The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

MagicTiles. ALife for Real and Virtual RoboMusic

Luigi Pagliarini[1,2] Henrik Hautop Lund[1]

[1] Centre for Playware, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark
[2] Academy of Fine Arts of Macerata, Via Berardi, 6, 62100 Macerata, Italy

luigi@artificialia.com hhl@playware.dtu.dk
www.playware.dk

Abstract

In this paper, we define and trace the contours of a new cross-modal and cross-media approach to robotic systems. In particular, our system is based on the use of electronic tiles and oriented to the use of ALife based interactive robotic modules that can be either virtual or real. This approach, which relies on interactive parallel and distributed processing algorithmics, can be thought and used on many different domains, though in the specific experience presented here, it has been applied to music composition and remix. In details, we describe the initial MagicTiles product prototype and show, as an example, a musical application with which any user can create and perform RoboMusic. We show how the combination of the Modular Interactive Tiles and the MagicTiles tools might lead to a broader vision of robotic systems with a fluid flow between the physical and virtual. Fluidity between the physical and virtual, end-user authoring, and ALife control is conceived to open up technology to ordinary users as a tool for creativity. Finally, in this paper, we attempt to explore the theoretical characteristics of such an approach and exploit the possible playware application fields.

Introduction

In this paper we present the "MagicTiles" application, an ALife software based on and derived from our previous work on the Modular Interactive Tiles [1] and RoboMusic [2]. This attempt to move a hardware based paradigm to a software based one is due to a couple of main beliefs.

The first of these is that the social media are changing and are showing a strong demand for a mobile electronics representation (i.e. applications) of any given device and/or activity. Indeed it seems to be expected that any future playware [3] and/or robotics tool will require - for a proper and common use within a large community of people - to get somehow connected to a mobile phone, tablet (or similar) control system or, alternatively, have a given instance of it, as for example a simulated version of the tool/activity itself. This is because the new virtual media may allow users an easy access to social interaction, for instance mediated by the physical media.

Hence, the physical media is enhanced by the social, virtual media. On the other hand, the social, virtual media is also used to promote the users' interest in the physical media. Indeed, today users' access to the virtual is easy and cheap, and it allows the users to experience their own interest for the specific real technology (e.g. the physical media). Thereby, there is a bidirectional use of the social, virtual media (on smart phones, tablets, etc.) and the physical media, which can be seen as complementary, and to be integrating and promoting each other.

The second belief is that by realizing a software-based representation of the Modular Interactive Tiles, we can easily explore many different new algorithmic solutions and easily access a larger number of users. Therefore, it gives the chance to test both efficiency and popularity of our "games" and the concept of Modular Interactive Tiles itself.

As a starting point, we chose music as the application domain, building upon the concept of RoboMusic.

RoboMusic

The RoboMusic concept [2] was developed to allow any user to interact with a professional music performance through the interaction with physical, intelligent objects. For instance, in the first RoboMusic concert, the interaction happened with modular interactive tiles, rolling pins and cylinders [2] (see Fig. 1), and the Center for Playware later made RoboMusic versions using cubes [4, 5] and tiles. When interacting with these physical, intelligent objects, the user activates, changes or de-activates a sound (Fig. 2).

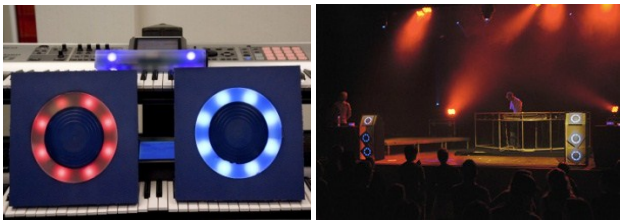

Figure 1. Left: Two Tiles and a RollingPin used as robotic instruments. Right: The RoboMusic live concert set-up, with Funkstar De Luxe and his control station in the center, and the robotic instruments on the left and right side of the stage.

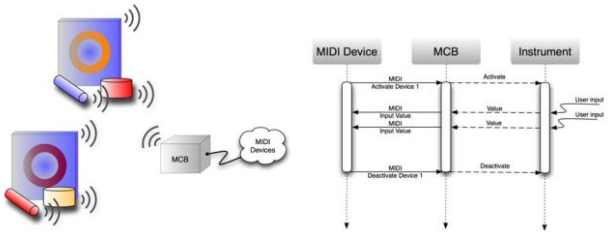

Figure 2. For the first RoboMusic concert, we used an interaction pattern in which physical objects as tiles, rolling pins, and cylinders ("Instruments") are communicating to a MIDI Control Box ("MCB"), which is connected to a PC running Cubase or Ableton Live ("MIDI Device"). Note that the original RoboMusic concept allows also communication in the direction from PC ("MIDI Device") to tiles ("Instruments") through the MIDI Control Box ("MCB").

The sound can be more or less complex. In order to produce an appealing sound, it is of crucial importance to create a RoboMusic soundscape which fits the RoboMusic concept of allowing the user to interact with the music, e.g. as in the behavior-based approach, where the programmer is designing the default music behavior, and the user provokes smaller behavioral deviations by the physical interaction [2].

Indeed, as a kind of *user-guided behavior-based system*, in RoboMusic, the design challenge is to create primitive 'robotic' behaviours and to coordinate these primitive behaviours in order for the music piece to emerge as the coordination of primitive behaviours. Thereby, a music composition emerges from the way the composer, musicians or audience interact with the 'robotic' instruments that provide the primitive behaviours.

Each 'robotic' instrument is used to trigger a particular primitive behaviour dependent on the interaction with the instrument(s). In RoboMusic, the primitive behaviours can be anything from a volume or a cut-off to a small sequence of tones. The music composer designs the way in which the primitive behaviours that are triggered should interact with each other.

Hence, as is the case when designing behaviour-based robots such as mobile robots (e.g. [6, 7]), the robot designer (in this case the music composer) designs the primitive behaviours and the coordination scheme. And, as is the case with *user-guided behaviour based robotics* [8, 9], if non-expert users (e.g. live concert audience) are supposed to manipulate and become creative with the systems, it is crucial that the designer (music composer) creates primitives on a fairly high abstraction level that allows the non-expert user to understand and have positive feedback from the human-robot interaction within a very short time frame.

The RoboMusic concept was applied on cubic I-BLOCKS [4, 5] which each represents an instrument or group of instruments. The individual I-BLOCKS orientation - which side is facing down - determines the variation of that specific instrument. The I-BLOCK LEDs change colour depending on their orientation, in order to make it possible for the user to remember and activate specific variations.

The musical setup can be seen in Figure 3. Note the black XBEE-enabled I-BLOCK, which communicates wirelessly with the MIDI Box. When instrument blocks are connected to this, music starts playing depending on the actual I-BLOCK's colour and orientation. The music is loop-based, meaning that when active, each variation of each instrument is playing for a certain time and then repeating itself over and over until it is finally deactivated when the user removes the current instrument I-BLOCK from the structure or shifts its orientation.

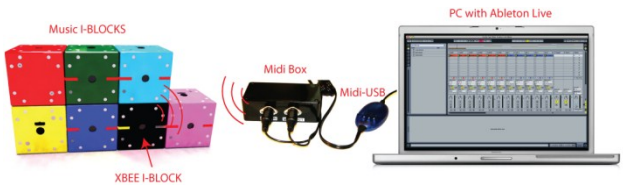

Figure 3. Music setup with I-BLOCKS, MIDI-box and PC.

The pieces of music that was made for the cubic I-BLOCKS were all constructed using these rules: There are five or six predefined instruments or groups of instruments, varying in type according to genre, and within each piece of music there are up to six variations per instrument type, and there can be an unspecified number of different instruments. Later, we also applied this approach to other intelligent building blocks such as the modular interactive tiles [1], where the color of the tiles LEDs would indicate the instrument, and the number of LEDs indicates the variation of the instrument type.

MagicTiles Interface

The MagicTiles application is a new, funny, musical app, which as RoboMusic can be used by users of different musical competencies - from precipitants to musician to expert composers - to both remix songs or experiment with new tunes. The initial MagicTiles app runs on iPads and iPhones. The MagicTiles application is made of a small number (i.e. more than 3 less than 10) of virtual music tiles. Generally, each single tile embodies a single instrument (although it can be programmed and used in many different ways), and each instrument-tile is capable of expressing seven different musical solution (i.e. loops) or variation of the same one. Every instrument of a MagicTiles song is wrapped inside a single tile and, therefore, we can have a tile for vocals, one for the rhythm guitar, one for the bass, one for the piano, etc.

Figure 4. MagicTiles for iPad.

At the start, the user is presented with all the tile-instruments of a given song inside the tiles board with the tile-instruments being disconnected from each other (see Fig. 4). Since a tile is disconnected from other tiles, it is not active and will not play.

Once the user drags and attaches a tile to another (see Fig. 5 left) the two tiles get synchronized and are able to play together.

To actually start the music, the user has to press the play/stop icon, positioned at the bottom-right side of the screen (see Fig. 4). At this point, the music will keep playing and the remaining music tiles can be connected to the former two enriching the music scenario of the song running. In the same way, by dragging the tiles around and rearranging their geometry, the user changes the music sequence and flow.

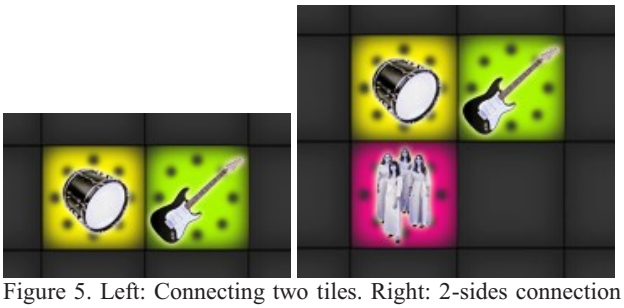

Figure 5. Left: Connecting two tiles. Right: 2-sides connection example.

Every tile has 7 music loops, one for each side (north side, east side, south side and west side) plus 3 extra music loops which will play when a tile is connected on, respectively, two, three or four sides. In this way, as in Fig. 5 left, the Drum tile will play the East Side loop and the Guitar tile will play the West Side loop, and as in Fig. 5 right, the Drum tile will play the *2 Sides* loop, the Guitar tile will play the West Side loop and the Choir tile will play the North Side loop.

Likewise, as in Fig. 6 the Drum tile will play *3 Sides* loop, the Choir tile the North Side loop, the Guitar tile will play the West Side loop, and the Bass tile the East one.

Figure 6. A *3 Sides* connection example.

Growing the number of instruments/tiles impressively increase the number of possible combinations and out of these simple rules, one obtains an extraordinary number of musical possibilities (e.g. by using 6 tiles, and just counting the 1-side connections, we reach 4^(6-1)=1024 music variations of the same song).

MagicTiles' usability has already been pilot-tested with 5 years old children and can be considered simple to use, and to be an effective and funny instrument to play with. Further, this first version of the MagicTiles app has two more exciting features, which enhance the user

interactivity and possibility: the *Loop Recorder* and the *SkipBar Tool*.

The Loop Recorder

The previous section described the so-called *Pre-recorded* loops tiles, trying to understand their meaning and logic. Differently from that, MagicTiles also features two other kinds of music tiles, the *Pre-recorded* loops tiles and the *User-recording* loops tiles (see Fig. 7).

The *User-recording* tiles differ from *Pre-recorded* tiles for two specific characteristics:

1. The music loops are not given and have to be recorded by the user (e.g. at run-time);
2. The number of different loops a *User-recording* tile can express are only four (i.e. north, east, south and west side loops).

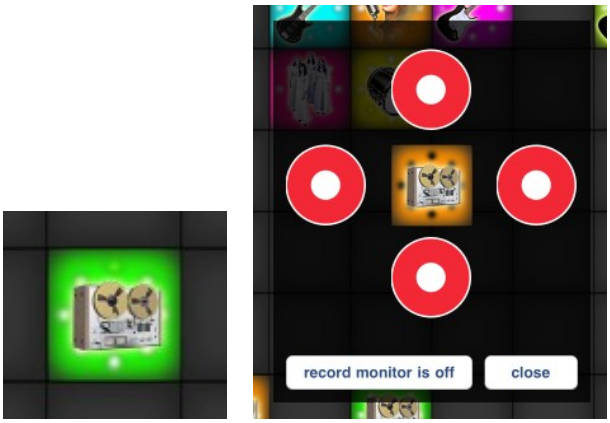

Figure 7 Left: The *User-recording* tile icon. Right: The loop recorder menu.

When the user double touches one of the User-recording tiles (see Fig. 7 left), the MagicTiles app shows the loop recorder menu (see Fig. 7 right). The loop recorder menu allows the user to record a loop for each side of the user-recording tile through the iPhone or iPad built-in microphone. The application shapes the recordings to exactly fit to the length of all the other pre-recorded tile loops of the actually running song. Such process gives the user the great advantage of always being on the beat. As a consequence, the ability of recording a proper music sequence/loop will simply rely on the user musicality, itself. Technically speaking, such a feature works at the lowest possible latency, because it relies on the powerful audio framework that the native iOS devices comes with.

Another tool for user interactivity is the SkipBar Tool that allows the user to skip the current (playing) bar and to repeat it or use it like for real sampler. The bottom of Fig. 4 shows the SkipBar with 8 buttons/areas (i.e. one for each single bar). When pressing one of them, the user let the music flow to jump or repeat one single bar music.

MagicTiles Algorithms

The Sound Patterns

Every single tile can activate seven different music loops. Each sound is activated accordingly to the following rules.

If only one side of the tile is connected, the tile itself reproduces one out of four available loops, which is:

1. One for the North side
2. One for the East side
3. One for the South side
4. One for the West side.

Besides the above sounds, there are three extra music loops, which are activated when:

5. Two sides of the given tile are connected simultaneously;
6. Three sides of the given tile are connected simultaneously;
7. Four sides of the given tile are connected simultaneously.

(See Fig. 5 and Fig. 6 for visual examples).

Sound Files Specifications

To be used with the MagicTiles app, sound files must be:

1. of an uncompressed sound format (e.g. AIF or .WAV);
2. same length (at the moment, 8 bars for every musical loop, independently from the song metronome).

Further, the sound files should preferably only be *monophonic files*, since stereo files may demand too much memory.

Loops may be denominated in any way. For example, a good practice would be to name loops with a specific *Song/AuthorName_* plus *InstrumentName_* plus the *SideNames*, to keep track of the musical meanings, as shown here:

The Drums Tile of the Bob Marley Remix files could be denominated:

1. MARLEY_DRUM_N
2. MARLEY_DRUM_E
3. MARLEY_DRUM_S
4. MARLEY_DRUM_W
5. MARLEY_DRUM_2
6. MARLEY_DRUM_3
7. MARLEY_DRUM_4

Meaning that the first 4 loops are denominated accordingly to the activated side, and the remaining 3 loops are the extra loops for "2 sides connected", "3 sides connected", "4 sides connected".

Synchronization of audio files

With MagicTiles, we developed a novel way for getting rid of a metronome. First of all, we made the music loops have a fixed length (i.e. a parameter that you can tune inside your audio editor, like Ableton Live, Logic, Cubase, etc.). Every single loop is stored inside a *C* programming language structure variable defined with the keyword "*struct*" that contains some variables. As in Fig. 8 one of those variables is sampleNumber. This var contains the current sample number of a music loop, this *struct* is filled during the app startup where all the files are red and stored into memory. The sampleNumber is updated during playback continuously.

```
// Data structure for mono or stereo sound, to pass to the application's render callback function,
//    which gets invoked by a Mixer unit input bus when it needs more audio to play.
typedef struct {
    BOOL            isStereo;         // set to true if there is data in the audioDataRight member
    UInt32          frameCount;       // the total number of frames in the audio data
    UInt32          sampleNumber;     // the next audio sample to play
    AudioUnitSampleType  *audioDataLeft;    // the complete left (or mono) channel of audio data read from an audio file
    AudioUnitSampleType  *audioDataRight;   // the complete right channel of audio data read from an audio file
    BOOL            isRecording;
    UInt32          startSample;
} soundStruct, *soundStructPtr;
```

Figure 8. The audio files C struct.

Now, suppose that the drum tile is playing and we want to add the guitar tile. When the user connects the guitar tile to the drum tile, it is known that the drum tile is playing the "x" sample number, so the system simply updates the sampleNumber variable inside the guitar struct to get them synchronized.

ALife in MagicTiles

The MagicTiles app is both inspired and aiming at the use of ALife paradigms. Indeed, as software conception it was deeply inspired by cellular automata research, from which it inherits part of the cells interactive logic. The MagicTiles algorithm could be properly defined as a User Based Interactive Cellular Automata algorithm where - for the moment being - the basic cell rules are predefined (see the MagicTile Algorithm section for further explanations). In this cellular automata like approach the *User* - intended either as the one who "composes the musical units" or the one who "manipulate the software" by arranging them in the virtual space – plays a major role in leading the (musical) output flow.

Further, the MagicTiles conception effectively derives from Music Cubes and Modular Interactive Tiles, and by doing so, it largely expresses and inherits principles coming from embodied artificial life. As a consequence, the musical outcome the MagicTiles, as for the Embodied ALife algorithms, implements a form of intelligence without either a static or a physical representation. As Brooks enounced [10], in MagicTiles the so-called musical intelligence is approached in an incremental manner, and it is given a strict reliance on interfacing to it through perception and action, letting the reliance on representation gradually disappear.

Apart from improving the interface and the interactivity, as an experimental platform the MagicTiles opens up for future research that goes deeper in incorporating ALife paradigms. An intelligent serve-routine may be defined to learn from the user activity and to build a user profile – a personalized fitness formula – which may facilitate users to manipulate music elements and reach their desired goal. In addition, the MagicTiles app can be introduced as a social component that by connecting its own community of users - through the use of classical interactive genetic algorithms – may implement a refined form of collective Evolutionary Music.

Discussion and Conclusion

In this paper, we presented a new RoboMusic application, MagicTiles, which aims at exploring a new cross-modal and cross-media approach to robotic systems. Through such an application any user can create and perform RoboMusic by assembling a number of interactive virtual tiles on the screen. After our past efforts in building physical interfaces, we decided to conceive the MagicTiles platform to show how the combination of the physical interface (i.e. Music Cubes and Modular Interactive Tiles) and the virtual interface (i.e. MagicTiles) tools might lead to a broader vision of robotic systems with a fluid and bidirectional flow between the physical and virtual. Indeed we believe that such a fluid and multifaceted representation of a single tool/activity may widely enhance the user immersion into a *one reality* that combines the physical and virtual.

It is interesting to observe a similar trend in robotics with the current development of cloud robotics, which combines the physical with the virtual. This may be in the form of physical robots connected to the cloud, which performs the computing such as object recognition, speech recognition, etc. for the physical robot.

Differently from most other modular interactive systems, the MagicTiles application allows end-user authoring of the contents of the individual modules, therefore enabling a high level of personalization – that even can be applied run-time – of the outcome. Our concept is inspired by ALife control systems (e.g. Cellular Automata, Embodied AI, etc.) and aims at using further traditional ALife resources to open up to single and ordinary users, as well as, building a new collective and evolutionary tool for creativity.

Acknowledgements

The authors wish to thank Massimiliano Leggieri and Andrea Gabriele for the help in developing the first prototype system. Also thanks to the Centre for Playware colleagues for valuable discussions.

References

[1] H. H. Lund. Modular Robotics for Playful Physiotherapy, in *Proceedings of IEEE International Conference on Rehabilitation Robotics*, IEEE Press, 571-575, 2009.
[2] H. H. Lund, and M. Ottesen. RoboMusic – A Behavior-Based Approach, *Artificial Life and Robotics Journal, 12: 1-2*, pp. 18-23, 2008.
[3] H. H. Lund, T. Klitbo, and C. Jessen. Playware Technology for Physically Activating Play, *Artificial Life and Robotics Journal, 9:4,* 165-174, 2005.
[4] J. Nielsen. User Configurable Modular Robotics - Control and Use, Ph.D. thesis, University of Southern Denmark, Odense, 2008.
[5] K. Falkenberg, N. K. Bærendsen, J. Nielsen, C. Jessen, H. H. Lund. RoboMusic with Modular Playware. *Artificial Life and Robotics, 15:4*, 369-375, 2010.
[6] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation, 2(1):*14--23, 1986.
[7] R. C. Arkin. *Behavior Based Robotics*, MIT Press, Cambridge MA, 1998.
[8] H. H. Lund. Modern Artificial Intelligence for Human-Robot Interaction. *Proceedings of the IEEE journal, 92:11*, 1821-1838, 2004
[9] H. H. Lund. *Robots at Play*. ISBN 978-87-992302-0-4, 160 pages, Odense, 2007.
[10] R. A. Brooks, Intelligence without representation, *Artificial Intelligence 47*, 139–159, 1991.