Using MIDI to Modify Video Game Content

Jukka Holm
Nokia Research Center
Visiokatu 1
33720 Tampere, Finland
jukka.a.holm@nokia.com

Juha Arrasvuori
Nokia Research Center
Visiokatu 1
33720 Tampere, Finland
juha.arrasvuori@nokia.com

Kai Havukainen
Nokia Technology Platforms
Visiokatu 1
33720 Tampere, Finland
kai.havukainen@nokia.com

ABSTRACT

This paper discusses the concept of using background music to control video game parameters and thus actions on the screen. Each song selected by the player makes the game look different and behave variedly. The concept is explored by modifying an existing video game and playtesting it with different kinds of MIDI music. Several examples of mapping MIDI parameters to game events are presented. As mobile phones' MIDI players do not usually have a dedicated callback API, a real-time MIDI analysis software for Symbian OS was implemented. Future developments including real-time group performance as a way to control game content are also considered.

Keywords

Games, MIDI, music, rhythm games, background music reactive games, musically controlled games, MIDI-controlled games, Virtual Sequencer.

1. INTRODUCTION

Music has an important role in contemporary video games. It can help to make a right kind of atmosphere for gaming, and emphasize actions on the screen. It is common that the background music is adaptive i.e. it changes according to game events, between different parts of the game, and is synchronized to the game actions. As an example, when an avatar is moving in a safe area the music may be slow and relaxing, but during an enemy attack it becomes faster and more aggressive.

The development of an adaptive music soundtrack and sound effects for a modern video game is an expensive and time-consuming process. Due to this, many games just loop the same relatively short music files over and over. Some developers have started using songs from popular artists as background music in their games. Repetition has also its cost: The gamer may become bored with the non-adaptive soundtrack and turn it off after a while. The study by Cassidy et al. [19] suggests that the best player experience emerges when a player can choose a game's background music to something that he or she prefers. The scope of that study was limited to driving games.

Since 1990's, we have also seen the rise of musically oriented games. As Blaine points out in [14], the majority of these are so called "rhythm games" that prompt a single player or a group of players to perform rhythmic actions in time with a predetermined musical sequence. The game genre has also led to the development of new low-cost interfaces such as drum, guitar, and dancemat controllers that make the games more enjoyable to play. Many rhythm games suffer from the same problem as non-adaptive game soundtracks: As the number of music files is limited, the games may have quite short lifecycles. To improve the situation (and earn more money), some developers have started offering game upgrades such as catalogues of popular songs.

This paper describes a novel way to use music in games. Instead of adaptive background music that reacts to the game events, the authors propose the concept of games that react to their background music. Throughout this paper, this kind of games are referred to as "background music reactive games." In [17] and [18], the authors have also used the term "musically controlled games." The idea works both in the case of rhythm games and non-adaptive background music soundtracks.

The contents of this paper are as follows: Chapter 2 discusses some relevant previous work, Chapter 3 presents an overview of the idea, and Chapter 4 justifies using MIDI to control game content. Chapter 5 lists some interesting MIDI parameters and ways to use them to modify games. Chapters 6 and 7 discuss authors' test platform "AudioAsteroids" and MIDI analysis software for Symbian mobile phones. Finally, Chapter 8 draws some conclusions and Chapter 9 suggests some topics for future work.

2. PREVIOUS WORK

When thinking about "music games", most people are probably referring to "rhythm games." In PlayStation game "Parappa the Rapper" (NanaOn-Sha 1997) [9] and other similar rhythm games, the player has to trigger musical events by pressing specific buttons to the beat of the music. In "Gitaroo Man" [3] for PlayStation 2, the player must also follow the pitch of lead instrument with the joystick. In "Mad Maestro" [7] for PlayStation 2, the "appropriate" tempo of each song determines the speed with which the player must press the buttons. In all of these games, the sets of songs are fixed and the programmers have had to define the occurrence of each sequence in every song. In other words, the generation of game levels from pieces of music is not automatic.

In [14], Blaine discusses alternative game controllers that have been used with music games. These controllers border double-function as musical controllers. Experimental hybrids between games and new types of multi-user musical controllers, including Jam-O-Drum, Jam-O-Whirl, and Jam-O-World, have been discussed in [16].

One example of a video game in which music becomes the result of players' actions, that in themselves are unrelated to the theme of making music, is Shockwave game "BLiX" [10]. "Rez" (Sonicteam 2001) [11] for PlayStation 2 is a third-person shoot 'em up game with drum samples instead of conventional weapon sounds. When the player shoots, the triggering of samples is quantized so that they always match the background music rhythmically. It could be said that in games such as BLiX and Rez, player's actions on top of the background music produce a certain soundtrack along each play session. However, these games cannot be played with background music of player's own choice.

A couple of games that respond to the pitch of player's whistling, singing, or humming have been implemented. In many cases, the pitch is rounded to the nearest semitone. Hämäläinen et.al. [4] discuss games controlled by singing. One of their examples is a pitch-controlled Pong. The authors also discuss the technical and psychoacoustic issues of pitch detection. In karaoke game "Staraoke" (Intervisio 2003) for Windows 98/XP, the pitch of background song's melody forms a path. The player must guide his or her character through the path by singing the melody correctly. Pitch is represented on the vertical axis and time on the horizontal axis. [2]

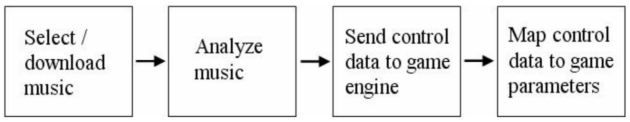
Of all music games on the market, the one that most closely resembles ideas presented in this paper is Playstation game "Vib-Ribbon" (NanaOn-Sha 1999) [8]. At the time of its release, Vib-Ribbon was welcomed as a refreshingly new kind of game. It is best described as an obstacle track game, in which the background music (any song from any audio CD) affects the appearance of obstacles, the points in time when these obstacles appear, and the spawning of certain additional objects. Player's character seems to walk along the stylized waveform. One issue with Vib-Ribbon is that the correspondences between characteristics of music and obstacle track are not that obvious for casual players. The obstacle track just appears different with different pieces of music.

In addition to audio-control, a small number of games that use MIDI controllers for input have been developed. David Bagno's "Musical Space Invaders" and "Music Scale Teacher" [6] for Windows 98/XP and Macintosh are note teaching games played with a MIDI keyboard. "Musical Invaders" [5] for Windows 98/XP is a music learning game that responds to real-time MIDI input. The goal is to play the notes as they appear on the screen. By doing this, the player performs a melody at the same time. Players can also load their own MIDI files and play them as game levels.

3. OVERVIEW

Starting a "background music reactive game" differs somewhat from traditional games. In the beginning, the player must first select a music file or collection of files to be used in the game. The music is analyzed for relevant musical parameters (see Chapter 5) either in real time, in larger buffers, or the whole song can be processed before the game starts. The resulting control data is then sent to the actual game engine, which maps it to selected game parameters. After this, the player starts the game and tries to play through as many files as possible. Depending on the game type and implementation, each file may be considered as one unique game level.

When music is analyzed to produce control parameters for the game, novel ideas can be found. Even a very trivial game can be made interesting if the player can affect the difficulty level by changing the background music to his or her favorite tune. A player may end up saying things like: "I passed the game with Queen's Show Must Go On, but Steep's Rise is far too difficult for me!" In the case of mobile phones, even ringing tones could be used to control game parameters.



Select / download music
Analyze music
Send control data to game engine
Map control data to game parameters

Figure 1. Starting a background music reactive game.

The concept also offers many imaginative possibilities for the game designers. As an example, consider a submarine combat game. It is possible to use music to control the behavior of the enemy so that it can shoot only when a certain note in the background song is played. Also elements not essential to the gameplay can be controlled by the music, for instance, the underwater plants may sway in the rhythm of the background song. In conventional game implementations, these actions would be controlled by a random generator or artificial intelligence (AI). Through music-control, the actions do not appear random.

Examples of musically controllable game elements and characteristics include e.g.:

- Speed and difficulty level of the game;
- Location of game objects (enemies, ammo, guns, bonus objects, buildings, etc.);
- Number, size, type, color, and shape of objects;
- Time and frequency of appearance of new objects;
- Movement (e.g. speed, direction, rhythm, starting point, trajectory) of objects;
- Properties of avatars (e.g. skills and endurance);
- Context of gameworld (e.g. location of game events, time of day); and
- Camera angle.

4. BENEFITS OF MIDI

With the current technology, MIDI has certain advantages over other file formats such as Wave and MP3. As MIDI is a symbolic format, it is more precise to use to control game content than digital audio. Certain MIDI parameters such as Note On messages can be directly used as game control parameters, while more control parameters can be found by making some simple calculations based on MIDI events for instance on a specific channel.

The file size of MIDI is considerably smaller than that of digital audio. Because of this, a larger amount of files can be stored to e.g. a portable device and new files can be rapidly downloaded over the air. Due to the recent increase in storage space and computational power, there are no longer strict limits to the size of wavetable soundbanks and sophisticated synthesis methods such as physical modeling can also be used to generate the sounds.

MIDI has also some important benefits if compared to other symbolic audio formats such as Open Sound Control [1]. The format is widely spread so there are lots of software and hardware tools available. People can easily download new songs from the Internet, and in the case of mobile phones even ringing tones can be used to create new game levels.

5. MAPPING MIDI EVENTS TO GAME PARAMETERS

Mappings between musical control parameters and game parameters are most effective when players immediately understand the relationship between what they hear and what they see. If a player knows a certain piece of music well and understands the mapping used in the game, he can anticipate some of the actions that will occur in the game.

In order to support a large number of players having different musical tastes, the mapping should not be tailored to a single genre. There are undoubtedly interesting differences between musical styles, but in most cases the game designer should aim at a generic mapping that works nicely in the case of any genre. It is beneficial if distinct musical genres produce distinct game experiences, but this should not be done at the cost of the general playing experience.

While most MIDI parameters affect how the music sounds like, some may not be noticed by inexperienced listeners. Parameters that are understood by most players should usually be connected to major foreground events in the game, while the not as obvious ones can be used to control less important things like background graphics and so on.

For the purposes of background music reactive games, MIDI parameters and control data calculated based on them can be divided into three principal groups: “Event”, “state”, and “transition”. In the following, we describe these groups in more detail (but not exhaustively) and make suggestions on how useful they would be to control a game.

5.1 Event Parameters

“Event” parameters are musical features that occur occasionally and last for a brief time. They are appropriate for e.g. triggering new objects or causing some abrupt actions in the game. Most useful musical features belong to this parameter group.

According to our experiences, the most useful MIDI message type belonging to this category is Note On. All Note On events in the song or only certain pitches could be mapped to e.g. spawn new game objects to the screen. Another interesting possibility would be to map higher notes to spawn objects to the top of the screen, and lower notes to the bottom of the screen. Note On velocities, which typically have values above 50, can also be used to modify the game.

An average listener seems to notice quite well when a certain percussive sound (such as bass drum or crash cymbal) has an effect on the game. In the case of polyphonic music, mappings between harmonic notes (especially those on accompanying tracks) and game events seem to be more difficult to notice.

In general, MIDI channels that have a low number are more widely used than channels having a high number. Channel 10 is an exception, as almost all MIDI files include at least one drum or percussion track. If only a subset of MIDI channels is mapped to the game parameters, drums are a good alternative as an average listener can easily separate them from the mix.

Another interesting musical feature, although not directly a MIDI event, is polyphony. As high-polyphony music often sounds more massive and intensive than low-polyphony, exceeding a certain polyphony threshold could be mapped to e.g. game’s difficulty level, amount of enemies, and so on.

By mapping Program Change messages to game parameters, it is possible to modify the game according to instruments used in the song. Traditional instruments such as piano are really popular, while e.g. different sound effects sounds are quite rarely used. Almost all songs include at least one drum or percussion track, and the basic drum set (bass drum, snare drum, hihat, etc.) is usually used.

Pitch Bend MIDI event is typically used in the case of stringed instruments and lead sounds, and its effect on the music is quite clear. The event can be used e.g. to control the vertical position of objects on the screen.

System Exclusive and NRPN messages are quite laborious to create. If they occur in a song, the composer has most probably used them to modify some parameter of some specific synthesizer. If another synthesizer is used to play the song, it is very likely that the listener will not hear the effect at all. These messages are really rare, and therefore can be used to create random –like surprise elements to the game.

There are 127 different MIDI Control Change messages, most of which are rarely used in songs. Popular messages include Channel Volume (#7), Pan (#10), Modulation Wheel (#1), and Damper pedal i.e. sustain (#64). Their effect on the music is quite evident, while some other CC messages may be such that an average listener does not notice if they have been used or not. Rare CC messages should be mapped to create random –like behaviour to the game or neglected.

5.2 State Parameters

“State” parameters are musical features that stay more or less the same for a longer time. This kind of features are suitable for controlling longer-term game properties such as speed of game and average number of enemies.

According to authors’ experiences, the most important and easily noticeable mapping seems to be connecting song’s tempo to control the overall speed of the game. Other interesting alternatives include e.g. controlling a single moving game object and modifying the difficulty level according to tempo. All MIDI files include a tempo meta-event, and it usually remains constant throughout the song. The tempo of most pieces of music lies between 60 and 140 BPM (beats per minute). Values outside this range can be used to create e.g. some kind of surprises and extreme speeds.

Key and time signature meta-events are much less common than tempo. If the time signature does not exist, a default signature of 4/4 is used. Most songs are in 4/4, so other divisions can be used to create surprises to the game. Key signature information (e.g. C major or F minor) could have an effect on e.g. the mood of the game, time of day, and so on. As lyric and text meta-events are mainly used in karaoke MIDI files (*.kar), a background music reactive game should not rely on them.

Other examples of state parameters include e.g. average polyphony inside a specified time window and “note density” i.e. the number of notes inside a specified time window.

5.3 Transition Parameters

By “transition” parameters, we are referring to significant changes from one musical quality to another. Examples include large intervals in a melody line, the change from silence between two pieces of music to an aggressive beginning of the next piece, large change in note density, and change in time signature. “Transition” parameters can be used for similar purposes to control game content as “event” parameters.

6. AUDIO ASTEROIDS TEST PLATFORM

The authors specified a demonstrative game that could be used to evaluate the “background music reactive games” concept in practice. The idea was to study useful connections between MIDI parameters and game content. An open source game “Maelström” [13], which is based on the well-known arcade game Asteroids, was modified accordingly and renamed “AudioAsteroids”. The game was selected because it is simple and intuitive, and has several types of objects flying around that can be controlled by music. The most important modification was the integration of a custom software synthesizer as game’s MIDI player engine. The synthesizer was able to analyze the MIDI file during the playback and thus control the actual game engine. The MIDI engine also supports simultaneous real-time input from a MIDI controller.

In the game, the player controls with the alphanumeric keyboard a spaceship that must avoid colliding with asteroids and other objects flying around in space. The player must attempt to shoot dangerous objects such as asteroids, enemy ships, and black holes with the ship’s laser weapon. There are also some bonus objects the player must collect in order to get more points, more lives, etc.

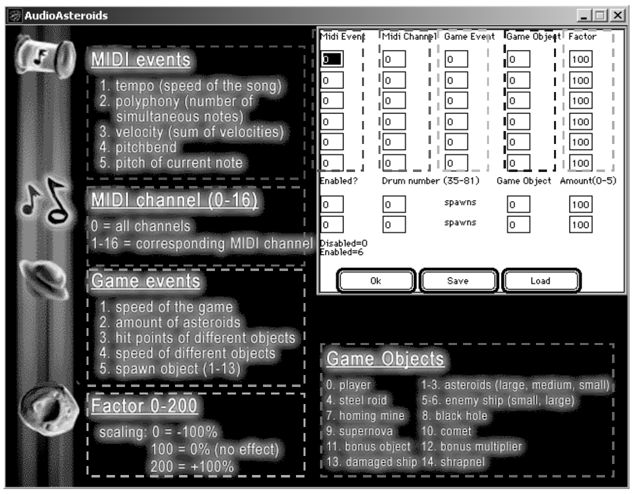
6.1 Defining Connections

The players were able to define the connections between MIDI control parameters and game events by themselves. Two types of musical control parameters, namely “event” and “state”, were supported. The properties (e.g. speed and amount) of game objects could be controlled by musical events like the pitch of a note and the number of simultaneous notes. Overall speed of the game could be made dependent on the musical tempo, and so on.

Figure 2 shows a window where a mapping for the game is specified. A single connection could be described with the following formula: ”A MIDI event from a certain MIDI channel is connected to a game event that affects a game object with a certain factor”. Connections can be scaled (“fine-tuned”) with a factor between –100% and 100%, which are represented in the game as integers 0 and 200, respectively. High positive percentage values mean that the MIDI event has more effect on the game event. As an example, when the tempo of a MIDI file increases the speed of the game could also increase. Negative percentage values have the opposite effect (e.g. when the tempo increases the speed decreases). Value 0% has no impact, so if all the factors are set to it the game works just like the original Maelström.

Every musical parameter can control multiple game events, and every game event can be controlled by multiple musical parameters. In the latter case, the final game event value is a sum of affecting musical parameters.

AudioAsteroids has also two special connections that are used in a different way from the basic connections. After turning either special connection on, the user has to select a drum sound from the standard General MIDI 1 [13] drum kit, some game object, and specify how many of these objects can exist simultaneously. When the selected drum is played in the song, it will spawn the chosen game object to the screen.



MIDI events
1. tempo (speed of the song)
2. polyphony (number of simultaneous notes)
3. velocity (sum of velocities)
4. pitchbend
5. pitch of current note

MIDI channel (0-16)
0 = all channels
1-16 = corresponding MIDI channel

Game events
1. speed of the game
2. amount of asteroids
3. hit points of different objects
4. speed of different objects
5. spawn object (1-13)

Factor 0-200
scaling: 0 = -100%
100 = 0% (no effect)
200 = +100%

Midi Event	Midi Channel	Game Event	Game Object	Factor
0	0	0	0	100
0	0	0	0	100
0	0	0	0	100
0	0	0	0	100
0	0	0	0	100
0	0	0	0	100

Enabled?	Drum number (35-81)		Game Object	Amount(0-5)
0	0	spawns	0	100
0	0	spawns	0	100

Disabled=0
Enabled=6

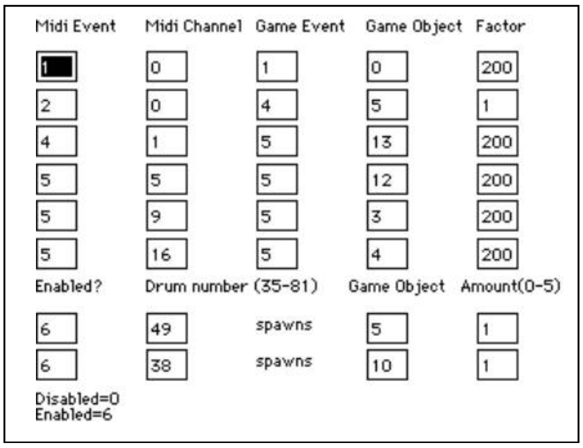
Game Objects
0. player 1-3. asteroids (large, medium, small)
4. steel roid 5-6. enemy ship (small, large)
7. homing mine 8. black hole
9. supernova 10. comet
11. bonus object 12. bonus multiplier
13. damaged ship 14. shrapnel

Figure 2. Defining connections in AudioAsteroids.

6.2 Example of Generic Mapping

Although the player can make his or her own mapping before starting the game, this can be an iterative and very time-consuming process. Therefore, a representative set of connections (which would be as illustrative as possible for several types of background music) was defined. The set was stored as a generic connection file that is delivered with the game package. The selected mappings are shown in Figure 3, where:

- Tempo of music controls game’s overall speed;
- Combined polyphony of all MIDI channels affects the de-acceleration of UFOs;
- Pitch bend amount on channel 1 controls the number of damaged spaceships on the screen;
- Pitch of current note on channel 5 affects the amount of bonus multipliers;
- Pitch of current note on channel 9 controls the number of small asteroids;
- Pitch of current note on channel 16 spawns steel asteroids to the screen;
- Hit of crash cymbal spawns the smallest amount of UFOs; and
- Snare drum spawns the smallest amount of comets.



Midi Event	Midi Channel	Game Event	Game Object	Factor
1	0	1	0	200
2	0	4	5	1
4	1	5	13	200
5	5	5	12	200
5	9	5	3	200
5	16	5	4	200

Enabled?	Drum number (35-81)		Game Object	Amount(0-5)
6	49	spawns	5	1
6	38	spawns	10	1

Disabled=0
Enabled=6

Figure 3. Example of a mapping used in AudioAsteroids.

6.3 Findings from AudioAsteroids

Many MIDI files were tested with AudioAsteroids. In addition to showing the value of a generic mapping, they also proved that the concept works nicely in practice. The selected generic mapping worked well on any MIDI song generating numerous variations to the game.

The authors learned that the most important and easily noticeable mapping was connecting song's tempo to control game speed. Another very perceivable connection was mapping a certain drum being played to generate a specific game object. In the case of other mappings, it was not always evident why some things happened on the screen. However, the fact that different songs made the game appear and behave differently was enough to provide a satisfying gaming experience for people who tried the game. In a sense, any MIDI file in the player's collection could render a different game level.

In general, musical people seemed to enjoy the game more than those who had never played an instrument or listened to a lot of music. They also understood the used mappings better.

7. REAL-TIME MIDI ANALYSIS SOFTWARE FOR SYMBIAN

Mobile phones are becoming increasingly popular devices for playing games and listening to music. So far the main use of their MIDI synthesizers has been the playback of ringing tones. Therefore, mobile phones' MIDI players do not usually have a dedicated callback API (Application Programming Interface). A MIDI file can be played and stopped, but an application does not have any way to get detailed information about the contents of the file.

However, in the case of background music reactive games this information is required. It is also desirable that the game receives the information in real time and in synchrony with the music playback. Because of this, a MIDI analysis software module called ”Virtual Sequencer” or "VS" was developed. At the time of programming the software, target platforms were Nokia mobile terminals with Symbian 6.1 (including N-Gage game deck, [12]) and 7.0 operating systems. VS was implemented with C++ programming language.

In the case of a traditional MIDI player engine, a file parser software component reads the MIDI file to be played and sends the parsed data to a sequencer component. Sequencer is responsible for sending scheduled MIDI events to a synthesizer component at appropriate moments. Synthesizer generates the actual audio waveform and sends it forward to be played through loudspeakers or headphones. The implemented Virtual Sequencer software consists of only file parser and sequencer components, both of which are considerably simpler to implement and run than the synthesizer.

Figure 4 illustrates using VS in a game for a mobile phone. As VS has been separated from the actual game engine, it is possible to utilize the same code module again in other games. In the figure, the game engine commands VS and MIDI player blocks to load the same MIDI file. (Here term MIDI player refers to the software or hardware synthesizer of the used mobile phone model.) Both blocks then parse the MIDI data. After receiving a play command, MIDI player's sequencer component starts sending scheduled events to its synthesizer part, while VS's sequencer starts sending control events to the game engine. The game engine then maps these events to selected game parameters, and the game reacts to the music in real time.

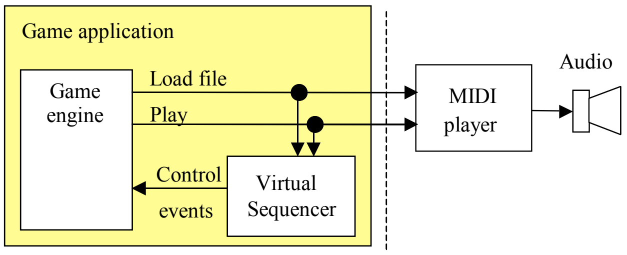


Figure 4. Illustration of using Virtual Sequencer in a game.

Virtual Sequencer is a light-weighted component that does not consume an excessive amount of processing power even when using complex MIDI files. It is a dynamically linked library (DLL) that is used through its API. VS does not generate any sound or interfere with Symbian's native MIDI player. The operation of VS is two-way: It can be instructed as a normal MIDI player (load, play, stop, jump back to song start) and it can make callbacks to inform its host application (e.g. a game) about MIDI events, polyphony levels, etc. that the host wants to be informed about. Some of VS's control events are derived directly from individual MIDI messages, while others are based on some simple calculations. A good example of the latter is "polyphony *N* was exceeded" control event.

The host can be informed via callbacks when any of the following situations occurs during playback:

- Tempo Change message is used in the song;
- A certain number of notes is played simultaneously;
- Note On or Note Off message is used on certain MIDI channels;
- Program Change, Pitch Bend, or Control Change (CC) messages are used on certain MIDI channels;
- SysEx message or meta-event is used in the song;
- Certain notes are played on certain MIDI channels;
- NRPN, Polyphonic or Channel Key Pressure events happen on certain MIDI channels;
- Certain instrument (i.e. defined instrument number) is played (by Note On) on certain MIDI channels; and
- Any MIDI message is used on certain MIDI channels.

Each callback includes all relevant MIDI data as parameters. For example, when Note On is called it is important to know also the note number and its velocity.

8. CONCLUSIONS

This paper elaborated on the concept of "background music reactive games", where game's background music is used to control game parameters and thus actions on the screen. Each song generates a new game level with varying characteristics and difficulty, and players can try to solve playlists of their favorite music or even mobile phone ringing tones.

A test platform called AudioAsteroids was implemented for experimenting with different mappings. A generic mapping was defined in order to test the game behavior with several types of music. It was learned that the most important and easily noticeable connection was controlling game's speed with the musical tempo. AudioAsteroids convinced the authors that the idea of using music to control game content works in practice, and that MIDI is a suitable format for it. By defining an appropriate mapping between musical control parameters and game parameters, it is possible to develop games that behave differently with each piece of background music. The concept introduces a new dimension into the experiences of gameplay when players realize how game content and behavior changes as a result of certain characteristics of the background music.

The benefits of using MIDI were also discussed. Three types of control information ("event", "state", and "transition") can be effectively calculated from MIDI data, and one-to-one mappings between specific musical events and game events are possible. Similar exact information is currently very difficult to extract from digital audio, as practical digital audio analysis algorithms predominantly include detecting changes in sound volume levels, beat tracking, and monophonic pitch detection. To name one possible future implementation, MIDI-control is a way to realize rhythm games that can be played with any MIDI file as background music.

In order to enable the creation of Symbian games on mobile phones, a software component called "Virtual Sequencer" was implemented.

9. FUTURE WORK

An exciting future development would be to combine the activities of playing a game with those of making music with musical controllers. Aspects of group performance could be introduced to this activity. For example, one of the performers could control the hero in the game, while the musical performance of others would create obstacles for the hero. In [15], Blaine and Fels have discussed the principles of collaborative musical interfaces. These guidelines could be adapted into the design of background music reactive multiplayer games.

One constraint in AudioAsteroids was that it is controlled with the alphanumeric keyboard, which most people do not regard as a real-time musical controller (despite it can be used to trigger samples). AudioAsteroids (and many other video games) can be played with as few as three or four distinct buttons, so there are several musical controllers that could be used for this purpose. The use of these controllers as well as new mapping strategies should be examined carefully. In addition, more detailed user testing than done so far would be required.

Theoretically, there are three ways to combine real-time MIDI input with video games like AudioAsteroids. The game could be based on an exclusive real-time performance or the playback of a prefabricated MIDI file with live parts performed on top of it. In addition, the actions of the player (e.g. maneuvering a space ship and shooting) could trigger musical sounds instead of sound effects. All these interesting possibilities can only be explored by implementing new game prototypes.

10. ACKNOWLEDGEMENTS

The authors would like to thank their colleagues Mikko Heikkinen and Antti Eronen as well as Nokia Research Center student trainees Lauri Oikari and Olli Etuaho for their contribution to the project.

11. REFERENCES

[1] Open Sound Control. http://www.opensoundcontrol.org/, 25.1.2006.

[2] Staraoke. http://www.staraoke.fi (only in Finnish), 25.1.2006.

[3] Gitaroo Man. http://ps2.ign.com/objects/015/015184.html, 25.1.2006.

[4] Hämäläinen, P., Mäki-Patola, P., Pulkki, V., and Airas, M. Musical Computer Games Played by Singing. In *Proceedings of the 7th Int. Conference on Digital Audio Effects (DAFx'04)*, Naples, Italy, October 5-8, 2004.

[5] Musical Invaders. http://www.dmi.usherb.ca/minvaders/en/, 25.1.2006.

[6] Musical Space Invaders and Music Scale Teacher. http://www.sharewaresoft.com/Music-Scale-Teacher-download-5736.htm, 25.1.2006 .

[7] Mad Maestro. http://ps2.ign.com/objects/017/017479.html, 25.1.2006.

[8] Vib-Ribbon. http://www.vib-ribbon.com/, 25.1.2006.

[9] Parappa The Rapper. http://www.parappa-the-rapper.com/, 25.1.2006.

[10] Blix. http://www.shockwave.com/sw/content/blix/blix.html, 25.1.2006.

[11] Rez. http://www.sonicteam.com/rez/, 25.1.2006.

[12] Nokia N-Gage. http://www.n-gage.com, 25.1.2006.

[13] Maelström, S. Lantinga/Ambrosia Software, http://www.devolution.com/~slouken/Maelstrom/, 25.1.2006.

[14] Blaine, T. The Convergence of Alternate Controllers and Musical Interfaces in Interactive Entertainment. In *Proceedings of New Interfaces for Musical Expression (NIME) Conference*, Vancouver, Canada, May 26-28, 2005.

[15] Blaine, T., and Fels, S. Design Issues for Collaborative Musical Interfaces and Experiences. In *Proceedings of New Interfaces for Musical Expression (NIME) Conference*, Montreal, Canada, May 22-24, 2003.

[16] Blaine, T., and Forlines, C. JAM-O-WORLD: Evolution of the Jam-O-Drum into the Jam-O-Whirl Gaming Interface. In *Proceedings of New Interfaces for Musical Expression (NIME) Conference*, Dublin, Ireland, May 24-26, 2002.

[17] Holm, J., Havukainen, K., and Arrasvuori, J. Novel Ways to Use Audio in Games. In *Proceedings of Game Developers Conference (GDC)*, San Francisco, USA, March 7-11, 2005.

[18] Holm, J., Havukainen, K., and Arrasvuori, J. Personalizing Game Content Using Audio-Visual Media. In *Proceedings of Advances in Computer Entertainment (ACE) Conference*, Valencia, Spain, June 15-17, 2005.

[19] Cassidy, G., MacDonald, R., and Sykes, J. *The Effects of Aggressive and Relaxing Popular Music on Driving Game Performance and Evaluation.* Abstract in http://www.gamesconference.org/digra2005/viewabstract.php?id=94, 2.4.2006.