BEATLED - THE SOCIAL GAMING PARTYSHIRT

Tom De Nies
Ghent University - IBBT
tom.denies@ugent.be

Thomas Vervust
Ghent University - CMST
thomas.vervust@ugent.be

Michiel Demey
Ghent University - IPEM
michiel.demey@ugent.be

Rik Van de Walle
Ghent University - IBBT
rik.vandewalle@ugent.be

Jan Vanfleteren
IMEC/Ghent University - CMST
jan.vanfleteren@ugent.be

Marc Leman
Ghent University - IPEM
marc.leman@ugent.be

ABSTRACT

This paper describes the development of a social game, BeatLED, using music, movement and luminescent textile. The game is based on a tool used in research on synchronization of movement and music, and social entrainment at the Institute of Psychoacoustics and Electronic Music (IPEM) at Ghent University. Players, divided into several teams, synchronize to music and receive a score in real-time, depending on how well they synchronize with the music and each other.

While this paper concentrates on the game design and dynamics, an appropriate and original means of providing output to the end users was needed. To accommodate this output, a flexible, stretchable LED-display was developed at CMST (Ghent University), and embedded into textile.

In this paper we analyze the characteristics a musical social game should have, as well as the overall merit of such a game. We discuss the various technologies involved, the game design and dynamics, a proof-of-concept implementation and the most prominent test results.

We conclude that a real-world implementation of this game not only is feasible, but would also have several applications in multiple sectors, such as musicology research, team-building and health care.

1. INTRODUCTION

In today's games, the social aspect has become more than just an extra feature. Game developers are incorporating social interaction as a key feature into their games, and are researching alternative ways for users to interface with the gaming platforms. BeatLED is a so called "social" game, and this paper will describe it's key features, internal workings and applicability.

First we narrow down the general concept of a "social game", and how this concept relates to earlier work. We also motivate why the game was developed, and how it can be applied in different fields. Next, we present the various technologies and algorithms used to accommodate the game. The game dynamics are described, followed by an insight into the proof-of-concept implementation. We also discuss the set-up and results of the various tests that were performed. These tests include algorithm performance tests and general test sessions with actual end users. Finally, the future plans and possibilities for the project are suggested and a conclusion is made.

2. SOCIAL GAME

2.1 What/Why?

Interaction - on-line or otherwise - has become a must for all recent games. Studies show that a large part of the latest console gaming generation considers the social aspect as the most important motivation to play [1]. Typical examples of *social games* are those developed for the Nintendo Wii consoles. When we examine these games, we are able identify the following features:

Multiplayer Naturally, in order to be called "social", a game should facilitate more than one player. Ideally, easy, even dynamic expansion of the group should be possible. In the best case, one can choose the number of players arbitrarily.

Interactivity In order to be challenging and fun, the game should include a tight action-perception coupling, both between the players and the game, as among the different players themselves. A slow responding game, where users need to wait for feedback based on their actions, is not an option.

Intuitivity The game itself, its rules and its controls, should all be intuitive and easy to comprehend. During the gameplay, players should not be focusing on how to control the game, or try to comprehend its rules. Instead they should only be focusing on the general idea of the game, and on the other players. Keeping the controls natural not only allows the players to be fully absorbed by the game, but also makes it easy to introduce new players to the game, hence contributing to the social factor.

Motivation (to play in group) The game should not only encourage participation, it should also easily attract more players. For example, certain game modes could only be made available when a predefined number of players is reached. To make the game inherently more challenging with more players would be even better.

Social bonding The game should incorporate a certain *social bonding* factor. This means playing the game should be beneficial to the relationships between the players. This factor is also present in many of the team-building games, often seen in larger businesses.

During development of this game, it is important to keep these features in mind. They are also recognized in many console games, especially those intended for Nintendo Wii or Microsoft Kinect. We will attempt to incorporate all of these features in our social game.

The game we are building is centered around a widespread social activity, namely dancing to music. Dancing is by nature an intuitive and social activity, it allows you to interact with other people without conversation, and the larger the group, the more fun is guaranteed. All five of the above factors can already be discerned in that activity. Therefore, building a social game using music and movement is a logical step.

We will build a game that allows players to dance to music, and to receive a certain score, depending on how well they synchronize with the music, and with each other.

2.2 Similar Games & Applications

To describe all games based on music, dance and/or movement is beyond the scope of this text. We will highlight the most relevant and well-known examples.

2.2.1 Similar Games

Dance Dance Revolution DDR, by Konami, is probably the best-known dance game. Players receive instructions on a screen in the form of a sequence of arrows, each arrow corresponding with a specific movement. When they make the correct movement at the correct time, they receive a positive score. Their movements are recorded using a proprietary dance mat, as shown in figure 1. The number of players is limited, with a maximum of 2 or 4 players. Players are also very limited in movement, partly because they are obliged to follow the directions of the game and partly because of the limited space they have to dance.

Just Dance The spatial limitations of DDR are partially solved by Just Dance for Nintendo Wii. Here, players interact only with the controller, but again, they must watch the screen and follow the specific instructions of the game, as shown in figure 2.

Dance games for Kinect Another approach to dance games is presented by the Kinect, a camera-based game system recently developed by Microsoft. These games are very similar in functionality to the previously discussed games. Players have to follow the instructions provided on the screen, only now these instructions represent more complex movements. Using its depth-sensitive camera, the Kinect is able to track detailed movements of the players and match them to the instructions. However, this also implies that players need to remain inside the field of view of the camera.

Figure 1. DDR dance mat: players have to match their movements to arrows shown by the game.

Figure 2. Just Dance 2 for Nintendo Wii screen interface. The hand that holds the wii remote is highlighted.

The major difference between all currently available games and ours, is that our game takes into account the *mutual synchronization* between the movements of players. It also imposes *no limitations in space*.

2.2.2 Exergames

The games described above are so called *exergames*, since many people use them as a motivating means of physical exercise. DDR was even deployed in some schools, as part of the exercise program [2]. This shows that a social game such as the one discussed in this paper can also be applied as a means of exercise .

2.3 Previous Research

The previous (and first try at) implementation of this game, Sync-In Team [3], was used as a research tool, used to aid in research toward synchronization of movement and music. The game also allowed researchers to study social interaction between dancing people, a phenomenon known as *entrainment* [4].

This game captured the movements of 4 players, divided into 2 teams, using accelerometers. The players were dancing to a series of audio tracks. Using a simple Fourier Transform based algorithm, the tempo of the movements was calculated, and compared to the Beats Per Minute (BPM) of the music.

A team whose tempo lay close to the BPM, got an increase in score, while the score decreased for a team whose tempo was too far off.

Scores were projected onto the floor, using growing and shrinking patterns in different colors, each color corresponding with a team, as pictured in Figure 3.

This approach had some limitations. The algorithm used to calculate the tempo was based on an FFT (Fast Fourier Transform), using a time-window of at least 4 seconds.

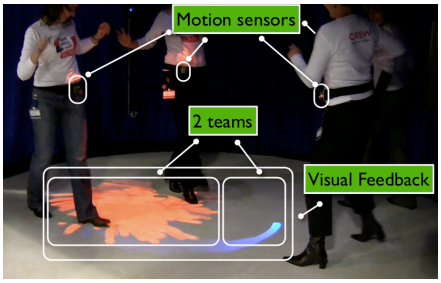


Figure 3. Sync-In Team score visualization using colored projection on the floor

This resulted in a poor response-time, and other limitations, such as no record of phase, or varying tempo.
The game was limited to 4 players, and 2 teams. Also, the manner of output for scores limited the players in space and movement, forcing them to look down.

2.4 Goal

Our goal is to resolve the issues that presented with the original Sync-In-Team Game, while developing a new and improved interactive social game, BeatLED.

The game should meet the following requirements:

- Allow at least 4, and preferably an arbitrary number of players to dance to a series of audio tracks, divided into a number of teams.
- Capture their movement data and synchronize it with the selected music, using a synchronization algorithm that performs in nearly real-time.
- Output a score in an original way, not limiting the movement space of the players.

This last aspect will be realized using the hardware developed at CMST, namely a flexible, stretchable LED-Matrix, embedded into a t-shirt.

2.5 Motivation & Applications

A social game like ours can have numerous applications in all fields. Next to personal entertainment, the game's social bonding and motivating characteristics could be applied to business, interpersonal relationships and even medicine. Applications could include team-building sessions (corporate or treatment-wise) or rehabilitation.

Apart from this, developing such a game represents an interdisciplinary challenge, and is bound to uncover techniques that can be applied in other applications.

3. TECHNOLOGIES

3.1 Accelerometers

In order to synchronize the movements of the players to the music, we need a device that captures these movements. For this we use accelerometers. An accelerometer is a device that measures proper acceleration (relative to free fall). Acceleration values are measured on three axes, so each movement direction can be represented, as can be seen in Figure 4, where the accelerometer is embedded in a Nintendo Wii remote. For use during the development stage, these Nintendo Wii remotes are chosen for their easy connectivity via bluetooth, their high availability and low cost.

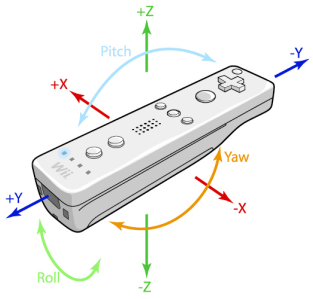


Figure 4. Nintendo Wii remote with acceleration axes x, y and z

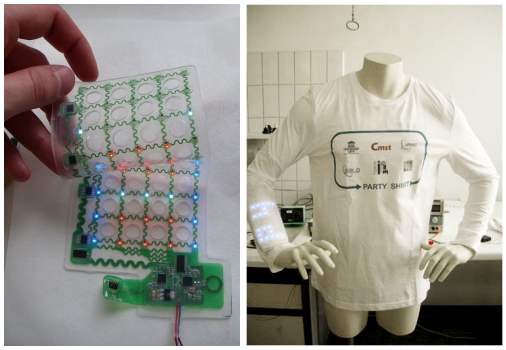
3.2 LED-Display

For the visual output of the game, a flexible, stretchable LED-matrix was developed at the Center of Microsystems Technology at Ghent University.

The novelty lies in the integration of the LED-display into textile, and the possibility to send data to this display wirelessly. So far, all commercially available textile that includes electronics lacks one or more of the key properties available with the CMST LED-Display. CMST has access to an in-house lab with great facilities and know-how, which were applied to obtain a stretchable LED display integrated in a T-shirt, as shown in figure 5(a).

The main focus of the design, beside stretchability, is on size and power consumption, since this is a battery application. All the electronics needed to control the display and the wireless communication need to be small in comparison to the display itself.

The final circuit design was assembled on a flexible design and molded in silicone, rendering it some rigidity, as shown in Figure 5(b). This mold could then be attached inside the sleeve of the game T-shirt.



(a) (b)

Figure 5. The finalized LED-display (a) silicone mold (b) embedded in T-shirt

4. SYNCHRONIZING MUSIC & MOVEMENT

4.1 Input Data Processing

Before any processing is performed, the input data needs to be shaped into a format which is usable by the synchronization algorithm. The acceleration data needs to be captured and filtered, and the music needs to be annotated, detecting the beats in the audio track.

4.1.1 Acceleration Data

The acceleration signal $A(x)$ provided by the accelerometers described in 3.1 is sampled at a fixed *sampling rate*, with each sample consisting of three real values, one for each axis, normalized between -1 and +1.

Due to the high sensitivity, the signal is bound to contain several involuntary, irrelevant movements, as well as some noise. To counter this, the signal is passed through a *low-pass filter* before any further processing. The simplest type of linear digital filter, a *Finite Impulse Response* (FIR) filter is used, designed to attenuate all frequencies higher than $10Hz$.

4.1.2 Beat Detection

In contrast to the original Sync-In Team, where only the average Beats Per Minute (BPM) of the song was calculated, our application's algorithm needs more specific data. This data includes the locations of each beat within the track. Because developing a custom algorithm for this is not the focus of this research, we opt to utilize third party software for this, namely Beatroot [5].

Beatroot tracks the onsets of all beats in a .wav file, and outputs these onset timings (in ms) to a plain text file, which can be read by our synchronizing algorithm. Once the audio is analyzed by Beatroot, the obtained data can be used to calculate the *average BPM*.

$$bpm = \left\lfloor \frac{60}{I_b} \right\rfloor \tag{1}$$

In this formula, I_b represents the *mean beat interval*, defined as the average time between 2 successive beats, calculated as

$$I_b = \frac{L}{B} \tag{2}$$

with $L = length\,of\,audio\,track\,in\,seconds$ and $B = number\,of\,beats\,detected$.

4.2 Synchronizing Algorithm

The data described in the previous paragraph is now used to associate a score to the synchronization between the movement data and audio. To do this, a newly developed *peak detection algorithm* is used.

4.2.1 Score Array

First, the beat data supplied by the beat detection algorithm (4.1.2) is converted to a *score array* S.

We associate a positive score with every sample of the audio track within a certain acceptable tolerance o before and after a beat. While doing this, the same sampling rate f_s as the accelerometers is used, thus creating an array with each element $S[t]$ corresponding to a sample of the movement data (with t the time of the sample). The acceptable tolerance is calculated as

$$o = \frac{I_{b\_samples}}{m} \tag{3}$$

where $I_{b\_samples}$ represents the average number of samples between two successive beats, calculated as $I_{b\_samples} = \frac{f_s * 60}{BPM}$ and m an adjustable parameter used to determine the tolerance in which a positive score is given. The further away from the actual beat position, the lower the associated score will be. The same principle is used for the *off-beat* locations, exactly between two successive beats, since people tend to synchronize to these as well. A simple example of the creation of the score array is shown in figure 6. Note that this illustration is simplified for visibility reasons. The tolerance in the figure is chosen quite high (100 ms) and the sampling rate very low at 20 *Hz*. In the real implementation the sampling rate would be much higher (ca. 100 *Hz*).

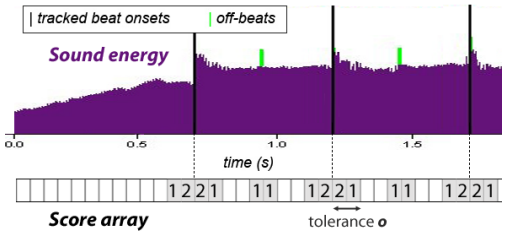


Figure 6. Creation of the score array from the detected beat and off-beat locations (simplified example)

4.2.2 Peak Detection

Next, the low-pass filtered acceleration signal is processed by a peak detection algorithm, for each axis. To detect peaks in a single accelerometer signal X (one axis) of fixed length, the following algorithm is used:

1. Calculate the average μ and the standard deviation σ of the sampled signal. (For computational reasons, an estimation for the average is used)

$$\mu = \frac{max(X) + min(X)}{2} \tag{4}$$

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=0}^{N}(X[i] - \mu)^2} \tag{5}$$

(Taking into account Bessel's correction for sampled standard deviation)

2. Create a binary form X_{bin} of the input signal with

$$X_{bin}[t] = \begin{cases} 1, & if\ X[t] > \mu + K.\sigma \\ 0, & else \end{cases} \tag{6}$$

with K a parameter adjusted along the amount of noise present in the signal. This process is illustrated in figure 7.

3. For every continuous interval where $X_{bin} = 1$ a peak is detected at the location in the middle of the interval.

This process is illustrated in figure 7.

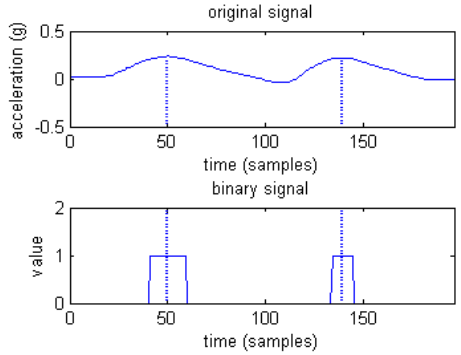


Figure 7. Original (filtered) acceleration signal (above) and its binary form (below), with detected beats represented by vertical, dotted lines

4.2.3 Scoring Algorithm

The final step is to assign a score to the detected peaks, depending on whether they correspond to a beat in the audio track. The complete synchronizing algorithm is described by the following steps:

1. Filter the input signal with the low-pass filter described above.
2. Buffer the signal into frames of $k.I_b$ samples.
3. For each frame of $k.I_b$ samples:
   (a) Detect a maximum of k peaks p_i , $i = 1, ..., k$ (only for the acceleration data in the current frame)
   (b) Store the location l_{pmax} of the highest peak p_{max}
   (c) The score is given by $s_{frame} = S[startindex_{frame} + l_{pmax}]$ with $startindex_{frame}$ the index of the first sample of the current frame, and S the score-array described above.

In this algorithm, k is a parameter specifying the length of the frame. In the proof-of-concept implementation, k is chosen as $k = 2$, meaning 2 beats can be expected each frame.

This algorithm is executed in parallel for each acceleration axis, detecting only k peaks, namely those leading to the highest scores.

5. GAME DYNAMICS

The game dynamics are an important factor to the gameplay and overall fun-factor of the game. For this project, a gameplay is implemented using *absorbing teams*, but thanks to the modularity of the synchronization algorithm, other gameplays are possible in future applications.

5.1 Absorbing Teams

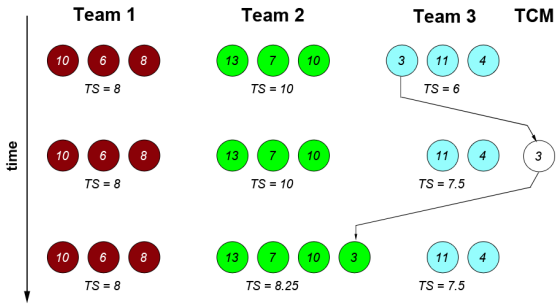


Figure 8. Example of Absorbing Team Game Dynamics: the player from team 3 is absorbed by team 2

The game mode we propose has all features of a social game as described in the beginning of this paper.

At the start of the game, all players are divided randomly into a user-specified number of teams, with a minimum of 1 player per team. It is clearly relayed to the players to which team they are assigned, using the color of the LED-display.

During the game, players are shown 2 scores: their *individual score* and their *team score*. The individual score represents how well this player is synchronizing to the music. The team score represents the mutual effort of the entire team a player is currently assigned to, and is calculated as the average of the scores of all team members.

After a predetermined time interval, the *lowest scoring player* of the *lowest scoring team* is removed from his/her team and transferred to *Team Change Mode (TCM)*. This will cause the team's average score to drastically improve (since they lost the low-scoring player).

After another, shorter time interval, the player in *TCM* is added to the *highest scoring team*, resulting in a drop in the team score of this team.

This way the highest scoring team figuratively *absorbs* the low scoring player from the lowest scoring team. This action then *equalizes* the team scores to a certain extent.

The game finishes when the music stops, or when all players have been absorbed by a single team.

An illustration of these game dynamics for 9 players is given in figure 8. In this example, the blue team (Team 3) is the lowest scoring team, and their lowest scoring player is removed from the team and set to TCM. This immediately affects the team score of team 3, changing it from 6 to 7.5. The changing player is then added to the highest scoring team, in this case the green team (Team 2), lowering the score of this team to 8.25.

5.2 Score Display

The scores are displayed using the LED-display developed at CMST. The outer rows represent the individual score, while the inner rows correspond to the team score, as shown in figure 9.

Scores are represented using a decimal system, with each colored led representing 10 points, and the position of the white led indicating the units. For example, in figure 9,

an individual score of 58 and a team score of 73 is shown. However, it is important to note that players shouldn't concern themselves with these scores, they are merely intended as an indication of how well they are doing.
Players should only make the simple association:
$more\ color = better\ score$.

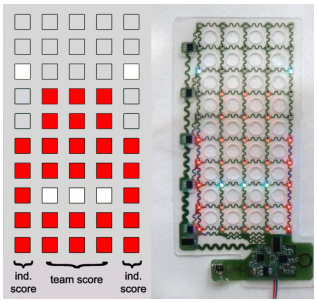


Figure 9. Scores: schematic (left) and real display (right). An individual score of 58 (outer 2 columns) and a team score of 73 (inner 3 columns) is shown.

6. PROOF-OF-CONCEPT IMPLEMENTATION

6.1 Technical Details

For the practical implementation of BeatLED, Java was chosen as a programming language, because of its portability, compatibility and ease of GUI design.

As mentioned before, the accelerometers embedded in Nintendo Wii remotes were chosen, which were easily interfaced via Bluetooth. However, because Wii remotes send their acceleration data in *bursts*, a resampling module had to be written, and was applied before inputting the data into the game.

Because the game was designed in parallel with the LED-display, the flexible visualization matrix was not immediately available. We opted to show preliminary output using projections on the ground, and later using rigid versions of the display.

6.2 Progress & Results

We succeeded in creating and testing a game with up to 10 players, and 5 teams. In fact, with the current game design, the number of players and team can be arbitrarily chosen at the start of the game, limiting the maximum number of players to the amount of Wii remotes that can be connected.

Because Bluetooth only supports up to 7 devices on 1 machine, we integrated the Open Sound Control [1] (OSC) protocol into the application. This way, additional accelerometers could be connected to a different machine, and transmit the acceleration data through a network connection to the main gaming machine.
This also allows for easy integration with other types of accelerometers. Future developers could easily create a separate module to send (correctly formatted) data coming from different accelerometers over the network to the existing game.
We also opted to send the scores in the OSC format, to allow for easy connectivity of additional score displays, and improve scalability and modifiability. Figure 10 shows a schematic view of the set-up, where one laptop is used for I/O, and one for processing. Note that there can also be one all-in-one machine, or several I/O computers and one processing computer.

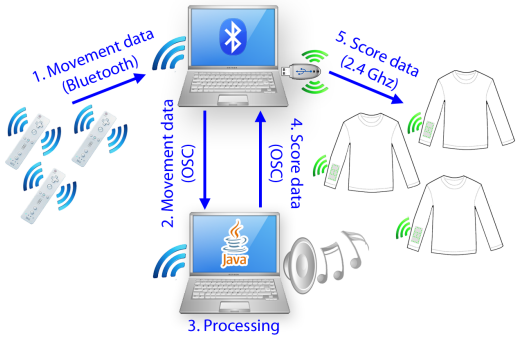


Figure 10. Schematic view of the data-chain using Open Sound Control (OSC). (During the game, the wii remotes are attached to/held by the players.)

Three different synchronizing algorithms were tried out: the original FFT algorithm, an algorithm based on adaptive oscillators, and a peak detection algorithm (described above). The peak detection proved to be the best performing and was left in the implementation. Its parameters were optimized, ensuring a realistic representation of the level of synchronization of music and movement.

All components are modular and can be interchanged with new implementations, leaving room for future adaptations of the game.

7. TESTING

Two types of tests were performed in order to evaluate the proof-of-concept implementation. First, the accuracy of the synchronizing algorithm had to be determined. When the final implementation was completed, a series of test sessions were organized, allowing real end users to play and evaluate the game.

7.1 Algorithm Tests

The accuracy of the peak detection was tested using the *peak detection error rate*, defined for a fixed length frame of samples i as

$$E_{peaks}(i) = \frac{|N_{detected}(i) - N_{annotated}(i)|}{N_{annotated}(i)} \qquad (7)$$

with $N_{detected}(i)$ the detected number of peaks in the frame, and $N_{annotated}(i)$ the number of peaks visible in the data.

We obtain the *average peak detection error rate* by averaging these results for all frames of a set of test data.

$$E_{peaks-avg} = \frac{1}{F}\sum_{i=1}^{F} E_{peaks}(i) \qquad (8)$$

[1] http://opensoundcontrol.org/

A test set of accelerometer data was used, sampled at 200Hz with movements at varying tempo, with each visible peak carefully annotated. The results are visible in table 1. We can clearly discern that even at a frame size of 200 samples (corresponding with 1 second), usable results are obtained. This is clearly an improvement over the original FFT algorithm, which required a frame size of 4 seconds.

frame size (samples)	100	200	400	800
$E_{peaks-avg}$	-	3.9%	1.5%	1.8%

Table 1. Average error rate of detected number of peaks for data with varying tempo

7.2 Test Sessions

7.2.1 During development

During development, monthly test sessions were organized, using the latest version of the soft- and hardware. User input influenced the following design decisions:

Location of the motion sensors It became quickly apparent that users who held the motion sensor in their hand moved significantly less than users with the sensor attached to their body in an unobtrusive place. This was applied in all later test sessions.

Game and song duration Using trial and error, an average length of 1 minute was chosen for the audio fragments in the final implementation. User input showed that a sequence of 4 tracks proved ideal.

Cumulative score vs. Sliding window A choice can be made between a *cumulative score* and a *sliding window* score. Cumulative scores are simply added to a player's previous scores, throughout the entire game, while a sliding window calculates a player's score as the average of his/her last N scores, with N the window size. The choice of scoring method greatly affects the strategies applied by the players. Users were inconclusive about which they liked best.

7.2.2 With finalized software

The tests with the final proof-of-concept software were carried out using a less costly, rigid version of the LED display, allowing us to test the final game thoroughly, accurately and without risk of damaging the more expensive flexible demonstrators.

A short questionnaire (10 questions) was presented to the users of the final test sessions. Although it was filled in by a limited number of people, we were easily able to discern some trends. The answers clearly indicated that most aspects of the game are positively received. User opinions were very positive about the *game rules*, *feedback delay*, *game and audio duration* and *team change speed* . However, they were rather divided about the *score calculation* method (as mentioned above) and the *score visualization*. While half of the users thought the scores were clearly presented, the other 50% thought the exact opposite. This shows that there might be a need to further examine the score representations.

8. FUTURE WORK

In the future, research in this topic could be continued by exploring other algorithms for synchronization, and by inventing new game modes. The results and opinions gathered from further test-sessions can be used to improve user experience.

Our software allows musicologists to research natural synchronization of people with music and each other, in a social context. More players could be added, and one could even imagine a mass-player game, where real results about social behavior would become apparent.

The LED visualization could be further expanded, firstly by adding more LEDs, and secondly by improving wireless communication, allowing us to show more complex images or even video. The user feedback indicated that a simpler visualization, which is easier to understand, might also improve user experience.

9. CONCLUSIONS

We developed a viable and usable social game, which meets all of the discussed requirements. We used readily available equipment, combined with state-of-the-art, newly developed technology.

While this is a proof-of-concept implementation, we showed that a social game such as ours could offer many possibilities, not only for research, but also in the entertainment sector and even health care.

Acknowledgments

We would like to thank all participators in this project, especially the supervisors at IPEM, MMLab and CMST. Also, special thanks are in order to the Belgian Industrial Research & Development (BiR&D) committee for their financial support, allowing this project to become a reality.

10. REFERENCES

[1] N. Games, *Video Gamers In Europe*. Interactive Software Federation of Europe (ISFE), 2008.

[2] S. B. Yang, S. and G. Graham, "Healthy video gaming: Oxymoron or possibility?" in *J. of Online Education - Vol. 04*, 2008.

[3] M. Leman, M. Demey, M. Lesaffre, L. van Noorden, and D. Moelants, "Concepts, technology, and assessment of the social music game "sync-in-team'," in *Proceedings of the 2009 International Conference on Computational Science and Engineering - Vol. 04*, Washington, DC, USA, 2009, pp. 837–842.

[4] M. Leman, *Embodied Music Cognition and Mediation Technology*. The MIT press, 2007.

[5] S. Dixon, "Evaluation of the audio beat tracking system beatroot," in *Journal of New Music Research - Vol. 36*, 2007, pp. 39–50.