Internship report

Around TransISTor

18th of June - 20th of July 2007

Prague, Czech Republic

Supervisor : Pavel Smetana

Lévy Benjamin

Deuxième Année

Acknowledgement

I would like to express my gratitude to all those who gave me the possibility to achieve this internship.
I want to thank in the first place Pavel Smetana, director of the CIANT who received me in his Centre and Lab and gave me the great opportunity to work freely on a subject I like very much. I am deeply indebted to the teacher of ENSEA, Sylvain Reynal, thanks to whom I got this very interesting internship.
I also want to thank every member of the CIANT team who welcomed me in their work, interests and projects. I have furthermore to thank especially Aurélie Besson who really took me in during my whole time in Prague.
Finally I want to thank all the people from every corner of Europe I met in Prague, they shared with me a little of their experiences, ideas and energy. Thank you very much.

Contents

Introduction

When I arrived, I was expecting to be assigned on helping the other engineers of the CIANT on video or 3D animation programming for the session II and III of the TransISTor workshops. But as I introduced myself as interested in learning video or 3D animation but already familiar with audio and music technologies, my supervisor – who did know what is possible with motion capture and music – asked me to work on my own and try to produce something fitting to be seen during the session II of TransISTor with their motion capture system used on a musical aspect.Then I had two weeks to handle their system and use it.

There were two main parts in this first mission. The first part was to discover and understand the motion capture system named Polhemus and to interface it with the software I am used to program for two years now, to create computer assisted music named Max/MSP. The second part was to discuss with the dancer and decide what musical objects to program ; and to have finally something working and the more (artistically) interesting as possible to present.
During the third week, the week of the TransISTor Session II, I had to finish the building and testing then perform this Music-Dance interaction in front of the participants of TransISTor. During this week I was in permanent contact with people from several European countries, participants and lecturers.
As a member of the CIANT team, I also gave a lecture about the software (Max/MSP) I used for the real-time audio interaction with dance. In about one hour, I had to give a little idea to the participants of what this software is, what is possible with it and how it works.

During the session III (fourth week of my practice), I attended the lectures and had to prepare a little demonstration of what is possible with the development-cards Arduino and Max/MSP.

Finally, the fifth week of my practice was devoted to finalizing everything I have done : porting my programs on Windows XP, cutting the video of the performance of the Session II, gathering every documentation and source code etc in order to deliver a DVD with all my work to the CIANT. I also attend some other projects meetings such as **CASPAR**.

Part I

The CIANT

The acronym CIANT stands for " Centre International pour les Arts et les Nouvelles Technologies ". The CIANT was found in 1998 by Pavel Smetana (actual director) and Pavel Sedlák (actual deputy director) in Prague. It is the bigger structure dealing with Art and new technologie in Czech Republik.

Chapter 1

Status & Structure

1.1 Aims

The CIANT is an independent non-profit organisation that initiates and supports partnerships with individuals as well as institutions having a background intersecting art, science and technology. It operates an art laboratory focused on the innovative experiments with development of interactive media. It produces and co-produces artistic projects, coordinate residencies for artists and participates in the promotion of new media art and culture. It organises festivals, workshops and conferences. The CIANT stimulates networking and participation in public debates while promoting interdisciplinary approach in the field of artistic and research creativity as well as technological innovation.

1.2 Facilites

The main facilites of the CIANT, based in the city quarter of Žižkov in Prague are organised around two main desks, a laboratory and a gallery. Another gallery downtown Prague belongs to the CIANT. The two desks are the director's desk and the common desk where all administrative works, meetings and paperworks are being done mainly by a secretary and a production manager. In the lab where four to eight people are developing and programming on several computers is done the main part of the technical research around art and new medias, mostly in video and 3D world and animation. In the gallery, a block away from the lab and desks building, are taking place some exhibitions and conferences in the field of research of the CIANT.
I personnaly attended four presentations of research and artworks and saw two exhibitions of the gallery.

1.3 Working at the CIANT

As the CIANT is a small stucture, the work is organised around projects everyone is aware of. One or two persons are responsible for each project. These projects are fully discussed and progress are reported to the director and all the CIANT team in a weekly meeting on Monday morning gathering everybody working for the CIANT.

Chapter 2

Team

2.1 Directors

Pavel Smetana is the CIANT director. He has managed European projects supported within CULTURE 2000, MEDIA Training and the 5th Framework Programme (IST). He was an expert on New Technologies and Arts at the Council of Europe. In 1996 he co-organized a Prague conference entitled " A New Space for Culture and Society " and directed the ENTERmultimediale festival (2000, 2005). He has been heading the department of VR-3D-AL at the École Superieure des Beaux Arts in Aix-en-Provence. He is an artist in the field of virtual and mixed reality, interested in the development of interface technologies. He is the author of several known video and interactive installations, e.g. *Room of Desires* or *The Mirror*, some of which were awarded prizes such as Grand Prix de Locarno or UNESCO Prize.

Pavel Sedlák is the Deputy Director of CIANT in charge of development and documentation. He is active as an art critic and a curator. He initiated several European cultural activities and projects. He was the chief curator of the ENTERmultimediale2 festival of digital art held in Prague in May 2005. He publishes in the field of new media art and culture. After his residencies at the ZKM - Center for Art and Media in Karlsruhe and the internship at New Media Dept. of the National Gallery in Prague he worked at the Centre for Science, Technology, Society Studies of the Institute of Philosophy of the Czech Academy of Sciences. Recently, he was a member of the Prague Mayor's Committee on the city cultural strategy, a jury member of the CYNETart-06 festival held in Dresden, and an invited speaker at the 40th congress of AICA (International Association of Art Critics). In 2006 he co-founded C2C – Circle of Curators & Critics – which runs a Prague-based gallery focusing on interdisciplinary projects.

2.2 Engineers

Michal Máša was doing a PhD at Czech Technical University in Prague (CTU), where he graduated with a MSc in Computer Science in 1999. His PhD topic focused on multi-user collaboration in virtual environments and he has published several papers at various international conferences including Eurographics. His areas of interest are distributed systems, virtual reality and motion capturing.

2.3 Artists in residence

Ivor Diosi majored in information security, and wandered through courses on information systems management, philosophy and cinema history. He has been exhibiting since 1999 and received several awards and recognitions at international new media art festivals (ACA Tokyo, Vida 4.0, ZKM Constructed Life, and Transmediale 2004). Currently, he is an artist-in-residence at CIANT. He is best known for his game engine modifications, and has worked with virtual reality, artificial life and human interface design to produce art, which in his words " strives to be detached from socially conditioned perception and simian legacy, as much as possible, and focus on what really matters: the human being as autonomous conscience in the unvoid ".

Stephane Kyles is an artist in residence.
" I initially engaged in exploration of a certain esthetic of chaos which, in its least overflows, always put forward the real brittleness of the human body related to the current technological world. My works gave rise to many parts, sometimes unfinished, sometimes arranged, of which the aspects releve always of this post-punk radical esthetics which, today still, frightens and fascines at the same time. Parallel to this activity, I continue the development of several data-processing devices whose principle of operation is based on the data flows produced by Internet or all other sources of digital information. From this information, I propose to give body by the image, sound and the gesture has a virtual character from which the behavior, random, is freed in space. I thus try to manufacture body with the impalpable. I remain sensitive to the universe of *cyborgs* like has the revolution of computer and networks. My work, based on collective intelligence, tries to analyze the deviating practices likely to develop has great scale since the users of new technologies seem free to divert them. "

2.4 Other members of the team

Aurélie Besson : Production and events manager
Ivan Klíma : Engineer
Jan Šebek : Technical manager
Denisa Kera : Cultural manager
Sylvie Milerová & Kateřina Poláčková : Assistants

And a lot of others taking part of the work of the CIANT temporarily or permanently (practice, external engineers, artists...)

Chapter 3

Projects

Amoung the numerus projects the CIANT is taking part of, I have been involved in two particular project. The first one, TransISTor will be develop in the second part of this report.

The second one I have attended is the **CASPAR** project.
" CASPAR – Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval – will address the growing challenge facing society of a deluge of intrinsically fragile digital information, upon which it is increasingly dependent, by building a pioneering framework to support the end-to-end preservation *lifecycle* for scientific, artistic and cultural information, based on existing and emerging standards.
The ambitious challenge to build up a common preservation framework for heterogeneous data and a variety of innovative applications will be achieved through the following high level objectives :

- to establish the foundation methodology applicable to a very wide range of preservation issues. The guiding principle of CASPAR is the application of the OAIS Reference Model
- to research, develop and integrate advanced components to be used in a wide range of preservation activities. These components will be the building blocks of the CASPAR framework
- to create the CASPAR framework : the software platform that enables the building of services and applications that can be adapted to multiple areas

[...] "

Seventeen participants are involved in this project, amoung which figure some United Kingdom Universities, European Space Agency, UNESCO, Institut National de l'Audiovisuel, IRCAM, CNRS...
The goal of the meeting I attended was to sum up the state of the project between all the participants and have a critical look upon the already achieved parts in order to refocus and think around the main purposes of CASPAR.

Part II
TransISTor

TransISTor – Transdisciplinary Training in Information Society Technologies and Storytelling Media Creation – is a training initiative organised by CIANT in cooperation with FAMU – Film and TV Faculty of the Academy of Performing Arts in Prague – and New media studies at Charles University supported by the MEDIA Training Programme of the European Union.

The participants of these training are mostly student or recently graduated in movies or media academies from every country of the Europe Community. The lecturers are engineers, artists or researchers from all over Europe and from the CIANT.

For this third year of TransISTor, there were three sessions of three or four-day workshops. They were focused on computer games technologies while opening up their creative potential for non-gaming storytelling domains including art, cinema, TV, educational applications and cross-media productions.

I have been involved in the second and third sessions as a member of the CIANT team to organise, develop on the computer programs and help on the technical part of it. I also gave a lecture during the second session about the software I mostly used, Max/MSP®.

Chapter 4

Session II

4.1 Brochure

Motion capture and stereoscopy for games and film

During the intensive 4-day workshop you will investigate different motion capture techniques and carry out a collaborative project. Experienced tutors will explain the different motion capture systems, the process of setting sensors, recording sessions and the computer data post-processing. The goal is to experience the latest technologies, to integrate them and to create your own stereoscopic film with special effects.

07/03 - 07/06/2007 - Karlovy Vary during the International Film Festival

Schedule :

3rd of July
9:00 - 12:00 AM : Introduction into motion capture
1:30 - 05:00 PM : Workshop on stereoscopy and special effects

4th of July
9:00 - 12:00 AM : Motion capture and animation: case studies and practice
1:30 - 5:00 PM : VRML and workshop on the basics of motion capture

5th of July
9:00 - 12:00 AM : Motion capture and 3ds max
1:30 - 5:00 PM : Workshop on stereoscopy and motion capture I. (FAMU and CIANT workshop)

6th of July
9:00 - 12:00 AM : Motion capture and game engines
1:30 - 5:00 AM : Workshop on stereoscopy and motion capture II. (FAMU and CIANT workshop)

4.2 Course

This Session II of workshops took place in a hotel up the hill of Karlovy Vary (Karlsbad, near the german border in western side of the czech republic). This location has been chosen for the two last year of TransISTor because the date matches with the International Film Festival of this town. The principle of the session was to give some lecture (passiv learning) during each morning and to make the participant use afternoons the presented technologies to produce a short movie by group of three or four. Four group were constituted and the following steps were for them to complete :

- building a scenario and a storyboard,
- choosing and shoot an outside décor the action taking place there with the stereoscopic technology,
- acquiring the data in 3D Studio Max® and match the movement of the camera inside the software (to prepare the inlaying of a virtual character),
- recording the movement of the character with the motion capture system
- building their 3D virtual character
- applying the recorded movement to the character and inlay in the décor
- cuting and finalizing.

Except some lectures given by members, the CIANT team, as organizing team, was there mostly to provide the technological means and knowledge to make the achieving of these short movies possible. With computers, sensors, network, screens, projectors... supplyed by the CIANT, two spaces were created in the main conference room of the hotel. A classical lecture space and a " motion studio space " with the motion capture system, computers and a dance space for the movement to be recorded (or used in realtime). The motion capture system was set up on a dancer especially come.

Chapter 5

Session III

5.1 Brochure

Serious games, web 2.0 and future cinema

This session is focused on the alternative forms of games and web 2.0 applications that are used for entertainment as well as for non-entertainment purposes such as art, edutainment, marketing, simulations, management and public policy. We will look closer at how these technologies are used for non-gaming events and processes, including simulations of business and military operations, psychotherapy or medicine. In the workshop you will create your own serious game scenario and experiment with web 2.0 applications, e.g. Google map mashups in art, business and cinema.

07/09 - 07/12/2006 - Prague

Schedule :

9th of July : Serious game and art
9:00 - 12:00 AM : Artistic Game Development: Tools, Techniques and Trends
1:30 - 05:00 PM : Workshop on serious games and art modification of games I

10th of July : Case studies of serious games
9:00 - 12:00 : Europe 2045 : an on-line collaborative computer game about " future of Europe, immigrants & subsidised cows ".
Serious games and military simulations : Armed Assault
1:30 - 5:00 PM : Workshop on serious games and art modification of games II

11th of July : New strategies and tools for serious/art games
9:00 - 12:00 AM : Samorost, case study of an art game.
Case studies and workshop on Google Earth
1:30 - 5:00 PM : Ultramundum workshop

12th of July
9:00 - 12:00 AM : Case study of Arduino Earthwalk instalation
1:30 - 5:00 PM : Arduino workshop

5.2 Course

This session III of TransISTor, set in the FAMU building was more oriented to lecture and examples of what is possible and made with the new web 2.0 technologies and games. The participants (some of them attendant session II and even session I) were listeners except for two workshops on Google Earth and Arduino.

Part III

Works

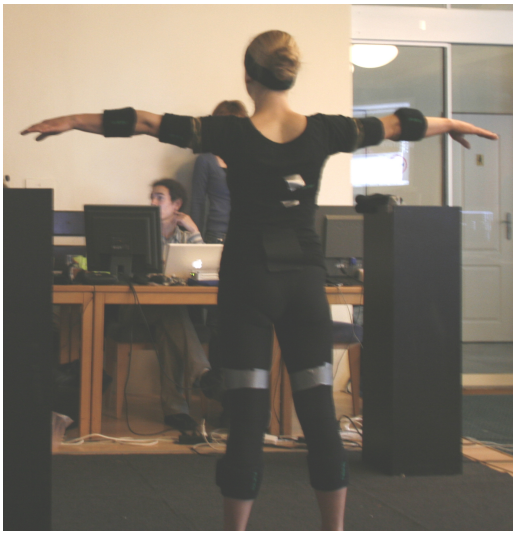
Chapter 6

Music-Dance interaction

6.1 Motion Capture : Technical point of view

6.1.1 The Polhemus System

The *Liberty LATUS (standing for Large Area Tracking Untethered System)* produced by **Polhemus** is an electromagnetic system of motion capture. The principle is simple : some little sensors (three small coils in the three directions) are placed on the body and surrounded in a large and powerful magnetic field especially created and calibrated. Each sensor is sending 6 coordinates (three of position along the Cartesian axis and three of rotation around these axis) around 100 times per second by radio-frequencies in float number format (see Polhemus documentation in appendix **??** for more details). Through the Polhemus software, then with a Windows software developed by the CIANT, these data are calibrated and mapped (following a body scheme) then sent in raw-float format on the IP-Network (broadcasted).
They used in the CIANT 11 sensors surrounded in a two-meters-sided cube. Sensors were set as followed : two sensors per arm, two sensors per leg, two on the body : down back and between shoulder blades and one on the head.



6.1.2 Max/MSP

To program and produce sound and music, I am used since my first practice (summer 2006) to the software Max/MSP (**Cycling'74**, **IRCAM**). Max/MSP is a graphical programing language which principle is to connect elementary boxes (embeding elementary function) to create an algorithmic sequence. The core of the program and the elementary functions are basically programmed in C language. A Software Development Kit is available to program your own boxes in C using an application programming interface (API). It is possible also to program boxes in Java or JavaScript.

6.1.3 Interfacing

After looking how to acquire the data on the network and parse it with the already existing boxes and tools, I arrived to the conclusion that the only way to get these data fully usable in Max/MSP was to program my own box. As I needed to control the UDP port and parse the raw float data following the scheme given by the CIANT (given in C++ language) I chose the language C which I already knew. While I was used to graphically programmed in Max/MSP, I had never programmed my own box using the Max/MSP API. It took me a whole week to build a fully functional box.

I worked in three steps on this development. The first step was to understand and learn how to use the Max/MSP API to program a box. To help me, I had only the Max/MSP Software Development Kit documentation and example and of course Internet to search some additional example or explanations. The API is quite complex and rather object-oriented-programing like. You have to write creation and deleting method for your object using a given syntax and define some classes corresponding to the object you need to program. For example, this is the code of an empty object which does nothing at all, has no inputs, no outputs :

```
#include "ext.h"

typedef struct _empty
{
        struct object m_ob;
} b_empty;

void *empty_class;

void *empty_new(void);

int main(void)
{
        setup((t_messlist **)&empty_class, (method)empty_new,0L,
                                (short)sizeof(b_empty), 0L, 0);
        return 0;
}

void *empty_new(void)
{
        b_empty *x;
        x = (b_empty *)newobject(empty_class);
        return x;
}
```

The second step was to learn how to program the controlling of an UDP port and to adapt it for the Max/MSP API. Once again, I mostly looked on Internet and found some documents, tutorials or help. Here is the basic code to open a UDP port :

```
void socket_bind(b_udp_parser *d, long port)
{
        d->sock = socket(AF_INET, SOCK_DGRAM, 0);
        if (d->sock<0)
                error("in opening socket");
        //else
                //post("socket : ok");

        d->source.sin_family = AF_INET;
        d->source.sin_port = htons(port);
        d->source.sin_addr.s_addr = INADDR_ANY;
        fcntl(d->sock,F_SETFL,O_NONBLOCK);
        if (bind(d->sock,(struct sockaddr*)&d->source,sizeof(d->source))!=0)
                error("in binding socket");
        //else
                //post("bind : ok");
}
```

and the function to receive the data every microsecond on the port :

```
void recep(b_udp_parser *d)
{
        socklen_t size=sizeof(struct sockaddr_in);
        int nbcar=recvfrom(d->sock,d->buf, sizeof(d->buf),0,
                                (struct sockaddr*)&d->source,&size);
        if(nbcar<0)
        {
                d->try++;
                if(d->try==100)
                {
                        error("no incoming message, please retry");
                        stop(d);
                }
                return;
        }
        parse(d,(char*)d->buf,nbcar,d->sensors);
        d->try=0;
}
```

Finally I had to handle the different process in order not to freeze Max/MSP waiting for the data. There, I used the possibility of the Max scheduler to get separate internal thread and clock to get a waiting state without influencing on the other process. This behave somehow like a multi-thread processor.
To test my boxes I used the *MoCap Spoofer*, a software programed by one of the engineers of the CIANT to simulate the sending of Motion Capture data on the network.

N.B. : I worked on my Macintosh laptop and I had to port every object I programed on Windows during the last week of my internship. This is important because the managing of the socket (UDP Port) is quite different between Mac OS X operating system (which is an UNIX system) and Windows XP (see appendix **??** for both of the complete code sources).

6.2 Motion Capture : Artistical point of view

6.2.1 Music

Thanks to Max/MSP, it is really easy to program almost everything in the music field and quite quick. Thus I had to choose what would be pertinent for this project with a dancer.

The first direction I chose was to use the less already-recorded material as possible and give greater place to synthesized sound. This choice was justified by the project itself. It would have been much less interactive and less interesting – on a musical point of view – to use some pre-recorded sound or already built loops etc. The part of my musical creating and interacting possibilities would also have been reduced with this kind of musical material.

After these thoughts and before talking or trying with the dancer I programed some little simple modules performing one type of sound synthesis : an additive synthesis module, a frequency modulation module, an amplitude modulation module, some effects modules (Low Frequency Oscillator, echo, filters...). My idea was to choose some parameters in each module that the dancer could control and build up with this a really new instrument based on the body and the wishes of the dancer. Here is an example (fig **??**) of a module creating interference between two oscillators. The frequency of the two oscillators is, at the beginning, the same and very low (50Hz). Then the distance between the two arms is linked to the two frequencies, increasing one, decreasing the other one to create sub-audio (but audible) amplitude interference between the sound of the two oscillators.

I used the *MoCap Spoofer* again to test the modules I programed without any dancer and even without the functioning of the motion capture system. It was really hard to get a good idea of what will render these modules in real-time because the *MoCap Spoofer* is functioning with a local database that sends real movement data previously recorded but twice slower as the real movement.

6.2.2 Dance

From the end of the second week, I met regularly the dancer who would be equiped with the Motion Capture system during the Session II of TransISTor. Fortunately she was really interested in this kind of interaction I was programing. So it has been possible to think and decide together what would be interesting to control with her movements.

In this experimentation the main problem was to make understandable for her, who is not musician, what her movements are controlling so that she would be able to play with it and create a real exchange between movement and sound. On one hand I had to make simpler my audio modules and clearer the link between the sensor and the sound. But on the other hand, to keep this performance interesting on an musical point of view I had also to keep a sufficient complexity of music construction. I discovered that this equilibrium is very hard to find but really interesting to search.

Unfortunately, we had only one opportunity to test the whole real-time installation before leaving Prague for Karlovy Vary. We had then three days of working during the first days of Session II to continue testing, improving and to make the *instrument* presentable. Three days are very short to succeed when people are working their whole life trying to create this kind of installation (for example in **IRCAM**, Paris) ! But we performed a little improvisation (around two minutes) with this *instrument* which has been appreciated and filmed (see the CD attached appendix **??**).

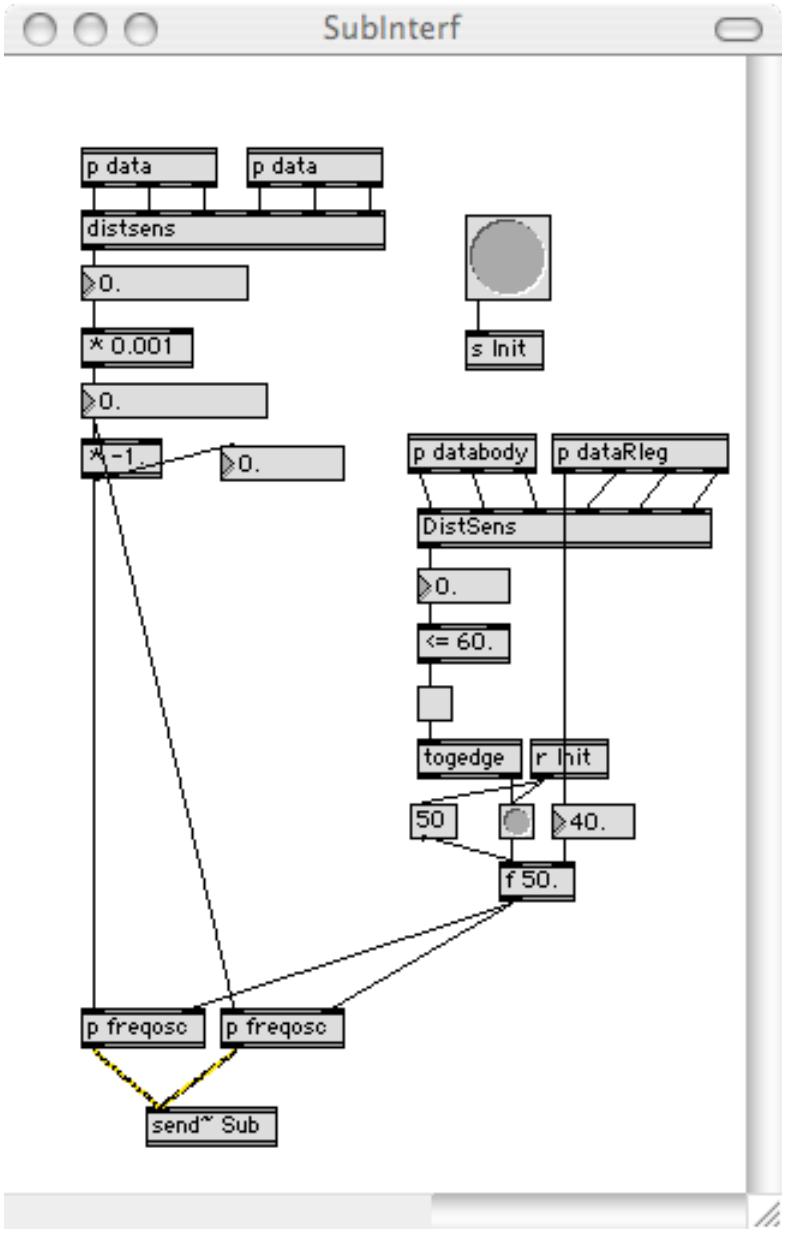


Figure 6.1: Amplitude interference oscillators

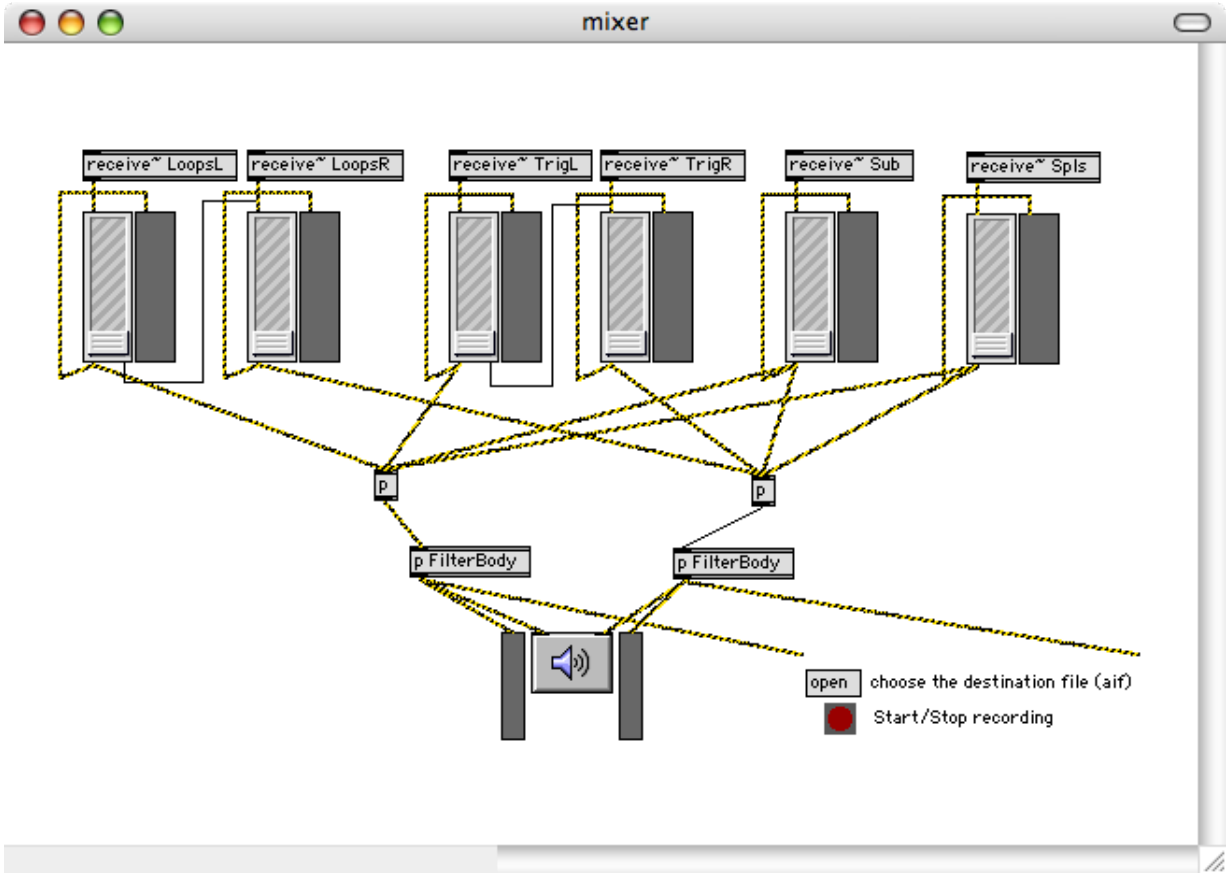


Figure 6.2: Mixing and recording *patch*

6.2.3 Interactivity

The final performance was made with five sound modules linked to each members and the dancer's body.

Her right arm was linked to a loop I synthesized. She controlled the position (in time) of different element of the loop. The reading speed of the loop was controlled by the horizontal position of her body in the dance area. And her right leg and left arm had also an influence over the timbre of the loop.

Her right and left arms were triggering some randomly-additive-synthesized sounds.

The distance between her two arms was linked – as said before – to the interference between two very low frequency oscillators. Her right leg had also the possibility to change the base frequency of the two oscillators.

The rotation of her head around the vertical (turning right and left) and the horizontal (noding) axis were controlling the reading of two pre-recorded samples.

Finaly, the altitude of her whole body was changing the cut-frequency of a low-pass filter.

All these sounds were sent to a mixing (and recording) *patch* (fig **??**) that I controlled in real-time. And I could also have an influence on all the modules she was playing with.

With this *instrument* we also recorded the soundtracks of two choreography included in two of the four short movies produced by the participants of TransISTor.

Chapter 7

Arduino hardware with Max/MSP

The only real workshop that took place during the Session III of TransISTor was about Arduino hardware interfaced with a computer. Two German students presented this kind of hardware and one of their project using it during one morning. The two following afternoon were dedicaced to workshops on these hardware interfaces to teach the basics of this to the participant.
I was asked by my supervisor to use also one of these development card and try to create some funny interactions with sound. Unfortunately the Arduino cards arrived to the CIANT two days only before the day of the presentation. Once again I had a very short time to prepare something.

7.1 Arduino hardware

Arduino is an open-source physical computing platform based on a simple i/o board and a development environment that implements the Processing language. Arduino can be used to develop stand-alone interactive objects or can be connected to software on your computer (e.g. Flash, Processing, Max/MSP). The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free.
Arduino is different from other platforms that can be found on the market because of these features :

- *The Arduino Project was developed out of an educational environment and is therefore great for newcomers to get things working quickly.*
- *It is a Multi Platform environment; it can run on Windows, Macintosh and Linux.*
- *It is Based on the Processing programming Integrated Development Environment.*
- *It is programmed via a USB cable not a serial port. This is useful because many modern computers don't have serial ports anymore.*
- *It is Open Source hardware and software - If you wish you can download the circuit diagram, buy all the components, and make your own, without paying anything to the makers of Arduino.*
- *The hardware is cheap. The USB board cost about EUR 20 and replacing a burnt out chip on the board is easy and costs no more than EUR 5. So you can afford to make mistakes.*
- *There is an active community of users so there is plenty of people who can help you.*

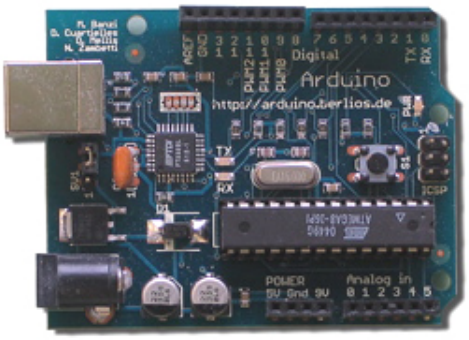


Figure 7.1: The Arduino board

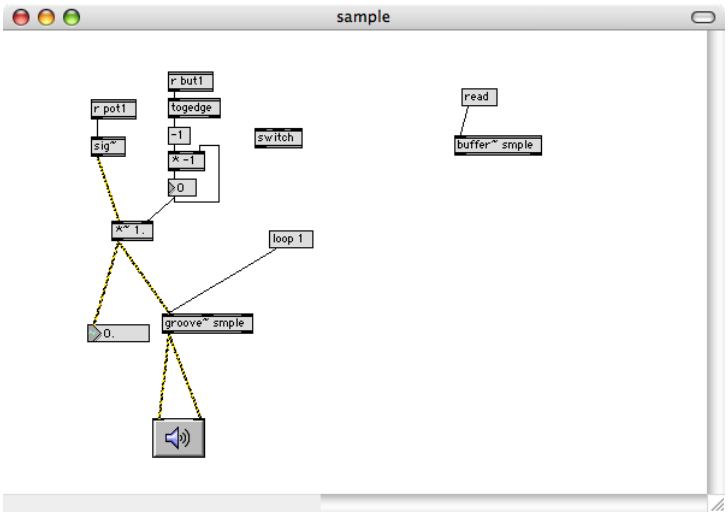


Figure 7.2: Potentiometer-controlled loop player

Technically, the Arduino board (fig **??**) is based on an *Atmega 168* processor and a *FTDI232R* USB interface. It has 13 digital in/outputs and 6 analog inputs (from 0 to 5V). It is USB or externally powered. An Integrated Development Environment is downloadable on the Internet. The language used is based on Processing script which is very close from a simplified C.
It is really very easily usable even for someone who had never worked on an electronic development card.

7.2 Realizations

As I had to use this board connected with Max/MSP, I had first to implement the firmware for Max/MSP on the microprocessor . This firmware is available on the website of Arduino and easy to use. It just works ! After this step I could receive and send data " direclty " from Max/MSP. My first programing was the control of the main audio volume of Max/MSP with a potentiometer.
Then in the time I had, I developed three little installations.
The control of a loop-player with a potentiometer (fig **??**), The the frequency control of an oscillator with an luminosity sensor (fig **??**), and a visualization of the sound output level of Max/MSP on 6 LEDs (fig **??**).

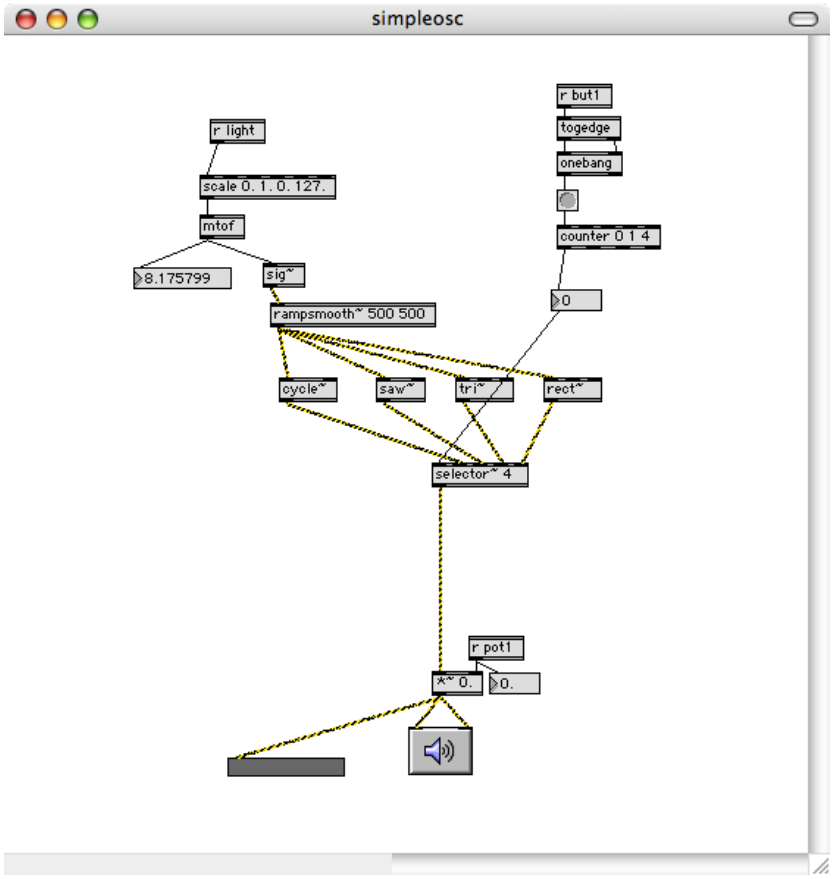


Figure 7.3: Light-controlled oscillator

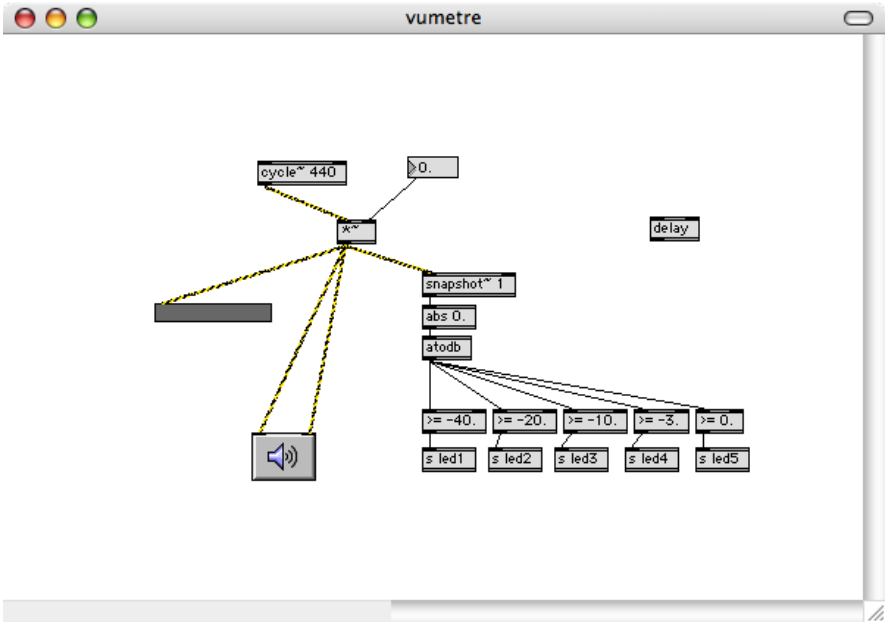


Figure 7.4: LED level visualizer

Conclusion

After my first internship (summer 2006) where I discovered the world of computer-assisted music – tools, softwares, problematics, instruments etc – I was hoping to find a new practice in this field in another place with other projects and orientations. I had the chance, thanks to Mr Reynal to find the CIANT in the beautiful city of Prague.
Then I arrived in Prague. Discovering the city as well as the team of CIANT and its functioning, I was given very soon the responsibility on a whole part of a very exciting project : TransISTor training sessions. Suddenly thrown in an environment where the border between the artist and the engineer no longer exists, my work promised to be very interesting.

During these five weeks working for the CIANT, considered from the beginning not only as an engineer but moreover as a musician, I really enjoyed building up a project which combined a technical part and an artistic part. I had to manage and finalize a whole project in a very short time, with the given constraints but also a lot of means and possibilities. The role of choices was very important and the achieving of something presentable too.
During these weeks I also followed the activities of the CIANT, attended conferences, openings, meetings in this field of arts an new technologies where I learned a lot and met some very interesting people from all over Europe. That will give me a lot of opportunities to travel and work outside France which I consider as very important in this field growing and evolving very fast all over the world.

At the end of my internship, certainly satisfied of my work, Pavel Smetana has offered me to come back for my six-month final internship next year and even to work for three years for the CIANT, mostly on an other project called **CO-ME-DI-A** (COoperation and MEdiation in DIgital Arts) dealing with the development of a network art platform in collaboration with (among others) : IRCAM (Paris), Casa Paganini (Genova), SARC (Belfast), KUG (Graz).
I am very proud of this offer and I consider it as the best reward of my work I could get.

Appendices

Appendix A

Polhemus brochure

LIBERTY LATUS

Large Area Tracking Untethered System

TOTALLY WIRELESS TRACKING

The LIBERTY™ LATUS™ (Large Area Tracking Untethered System) represents a whole new dimension in tracking technology, one that offers a totally wireless, full 6 Degree-Of-Freedom solution. The system has speed, ease-of-use via an intuitive Graphical User Interface (GUI) and is capable of tracking up to 12 independent markers over large areas. Because of the improved signal-to-noise ratios, LIBERTY LATUS offers increased stability while providing consistent high quality data, all while being completely untethered.

FEATURES

Wireless

Totally wireless markers are completely self-contained, each housing a lithium ion battery assembly that provides up to 2.5 hours of power. Each system may track up to 12 markers independently.

Reduced Distortion

The system is capable of reducing any distortion effects normally seen with long range electromagnetic systems because of its short range distributed receptor architecture, and enhanced signal-to-noise ratio.

Scalable

Four receptor channels are available on the base product; the system is upgradeable to 8, 12, or 16 receptor channels within the same chassis by having additional circuit boards installed. Each receptor can cover up to 50 sq. ft. (4.7 sq m).

Communications Interface

LIBERTY LATUS communicates via RS-232 serial or USB interface. Both are included in the base unit.

Multiple User Definable Profiles

The GUI allows for three independent user-definable profiles for setting system parameters such as filtering, output formats, coordinate rotations and much more.

Multiple Output Formats

Users may select position in Cartesian coordinates (English or metric); orientation in direction cosines, Euler angles or quaternions.

THE ONLY WIRELESS CHOICE

Unique in Wireless Tracking Technology

LIBERTY LATUS provides truly wireless tracking. There are absolutely no wires - each marker is self-contained. The system is capable of tracking up to 12 markers for full 6 Degree-Of-Freedom solutions over large areas. Each marker is tracked in space by a receptor that covers up to an 8 foot (2.44m) diameter. Each system is capable of connecting up to 16 receptors for total coverage of hundreds of square feet. Systems may also be concatenated for even larger area coverage. All wireless communication is via a proprietary magnetic data link.

Easy, Intuitive User Interface

LIBERTY LATUS comes standard with Windows® 2000/XP GUI and a comprehensive, easy-to-use Software Developers Kit (SDK). The GUI allows three independent user-definable profiles for setting system parameters such as filtering, output formats, coordinate rotations and much more. This is a valuable feature for multiple applications or users. For visualization, an integrated motion box provides navigable points of view and can include text data. Additional features include a data record/playback component, plus the ability to quickly export data via Microsoft® "Named Pipe".

AC Magnetics: increased Stability, Resolution, Speed and Range

Incorporating state of the art Digital Signal Processor (DSP) electronics in concert with AC magnetics provides the user with improved signal-to-noise ratios which increase range, stability, resolution and speed. The system is essentially unaffected by facility power grids or electric power motors, and provides update rates of 94 or 188 Hz measurements per second maintained for all markers, allowing for consistent, high quality data.

APPLICATIONS

Military Operations in Urban Terrain (MOUT)

Having the ability to track 12 markers over a large area and not having to be concerned about line-of-sight obstacles, LIBERTY LATUS makes an ideal system for MOUT applications. Tracking over an entire house or scene including stairways is easily achieved by appropriate receptor location. Placing a marker on the weapon and the head allows the instructor to track location (X,Y,Z) and direction (Az., El., Rl.) of both the weapon and the head for after-action review.

Biomechanical and Sports Analysis

With an update rate of 188 Hz per marker and no wires to encumber movement, LIBERTY LATUS can collect data from the swing of a baseball bat, an athlete's fast-paced movements, or gait movement and limb rotation for real-time analysis of both children and adults.

Virtual or Augmented Reality

From the beginning, Polhemus systems have been the selected choice for Virtual or Augmented Reality head and body tracking. A totally wireless system, LIBERTY LATUS is the only logical choice for CAVE, Power Wall, VR and AR applications.

LATUS

TECHNICAL SUMMARY

COMPONENTS

System Electronics Unit (SEU)
The SEU contains the hardware and software necessary to sense the magnetic fields generated by the markers, compute position and orientation, and interface with the host computer via RS-232 or USB.

Marker
Markers contain the electromagnetic source, control electronics and a rechargeable lithium ion battery assembly. The system reports the position and orientation of each marker that is within range of at least one receptor. The system is capable of accommodating up to 12 markers. The battery assembly provides power for approximately 2.5 hours and is easily removed for recharging. Markers weigh 2 ounces and are easily mounted on the body with our optional body mount kit.

Receptor
Receptors contain electromagnetic receiving elements cast into a solid assembly that detects the magnetic signals emitted by the marker(s) for up to 8 feet (2.44m) in diameter. Cable length is 60 feet (18.3m). This lightweight, small cube can be easily mounted to almost any surface. The system is capable of accommodating up to 16 receptors.

Battery Charger
The QUAD Charger is capable of charging four battery assemblies simultaneously. Charge time is approximately 2 hours.

POLHEMUS
First in the third dimension®

The systems are not certified for medical or bio-medical use. Any reference to medical or bio-medical use are examples of what medical companies have done with the systems after obtaining all necessary or appropriate medical certifications. The end user/OEM must comply with all pertinent FDS/CE and all other regulatory requirements.

SPECIFICATIONS

Update Rate
188 Hz for 1 to 8 markers
94 Hz for 9 to 12 markers

Latency
Approximately 5 milliseconds

Number of Wireless Markers
1 - 12

Number of Receptors
1 - 16

Static Accuracy
0.5 degree and 0.1 inch (0.254cm) using 1 marker and 1 receptor at 30 inches (76.2cm). Accuracy is installation dependent, typical accuracy may normally result in 1 to 3 degrees and 1 to 3 inches (2.54cm to 7.62cm).

Resolution
0.00015 inches (0.038mm) at 12 inches (30cm) range; 0.0012 degrees orientation

I/O Ports
USB; RS-232 to 115,200 Baud rate; both are standard

Range
Each receptor may report position and orientation of a marker within an 8 foot (2.4m) diameter

Multiple Systems
Multiple systems may be concatenated to extend range

Angular Coverage
All-attitude

Data Format
Operator selectable ASCII or IEEE 754 binary; English/Metric units

External Event Hardware Input
Reportable in output data stream

Output Sync Pulse
TTL frame sync output

Physical Characteristics
SEU
12.2 inches (31cm) L x 7 inches (17.8cm) W x 11 inches (27.94cm) H
10.5 pounds (4.8kg)

Wireless Marker
2.92 inches (7.4cm) L x 1.56 inches (3.96cm) W x 0.85 inches (2.16cm) H
2 ounces (56.7gm)

Receptor
2.5 inches (6.35cm) L x 1.4 inches (3.56cm) W x 1.4 inches (3.56cm) H 3.2 ounces (90.7gm)
Cable length: 60 feet (18.3m)

Power Requirements
85-264 VAC, 47-440 Hz, single phase, 50 W

Regulatory
FCC part 15, class A
CE: En50081-1, class A, emissions
EN50082-1, class 2, immunity
EN61010, safety

40 Hercules Drive • PO Box 560 • Colchester, Vermont 05446-0560
US and Canada 800.357.4777 • 802.655.3159 • fax 802.655.1439 • www.polhemus.com

ISO9001

Appendix B

Code

B.1 UNIX Code for the Max/MSP object bc.udp.parse

```
#include "ext.h"
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <MaxAPI/MaxAPI.h>

#pragma pack(1)
struct TRawSensor
{
        char leader[5];
        float pos[3];
        float rot[3];
        bool valid;
};
#pragma pack()
typedef struct TRawSensor TSensorMatrix[2][16];

typedef struct _udp_parser
{
        struct object m_ob;
        void* l_out;
        long tempo;
        void* max_schedul;
        void* s_schedul;
        void* c_clock;
        long port;
        Atom list_sens[7];
        int i_status;
        struct sockaddr_in source;
        int sock;
        int try;
        char* buf[500];
        TSensorMatrix* sensors;
} b_udp_parser;

void *udp_parser_class;

void *udp_parser_new(long tempo, long port);
void start(b_udp_parser *d);
void stop(b_udp_parser *d);
```

```
void socket_bind(b_udp_parser *d, long port);
void recep(b_udp_parser *d);
void close_socket(b_udp_parser *d);
void parse(b_udp_parser *d,char* mess,int size,TSensorMatrix* sensorMatrix);
void output(b_udp_parser * d, struct TRawSensor* sensdata,int numcapt);
void clock_function(b_udp_parser* d);
void udp_parser_free(b_udp_parser* d);

int main(void)
{
        setup((t_messlist **)&udp_parser_class, (method)udp_parser_new,(method)udp_parser_free,
        addmess((method)start,"start",0); //start message handeling
        addmess((method)stop,"stop",0); //stop message handeling
        return 0;
}

void *udp_parser_new(long tempo,long port)
{
        b_udp_parser *udpars;
        udpars = (b_udp_parser *)newobject(udp_parser_class);
        udpars->l_out = listout(udpars);
        if((*udpars->buf=malloc(sizeof(char)*500))==NULL)
        {
                error("memory allocation");
                exit(EXIT_FAILURE);
        }
        if((udpars->sensors=malloc(sizeof(TSensorMatrix[2][16])))==NULL)
        {
                error("memory allocation");
                exit(EXIT_FAILURE);
        }
        udpars->s_schedul=scheduler_new();
        udpars->tempo=tempo;
        udpars->port=port;
        udpars->c_clock=clock_new(udpars,(method)clock_function);
        return udpars;
}

void start(b_udp_parser * d)
{
        if(d->i_status==1)
        {
                post("MoCap already parsing on port %d",d->port);
                return;
        }
        post("MoCap parsing on port %d",d->port);
        d->i_status=1;
        d->try=0;
        socket_bind(d,d->port);
        d->max_schedul=scheduler_set(d->s_schedul);
        clock_delay(d->c_clock,0L);
}

void stop(b_udp_parser * d)
{
        if(d->i_status==0)
        {
                post("MoCap already stoped on port %d",d->port);
                return;
        }
```

```
        clock_unset(d->c_clock);
        post("MoCap stoped on port %d",d->port);
        close_socket(d);
        d->i_status=0;
}

void socket_bind(b_udp_parser *d, long port)
{
        d->sock = socket(AF_INET, SOCK_DGRAM, 0);
        if (d->sock<0)
                error("in opening socket");
        //else
                //post("socket : ok");

        d->source.sin_family = AF_INET;
        d->source.sin_port = htons(port);
        d->source.sin_addr.s_addr = INADDR_ANY;
        fcntl(d->sock,F_SETFL,O_NONBLOCK);
        if (bind(d->sock,(struct sockaddr*)&d->source,sizeof(d->source))!=0)
                error("in binding socket");
        //else
                //post("bind : ok");
}

void recep(b_udp_parser *d)
{
        socklen_t size=sizeof(struct sockaddr_in);
        int nbcar=recvfrom(d->sock,d->buf, sizeof(d->buf),0,(struct sockaddr*)&d->source,&size)
        if(nbcar<0)
        {
                d->try++;
                if(d->try==100)
                {
                        error("no incoming message, please retry");
                        stop(d);
                }
                return;
        }
        parse(d,(char*)d->buf,nbcar,d->sensors);
        d->try=0;
}

void close_socket(b_udp_parser *d)
{
        if(close(d->sock)!=0)
           error("in closing socket");
}


void parse(b_udp_parser *d,char* mess,int size,TSensorMatrix* sensorMatrix)
{
        char* eol = (char*)memchr(mess,10, size);
        if (eol == NULL)
        {
                //error("incorrect incoming message");
                return;
        }
        //*eol = '\0';
        int frame, frameSize, headerSize,recordSize;
        sscanf(mess, "Frame %d FrameSize %d HeaderSize %d RecordSize %d",&frame, &frameSize, &h
```

```
        int records = (frameSize - headerSize) / recordSize;
        char leader[6];
        recordSize = sizeof(float)*6+5;
        leader[5]='\0';
        int i=0;
        for (i=0; i<records; i++)
        {
                char *record = (char*)(mess + i*recordSize + headerSize);
                memcpy(leader, record,5);
                int board = leader[1] - '0';
                int station = leader[3] - '0' - 1 ;
                memcpy(&(sensorMatrix[station][board]),record, recordSize);
                sensorMatrix[station][board]->valid = true;
                int numcapt = (station)*8+board;
                output(d, sensorMatrix[station][board],numcapt);
        }
}

void output(b_udp_parser * d,struct TRawSensor* sensdata, int numcapt)
{
        SETLONG(d->list_sens,numcapt);
        int j;
        for (j=0;j<3;j++)
        {
                SETFLOAT(d->list_sens+j+1,sensdata->pos[j]);
                SETFLOAT(d->list_sens+j+4,sensdata->rot[j]);
        }
        outlet_list(d->l_out,0L,7,d->list_sens);
}

void clock_function(b_udp_parser* d)
{
        scheduler_set(d->s_schedul);
        clock_delay(d->c_clock,d->tempo);
        recep(d);
}

void udp_parser_free(b_udp_parser* d)
{
        stop(d);
        clock_unset(d->c_clock);
        clock_free(d->c_clock);
        scheduler_set(d->max_schedul);
}
```

B.2 Windows Code for the Max/MSP object bc.udp.parse

```
#include "ext.h"
#include <windows.h>

#define TRUE 1
#define FALSE 0
typedef int bool;

#pragma pack(1)
struct TRawSensor
{
        char leader[5];
        float pos[3];
        float rot[3];
    bool valid;
};
#pragma pack()
typedef struct TRawSensor TSensorMatrix[2][16];

typedef struct _udp_parser
{
        struct object m_ob;
        void* l_out;
        long tempo;
        void* max_schedul;
        void* s_schedul;
        void* c_clock;
        long port;
        Atom list_sens[7];
        int i_status;
        struct sockaddr_in source;
        int sock;
        int try;
        char* buf[500];
        TSensorMatrix* sensors;
} b_udp_parser;

void *udp_parser_class;

void *udp_parser_new(long tempo, long port);
void start(b_udp_parser *d);
void stop(b_udp_parser *d);
void socket_bind(b_udp_parser *d, long port);
void recep(b_udp_parser *d);
void close_socket(b_udp_parser *d);
void parse(b_udp_parser *d,char* mess,int size,TSensorMatrix* sensorMatrix);
void output(b_udp_parser * d, struct TRawSensor* sensdata,int numcapt);
void clock_function(b_udp_parser* d);
void udp_parser_free(b_udp_parser* d);

int main(void)
{
        setup((t_messlist **)&udp_parser_class, (method)udp_parser_new,(method)udp_parser_free,
        addmess((method)start,"start",0); //start message handeling
        addmess((method)stop,"stop",0); //stop message handeling
        return 0;
}

void *udp_parser_new(long tempo,long port)
```

```
{
        b_udp_parser *udpars;
        udpars = (b_udp_parser *)newobject(udp_parser_class);
        udpars->l_out = listout(udpars);
        if((*udpars->buf=malloc(sizeof(char)*500))==NULL)
        {
                error("memory allocation");
                exit(EXIT_FAILURE);
        }
        if((udpars->sensors=malloc(sizeof(TSensorMatrix[2][16])))==NULL)
        {
                error("memory allocation");
                exit(EXIT_FAILURE);
        }
        udpars->s_schedul=scheduler_new();
        udpars->tempo=tempo;
        udpars->port=port;
        udpars->c_clock=clock_new(udpars,(method)clock_function);
        return udpars;
}

void start(b_udp_parser * d)
{
        WSADATA WSAData; //for windows only
        if(d->i_status==1)
        {
                post("MoCap already parsing on port %d",d->port);
                return;
        }
        post("MoCap parsing on port %d",d->port);
        d->i_status=1;
        d->try=0;
    WSAStartup( MAKEWORD( 2, 2 ), &WSAData ); //for windows only
        socket_bind(d,d->port);
        d->max_schedul=scheduler_set(d->s_schedul);
        clock_delay(d->c_clock,0L);
}

void stop(b_udp_parser * d)
{
        if(d->i_status==0)
        {
                post("MoCap already stoped on port %d",d->port);
                return;
        }
        clock_unset(d->c_clock);
        post("MoCap stoped on port %d",d->port);
        close_socket(d);
        d->i_status=0;
}

void socket_bind(b_udp_parser *d, long port)
{
        int iMode;
        d->sock = socket(AF_INET, SOCK_DGRAM, 0);
        if (d->sock<0)
                error("in opening socket");
        //else
                //post("socket : ok");
```

```
        d->source.sin_family = AF_INET;
        d->source.sin_port = htons((short)port);
        d->source.sin_addr.s_addr = INADDR_ANY;
        iMode = 1;
        ioctlsocket(d->sock, FIONBIO, &iMode); //non blocking mode for windows
        //fcntl(d->sock,F_SETFL,O_NONBLOCK);     //non blocking mode for unix
        if (bind(d->sock,(struct sockaddr*)&d->source,sizeof(d->source))!=0)
                error("in binding socket");
        //else
                //post("bind : ok");
}

void recep(b_udp_parser *d)
{
        int size=sizeof(struct sockaddr_in);
        int nbcar=recvfrom(d->sock,(char*)d->buf, sizeof(d->buf),0,(struct sockaddr*)&d->source
        if(nbcar<0)
        {
                d->try++;
                if(d->try==100)
                {
                        error("no incoming message, please retry");
                        stop(d);
                }
                return;
        }
        parse(d,(char*)d->buf,nbcar,d->sensors);
        d->try=0;
}

void close_socket(b_udp_parser *d)
{
        if(closesocket(d->sock)!=0)
           error("in closing socket");
           WSACleanup(); //for windows only
}


void parse(b_udp_parser *d,char* mess,int size,TSensorMatrix* sensorMatrix)
{
        int frame;
        int frameSize;
        int headerSize;
        int recordSize;
        int records;
        char leader[6];
        int i;
        int board;
        int station;
        int numcapt;
        char* eol = (char*)memchr(mess,10, size);
        if (eol == NULL)
        {
                //error("incorrect incoming message");
                return;
        }
        //*eol = '\0';
        sscanf(mess, "Frame %d FrameSize %d HeaderSize %d RecordSize %d",&frame, &frameSize, &h
        records = (frameSize - headerSize) / recordSize;
        recordSize = sizeof(float)*6+5;
```

```
        leader[5]='\0';
        i=0;
        for (i=0; i<records; i++)
        {
                char *record = (char*)(mess + i*recordSize + headerSize);
                memcpy(leader, record,5);
                board = leader[1] - '0';
                station = leader[3] - '0' - 1 ;
                memcpy(&(sensorMatrix[station][board]),record, recordSize);
                sensorMatrix[station][board]->valid = true;
                numcapt = (station)*8+board;
                output(d, sensorMatrix[station][board],numcapt);
        }
}

void output(b_udp_parser * d,struct TRawSensor* sensdata, int numcapt)
{
        int j;
        SETLONG(d->list_sens,numcapt);
        for (j=0;j<3;j++)
        {
                SETFLOAT(d->list_sens+j+1,sensdata->pos[j]);
                SETFLOAT(d->list_sens+j+4,sensdata->rot[j]);
        }
        outlet_list(d->l_out,0L,7,d->list_sens);
}

void clock_function(b_udp_parser* d)
{
        scheduler_set(d->s_schedul);
        clock_delay(d->c_clock,d->tempo);
        recep(d);
}

void udp_parser_free(b_udp_parser* d)
{
        stop(d);
        clock_unset(d->c_clock);
        clock_free(d->c_clock);
        scheduler_set(d->max_schedul);
}
```

Appendix C

Video