The Game Pulse - Timing Game Events and Music Events

Richard van Tol MA, Sander Huiberts PhD

HKU University of the Arts Utrecht
Lectorate Play Design and Lectorate Music Design
PO Box 2471, 1200CL Hilversum, NL

Abstract

The aim of this paper is to examine how to design nonlinear music systems that allow for music events to coincide and correspond with dynamically changing game events. We address the challenges of connecting nonlinear music systems to gameplay and distinguish three approaches for timing time music events and game events: Trail, Sync and Lead. We explore these three approaches in the design case study *Gluddle versus The Supervision*, a game created by the authors. Our preliminary findings illustrate the benefits of combining Trail, Sync and Lead, positively influencing the game experience, but also the need for extra attention to gameplay balance and technical implementation.

1. Introduction

Nonlinear music systems for games have evolved incredibly over the past ten years. With sophisticated systems music designers try to create music that adapts to gameplay, to smoothly blend scenes, states and storylines. In contrast with traditional media such as film, a game music designer generally does not know upfront exactly when to time a particular music event. Therefore it can be rather challenging to design a nonlinear music system that allows for music events, such as musical transitions and progressions, to coincide and correspond with dynamically changing game events [1].

In this paper we will examine which methods and approaches exist to time nonlinear music and describe their benefits and disadvantages. We will first look into the timing of music events and their function in the game. We will then discuss three approaches of timing music events in a nonlinear music system. Through a design case study (see below) we explore the possibilities of these three approaches for designers.

This paper is a practice-based exploration from a designer's point-of-view. We will use the game *Gluddle versus The Supervision* (2012) [2] as a design case. This toy-like bouncy ball game has been fully developed by ourselves. Gluddle was chosen as a design case study because as its designers we had full access to the game's code and assets and therefore complete control over the game design and music system, enabling easier exploration of the various design parameters.

In the game, players control colourful bouncy ball creatures named Gluddle. Their opponent is The Supervision, a mysterious spying entity whose spherical robot eyes (nicknamed 'Bullseyes') try to rob the Gluddle of their privacy by freezing their movement and taking flashing snapshots. The goal of the game is to clear all Bullseye-infested levels by launching and bumping Gluddle against the evil Bullseyes afloat in the sky. A key mechanic is freezing Gluddle in mid-air, thus creating paths and boundaries for succeeding Gluddle to bounce upon. The game is developed in Unity and currently published on the iOS and Android platforms.

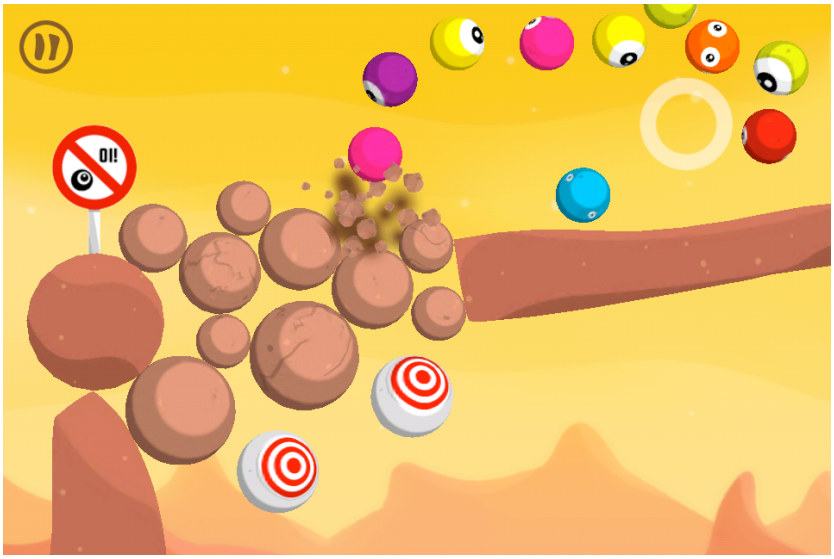

Figure 1: a screenshot of *Gluddle versus The Supervision*. The coloured spherical creatures are The Gluddle, the white globes with the red Bullseyes are The Supervision (henceforth 'Bullseyes').

2. Event Timing in Games

2.1. Free-time timing versus quantised timing

Events in games are commonly triggered by either the player or by timed functions [3]. Both are usually timed in 'free-time', which means these functions are executed the moment they are needed.

An alternative to triggering game events in free-time is 'quantised timing'. When quantised timing is used to trigger game events, these events are synchronised to a grid of specific time intervals. Such an alternative allows for the triggering of events in coincidence with the beat or pulse of the music.

Game repertoire offers many cases in which game events are synchronised to the musical pulse. While most of these examples include rhythm and music games, in which timing is an intrinsic part of the gameplay [4], there exists a smaller repertoire of games without such music- or rhythm-focused gameplay. Game event quantization is found in for instance Beat Sneak Bandit (2012), Monkey Island 2 (1991) and Mushroom Men (2008).

2.2. Phases of Music Event Timing and their function

The timing of music events in relationship to game events gives music designers the ability to create meaning and significance [5]. There are multiple approaches to describe temporal relationships between events, such as James F. Allen's interval-based temporal logic [6] which distinguishes 13 base relations between two intervals (events with a distinguished start, duration and end). For Gluddle's real time music scheduling system we focus only on the relationships between the initiations of game events and music events, and not on the duration or end of these events. Based on this simplified perspective, we would like to propose three key phases in music event timing: *Prelude*, *Perilude* and *Postlude*.

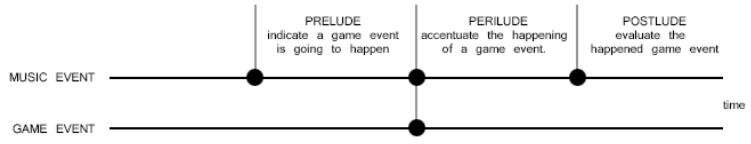

Figure 2: Three moments of music event timing in games: Prelude, Perilude and Postlude.

Prelude (literally meaning 'before play') describes the moment in which music is used to indicate a game event is going to happen, foreshadowing (the possibility of) the game event. Anticipating upcoming game events may improve the player experience and enhance flow in games [5]. 'Foreshadowing' is a technique that has a rich tradition in film music [7] but it is one of the more difficult functions to achieve in games due to their nonlinear nature [5,8]. Perilude (literally meaning 'during play') describes the moment in which music is used to accentuate the happening of the game event through synchronization. Postlude (literally meaning 'after play') describes the moment in which music is used to evaluate the happened event, providing reflection and afterthought. Postlude can be seen as a short moment of evaluation, emphasizing an event [5].

2.3. Event time scheduling challenges in nonlinear music systems

Depending on the techniques that are used in nonlinear music systems, the nonlinear music design challenges differ. In the case of 'horizontal re-sequencing' [9] - modifying the musical structure in real time - the duration of music cells defines the response time [10], see figure 3. This makes it more difficult to react in time compared to, for instance, 'vertical reorchestration' - adding musical layers to an already playing music cell [11]. In case of the latter technique, it is more difficult to react to changing events with alternating chord progressions.

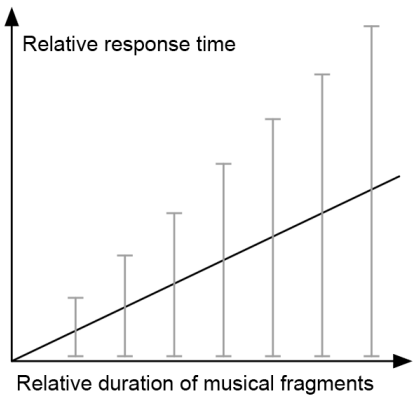

Figure 3: The length of musical fragments compared to the response time [10].

As Nispen et al [10] mention, the average response time of a music system depends on the length of music fragments, so increasing the duration of musical fragments increases the average response time of the system. Common methods to overcome response time latency (the result of waiting for a musical fragment to end) are abruptly starting a new fragment or crossfading between the current and the new fragment. While these methods have been used in games for over 20 years and have proven to work in several contexts, these are not apt for every music genre. Also, the player may ”become aware of the system and what the outcome of the interaction will be, causing annoyance, often making the player mute all sounds or end the game” [10].

3. Three approaches to time nonlinear music systems

3.1. Connecting game events and music events

By designing Gluddle, we examined how to make a music system respond appropriately to dynamic game events. When a particular event occurs, the aim is to make the music system react accordingly so that what is heard corresponds to what the player should experience. There are multiple design approaches to establish a temporal relationship between a game event and the corresponding music event. Below, we will discuss their properties and their possibilities.

3.2. Trail

The first approach, nicknamed *Trail*, is when the timing of a specified game event conducts the response of the music system. The music follows the game event in an *'As Soon As Possible'* -manner, always taking some time to respond depending on the remaining duration of the currently playing musical fragment. In this approach, the music changes when the appropriate musical timing is there, e.g. when the musical cell has stopped playing, when an exit or fade point in the cell is reached, or when the transition is accompanied by a stinger [8] that masks the transition.

This technique is flexible and supports many music styles. A downside is that it is always 'too late' in the sense that it is always subsequent to the action in the game - the music can never foreshadow or predict any action.

3.3. Sync

The second approach, named Sync, is the immediate triggering of music events synced to the timing of game events. When an event occurs, the change in music is triggered and executed immediately. This kind of relation between events and music is also found in the film scoring technique 'Mickey Mousing' where music is timed to or sync with the visuals on screen [12, 13]. Nispen et al [10] argue that although this immediate triggering of music in relation to action is objectionable from a musical perspective and could possibly diminish immersion, the technique seems widely accepted by players and is recognised as part of the game music tradition.

A recent example of a non-rhythm game using Sync is Super Mario Galaxy (2007), in which the tempo and pitch of the game's music is synchronised to the velocity of a ball (with lead character Mario on top) controlled by the player. In BeatBuddy (2013), the seemingly non-diegetic soundtrack is actually made up of (diegetic) sounding objects in the world, each cleverly adding a musical track in sync to create the game's soundtrack. But some games go one step further. Platform game Beat Sneak Bandit radically forces players to quantise their input to the games' soundtrack. Not syncing their touches simply results in a failure to interact with the game or even worse, in a penalty. However, it is this twist that makes this game fun and challenging.

Sync is a relatively easy way to time nonlinear music systems, but is not easily applied to every part of the game. In the case of the first two examples, many sounds related to the *Event* and *Interface* domains [5] are still in free-time and are not synced to the beat of music. In Beat Sneak Bandit these are synced to the beat of the music, restricting the player's freedom to interact with the game.

3.4. Lead

The third approach consists of quantizing game events to the musical pulse. In this approach, nicknamed Lead, the musical pulse conducts the timing of game events. It is not a new technique. iMUSE (Interactive MUsic Streaming Engine), a well-known interactive music system developed in the early 1990s by LucasArts' composers Michael Land and Peter McConnell, already allowed for synchronizing music with the visual action in the game, and for transitions from one musical theme to another, enabling nonlinear music in games. A somewhat lesser known fact is its ability to quantise the timing of game events to the timing of the music. For example, in Monkey Island 2 - Le Chuck's Revenge, the system is used to slightly pause

visual animations so that they are in sync with the music [15]. Another example is Mushroom Men (2008), which uses a combination of Sync and Lead.

Music Timing Approach	Game Event	<>	Music
Trail	Game event timing leads, is quantised to its own (free) timing		Music timing trails behind game event timing, but is quantised to its own pulse Music event trigger adds extra time (waits) until it is timed with musical pulse
Sync		Game event timing and music timing is synchronised	
Lead	Game events timing are synced to the musical pulse Game event trigger adds extra time (waits) until it is timed with musical pulse		Musical pulse conducts the timing of game events

Table 1: Overview of the three music timing approaches.

4. 4. Case study: Gluddle versus The Supervision

4.1. Gluddle's Nonlinear Music System

The aim of Gluddle's music system was to correspond with the hectic moments in the pinball-like gameplay. The music system can be regarded as an adaptive jukebox that semi-randomly triggers musical blocks of music in time, which can be overruled by important actions of the player and game events triggered by the game system. The levels of Gluddle are designed to support short, rapid moments of playfulness, hence the action-focused music system.

Gluddle features a 'totalitarian, synthetic polka-beat' soundtrack and combines the two techniques that were mentioned in section 2.2 for nonlinear music systems. Horizontal resequencing is used for alternating cells and triggering special event-cells in time. Vertical reorchestration is used for the triggering of atmospheric 'long layers' and the 'double time beat and hi-hats' that are triggered when the velocity of the active game character crosses a certain threshold. To increase the feeling of harmony between the game activity and music, the sound effects of the majority of events are in the key of the music (A flat).

The following overview is an illustration of the musical structure of Gluddle, simulated in a Digital Audio Workstation (Tracktion 4). Below the illustration, the functionality is described.

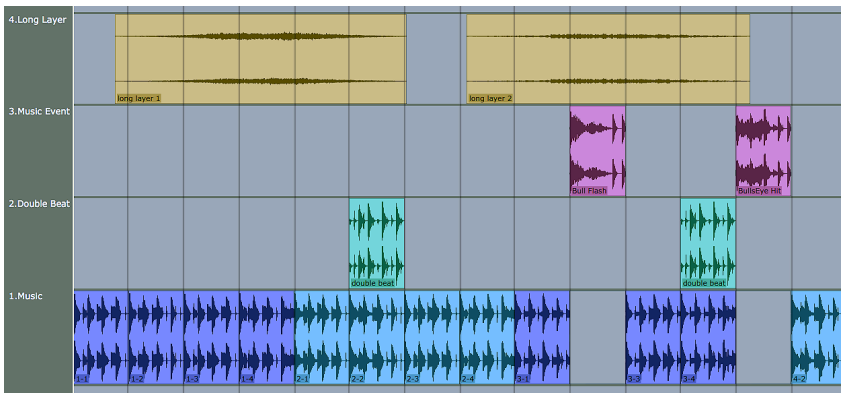

Figure 4: Visual mock-up of the music tracks in Gluddle's music system. Listen to this mock-up example at reference [16]. To see this system in action with gameplay, watch the video at reference [17].

1. Music
Musical cells of one measure are triggered in blocks of four cells (1-1, 1-2, 1-3, 1-4). The composition is either structured in blocks (in the first levels) or randomly plays blocks that are compatible with one another.

2. Double Beat
The double beat layer is triggered whenever the active Gluddle reaches a velocity above a certain threshold, by for instance bumping onto a special 'Bumper' object which gives the Gluddle a speed impulse.

3. Music Event
Whenever a special event occurs, the music system always plays the corresponding event music. To exemplify, hitting a Bullseye will play the Bullseye music event, the freeze music event will sound when being frozen, hitting a gravity-changing ball will play the gravity music event and unleashing ‘GigaGluddle’ (a notable event) will play the GigaGluddle music event.

4. Long Layer
The long layers have a duration of 7 to 12 seconds and are triggered in random intervals to add a more continuous feel to the short music cells of 1 measure. Gluddle does not feature sound objects related to the *Zone* domain [14]. Instead, these long layers are added to give Gluddle a more atmospheric character.

The game engine features a game metronome (nicknamed 'gametronome') that has been programmed to manage the timing and triggering of music fragments. The music is constructed using several arrays of 4 cells of 4 bars. One cell has a duration of 2.285 seconds (105 beats-per-minute) so the metronome pulse of one beat is 0.57125 seconds (see Table 2). This game metronome pulse is used for triggering several game events that will be described in 4.2.

Metronome - Circular Timer 1

2.285s
1 bar, 4 beats, 105 BPM

Metronome - Circular Timer 2

0.571s	1.143s	1.714s	2.285s
1 beat	1 beat	1 beat	1 beat

Table 2: Music pulse of Gluddle that is used for timing and quantizing the game events (in seconds). Some game events are triggered on a beat, while others are triggered per measure.

4.2. Game metronome usage

The game metronome is linked to several game events. A Bullseye will start charging whenever a Gluddle enters its vicinity. Depending on the design and difficulty of the level, the charging time and range differs per Bullseye. The moment for a Bullseye to freeze a Gluddle by a camera flash is the first beat of a music block. When the flash is due, a final check is made to see if the Gluddle is still in range of the Bullseye. If so, the Bullseye flashes and the Gluddle is frozen. If not, the Bullseye does nothing and discharges.

Prelude	Perilude	Postlude
Before Event Time	At Event Time	After Event Time
Gluddle is in range Bullseye starts to shake	If Gluddle is still in range: Bullseye Flash	Gluddle has been frozen
	Bullseye says 'Control', 'Freeze', etc.	Evaluation: Bullseye says something cynical ('hahaha')
Prelude: Music Tension sound fades in	Sync: Freeze Event Music	

Table 3: Prelude, Perilude and Postlude of a Bullseye in action.

Bullseyes animate and rotate to the beat of the music. Their verbal exclamations, for instance when the player touches them or when they freeze a Gluddle, are also synchronised to the beat. Gluddle that are frozen by a Bullseye become grey and start a pulsating animation on the beat. Other objects, such as the 'Tone Balls' make a sound on the beat when touched by a Gluddle.

The music event behaviour differs per event. When a Gluddle accidentally springs off the screen, the music system will try to respond with a downward melody. As a music cell is 2.285 seconds, it could take more than 2 seconds to play the music event belonging to this action. During playtests we determined not to bother the player with a past-time event that is not important anymore nor have the user

wait for 2 seconds for a new Gluddle to appear. In such cases, the music event will only play if the music cell start delay is less than 1.5 seconds. There is one notable exception: the music event for hitting a Bullseye is always played as it is an important positive reward for an event that should not be omitted and go unnoticed. Prioritisation of events proved a necessary ability of our real time scheduling system in order to make its behaviour consistent with gameplay.

5. Results

In Gluddle, we combined the three approaches for timing nonlinear music systems: some game events are timed immediately, some are delayed slightly and synced to the musical pulse. We found that it was not desirable to quantise player input to the music pulse, because the gameplay required free-time interaction. However, by subtly quantizing some important game events, we achieved a sense of harmonisation that gently stimulates players to interact in the beat (although this is not required), without Gluddle becoming a rhythm game. In a similar way that is described by Burns [18] about the rhythm game Planck (2009): "*If we quantize too much, such as delaying bass drum sounds to every whole note, the delay between the player's action and the sound of the note becomes noticeably long, and the "gamey" portion of the experience gets watered down. The first part of the Planck idea, then, was to try and combine instant visual feedback of game events (such as destroying an enemy) with time-delayed reactions that make musical sense. Planck's attempt to solve this problem centres around giving instant visual feedback of an enemy being destroyed, but by keeping a 'destroyed' version of the enemy in play until the next available note."*

Throughout the development and release of Gluddle we tested the game with a number of playtesters using multiple play sessions combined with observation and informal interviews. We noticed that many playtesters made positive remarks about the game's experience and "flow-y" feel. When asked why, none of the playtesters could accurately describe why the game felt like this, other than that it had "something to do with the music". The relationship between the timing of sound, animations, music events and game events seemed to prove too subtle to attract attention, but not too subtle to experience. We have yet to make a comparative study to see to what extent the game experience changes when the gametronome is disabled.

We made an important discovery that had a major impact on the game mechanics and the balancing of (the difficulty of) the game. By linking the reaction time of enemies to the musical pulse, the reactivity of the Bullseye decreases as it literally waits for the musical pulse. Since the Bullseye acts later than initially was intended, the game

became easier to play. The game therefore required more tweaking in order to balance it. This is an important point to consider when designing a game metronome that conducts the actions of enemies: the music system becomes a fundamental piece of the game system and is vital in balancing the game. Since we were the game's audio designers as well as game designers, it was easy to combine these tasks. If these two roles are fulfilled by different persons, it is very important for the two to communicate and collaborate closely [19].

And lastly, we found that modern game technology can still be a limiting factor, as sample accurate triggering asks for very precise co-routines. Music files need to be triggered with very little latency, otherwise noticeable gaps can be heard. As the framerate on mobile devices usually is 60 frames per second or lower, the maximum latency is about 16 milliseconds, and possibly more when the framerate drops due to performance issues. Although Unity has added sample-accurate triggering of music in Unity 4.0 with the introduction of *AudioSource.PlayScheduled()*, a scripted change of music (in other words: 'decision time') is often still based on the framerate. For correct starting of music loops, it is usually necessary to give the game engine some time to process the audio file in order to start it without causing a CPU-spike, which could add extra time before the intended response time.

make decision >>>	**preload sample >>>**	**actual event >>>**
framerate dependent latency - 16 msec or less	ideally 20 msec	is this event still relevant?

Table 4: the dilemma of synchronising music in a game engine which requires preloading, which can conflict with synchronising the music to real time events.

6. Conclusion and Discussion

In this paper, we discussed the possibilities of unifying the timing of game events and music and distinguished three approaches to time game music and game events: Trail, Sync and Lead. In case of the latter, the game system may pick a moment for the instantiation of game events based on a time that coincides with the musical timing.

So far, Lead seems to be only scarcely used in games compared to Trail and Sync. Arlauskas describes the use of game-metronomes [20]: *"They have a lot more to offer game developers than simply playing sounds to the beat of the music. What I really want to get across is how the metronome event system can help unify the timing of many elements in games, making them feel complimentary."* According to Arlauskas, a complimentary benefit is that the unification of game music and game events can lead to surprise through unexpected synchronicity: *"This sort of thing isn't immediately apparent as being 'on the beat', but lots of nice, seemingly coincidental (...) timings can happen."* The cases that use metronomes to synchronise game events incorporate slow to medium pacing. Future research could investigate how such a system performs in games with fast pacing.

We have experimented with measuring the timing of the player's actions in relation to the music and see if the player's timing is influenced by the music. Due to the latency caused by the framerate in Unity, we found that making exact measurements is difficult at this stage. Future research might include measuring the timing of the player with higher precision.

In our study of linking the pulse of game events to the pulse of the music, we found that it is possible to discern even more pulses during gameplay, such as the pulse of player interactions with the game. Distinguishing different pulses in gameplay may be a useful focal point for future research as harmonizing co-existing pulses may lead to more complementary designs.

It has been suggested that Italian scientist and philosopher Galileo Galilei used music in order to measure time in his experiments [21]. Concluding our design study we can say that using musical timing as a temporal grid for game events is a relevant and inspiring topic for game creators. We propose designers not only focus on the spatial composition of their artefacts, but also take notice of the temporal relationships between music events and game events in order to enhance and harmonize the gameplay experience.

7. References

1. Goles, N. (2010), Game Event Handling - Part 1. Retrieved 22nd July, 2013, from: https://nicolasgoles.com/blog/2010/12/game-event-handling-part-1/

2. Creative Heroes, Gluddle (2012). Retrieved 20th July, 2013 from: http://gluddle.com

3. Also known as coroutines or invokes. For an overview of common use of coroutines in Unity, please confer to: http://docs.unity3d.com/Documentation/ScriptReference/Coroutine.html

4. Kayali, F. & Pichlmair, M. (2008). Playing Music, Playing Games - Simulation Vs. Gameplay in Music-Based Games. In: Vienna Games Conference 2008 'Future and Reality of Gaming' (FROG).

5. Huiberts, S. (2010), Captivating Sound: the Role of Audio for Immersion in Games. Doctoral Thesis. University of Portsmouth and Utrecht School of the Arts, Portsmouth.

6. Allen, J. F. (1983), Maintaining knowledge about temporal intervals. Communications of the ACM. Volume 26 Issue 11, Nov. 1983, 832-843.

7. Mayrand, A. (2008). Functions of the Score: Foreshadowing. Retrieved 27th July, 2013 from the Getting the Score website: http://gettingthescore.com/?p=78

8. Young, D. (2012), Adaptive Game Music: the evolution and future of dynamic music systems in video games. Master's Thesis, Ohio University.

9. Marks, A. (2009), Game Audio Development. Delmar.

10. van Nispen tot Pannerden, T., Huiberts, S., Donders, S., & Koch, S. (2011). The nln-player: A system for nonlinear music in games. Paper presented at Proceedings of the International Computer Music Conference 2011, University of Huddersfield, England.

11. K. B. McAlpine, M. Bett, and J. Scanlan (2009). Approaches to creating real-time adaptive music in interactive entertainment: A musical perspective. In: The Proceedings of the AES 35th International Conference. AES Royal Academy of Engineering.

12. Prendergast, R. (1977), Film Music, a neglected art. W. W. Norton.

13. A wiki with examples of Mickey Mousing in video games can be found here: http://tvtropes.org/pmwiki/pmwiki.php/Main/MickeyMousing

14. Huiberts, S. & van Tol, R. (2008). IEZA: A Framework For Game Audio. Gamasutra. Retrieved January 23rd, 2008, from http://www.gamasutra.com/view/feature/3509/ieza_a_framework_for_game_audio.php

15. Silk, P. (2010). iMUSE Demonstration 3 - Other Tricks. Retrieved July 2nd, 2013, from http://www.youtube.com/watch?v=-XuClagw6IQ

16. http://downloads.gluddle.com/gluddle-musicdemo.mp4

17. http://downloads.gluddle.com/gluddle-musicexplanation.mp4

18. Burns, M. (2009). Planckogenesis, Part I: Quantizing Events. Retrieved July 25th, 2013, from: http://www.shadegrowngames.com/blog/2009/11/10/planckogenesis-part-i-quantizing-events.html

19. Huiberts, S. (2011), Listen! - Improving the Cooperation between Game Designers and Audio Designers. In: Think Design Play: The fifth international conference of the Digital Research Association (DIGRA). DiGRA/Utrecht School of the Arts. Retrieved from: http://www.digra.org/wp-content/uploads/digital-library/11313.06472.pdf

20. Arlauskas, B. (2012). A Metronome For Everyone. Retrieved January 8th, 2012 from http://gl33k.com/blog/2011/6/27/a-metronome-for-everyone.html

21. Drake, S. (1975), The Role of Music in Galileo's Experiments. Scientific American, p. 98.

Games

Beat Buddy (2013), Threaks.

Beat Sneak Bandit (2012). Simogo.

Gluddle (2012). Creative Heroes. iOS version.

Guitar Hero (2006). Harmonix Music Systems, RedOctane / MTV Games.

Guitar Hero II (2006). Harmonix Music Systems, RedOctane / Activision.

Monkey Island 2: Le Chuck's Revenge (1991), LucasArts.

Mushroom Men (2008), Red Fly Studio, Gamecock Media Group.

Patapon (2007), Pyramid Japan, Sony Computer Entertainment.

Planck (2009), ShadeGrownGames.

Rez (PlayStation 2, 2002). United Game Artists, SEGA.

Superhexagon (2012), Terry Cavanagh.

Super Mario Galaxy (2007), Nintendo