental music based on the implicit analysis of dynamics and content of human-computer interaction. This mapping is based on analysis of the dynamic and content of the human-computer interaction. In contrast to the most existing interactive music composition tools, which require explicit interaction with the system, we have provided a more flexible solution that implicitly maps user interaction parameters to the various musical features. Our approach is application independent, and it can be used with various applications. First experiences have shown that users really enjoyed using the software, and they especially liked the fact they were implicitly involved in the music composition loop.

Proposed solution is a flexible platform that could possibly be useful for various user groups. End users could get interactive, original and always new music adapted to their current computing activity. Internet media broadcasters could also connect user sensors with their program, for example, adding the ability to choose the next song based on the analysis of user current interaction.

In our future work we plan to extend our work in several directions. Firstly, we will try to train the system so that the user could change the music parameters manually, and the system could analyse the conditions before user decision, in order to find a pattern of music parameter switching. We also plan to add more intuitive interface for controlling the mapping among sensor values and music composer parameters.

REFERENCES

- [1] Marvin Minsky, "Music, Mind, and Meaning", *Computer Music Journal*, Fall 1981, Vol. 5, Number 3;
- [2] Buxton, W. (1995). *Speech, Language & Audition*. Chapter 8 in R.M. Baecker, J. Grudin, W. Buxton and S. Greenberg, S. (Eds.) (1995). *Readings in Human Computer Interaction: Toward the Year 2000*, San Francisco, Morgan Kaufmann Publishers.
- [3] Desney Tan et al, "The Infocockpit: Providing Location and Place to Aid Human Memory", *Percaptual User Interfaces - PUI 2001 Workshop*, 2001.
- [4] Joel Chadabe, *Electric Sound: The Past and Promise of Electronic Music*, Prentice Hall, 1996
- [5] Roads, Curtis, et al. *The Computer Music Tutorial*. Cambridge, MA: The MIT Press, 1996
- [6] Chong (John) Yu, "Computer Generated Music Composition", MSc Thesis, M.I.T., 1996.
- [7] Morwaread M. Farbood, Egon Pasztor, and Kevin Jennings, "Hyperscore: A Graphical Sketchpad for Novice Composers", *IEEE Computer Graphics and Applications*, January/February 2004, pp. 50-54.
- [8] Allison Druin, Juan Pablo Hourcade, "Interaction design and children: Introduction", *Communications of the ACM, SPECIAL ISSUE: Interaction design and children*, Volume 48, Issue 1 (January 2005), pp. 33-34..

- [9] Alexander Rigopoulos. "Growing Music from Seeds: Parametric Generation and Control of Seed-Based Music for Interactive Composition and Performance." Master's thesis, MIT Media Lab, 1994.
- [10] Jennifer Healey, Rosalind Picard, Frank Dabek "A New Affect-Perceiving Interface and Its Application to Personalized MUSIC Selection", PUI 98 Workshop.
- [11] Lalya Gaye, Lars Erik Holmquist and Ramia Mazé, "Sonic City: The Urban Environment as a Musical Interface", Proceedings of NIME 2003 New Interfaces for Musical Expression, McGill University, Montreal, Canada, May 2003, pp. 109-115.
- [12] Sharle Source Code Web Page, <http://home.comcast.net/~chtongyu/sharle/>, Last Visited May 22, 2005.
- [13] Paw (pro-active webfilter) Web Site, <http://paw-project.sourceforge.net/>, Last Visited May 22, 2005.
- [14] Muffin WWW Filtering System Web Site, <http://muffin.doit.org/>, Last Visited May 22, 2005.
- [15] Judith Donath, "A Semantic Approach to Visualizing Online Conversations", CACM, Vol. 45, No. 4, April 2002, pp. 45-49.
- [16] The Alexa Web Information Service, http://pages.alexa.com/prod_serv/WebInfoService.html/, Last Visited May 22, 2005.
- [17] Larson, Kevin, Czerwinski, Mary, "Web Page Design: Implications of Memory, Structure and Scent for Information Retrieval", In Proceedings of CHI '98, Human Factors in Computing Systems (LA, April 21-23, 1998), ACM press, pp. 25-32.
- [18] Jovanov, D. Starčević, A. Samardžić, A. Marsh, Ž. Obrenović, "EEG analysis in a telemedical virtual world", Future Generation Computer Systems 15 (1999), pp. 255-263.
- [19] Z. Obrenovic, D. Starcevic, E. Jovanov, V. Radivojevic, "An Implementation of Real-time Monitoring and Analysis in Telemedicine", Third IEEE EMBS Information Technology Applications in Biomedicine – Workshop of the International Telemedical Information Society ITAB-ITIS 2000, Arlington, Virginia, Nov 2000, pp. 74-78.