Swarm Lake: A Game of Swarm Intelligence, Human Interaction and Collaborative Music Composition

M. Kaliakatsos–Papakostas
School of Music Studies
Aristotle University of Thessaloniki
maxk@mus.auth.gr

A. Floros
Audiovisual Signal Processing Lab.
Dept. of Audiovisual Arts
Ionian University
floros@ionio.gr

K. Drossos
Audiovisual Signal Processing Lab.
Dept. of Audiovisual Arts
Ionian University
kdrosos@ionio.gr

K. Koukoudis
Dept. of Audiovisual Arts
Ionian University
t11kouk@ionio.gr

M. Kyzalas
Dept. of Audiovisual Arts
Ionian University
t11kyza@ionio.gr

A. Kalantzis
Dept. of Audiovisual Arts
Ionian University
t08kala@ionio.gr

ABSTRACT

In this work we aim to combine a game platform with the concept of collaborative music synthesis. We use bio-inspired intelligence for developing a world - the Lake - where multiple tribes of artificial, autonomous agents live within, having survival as their ultimate goal. The tribes exhibit primitive social swarm-based behavior and intelligence, which is used for taking actions that will potentially allow to dominate the game world. Tribes' populations also demonstrate a number of physical properties that restrict their ability to act illimitably. Multiuser intervention is employed in parallel, affecting the automated decisions and the physical parameters of the tribes, thus infusing the gaming orientation of the application context. Finally, sound synthesis is achieved through a complex mapping scheme established between the events occurring in the Lake and the rhythmic, harmonic and dynamic-range parameters of an advanced, collaborative sound composition engine. This complex mapping scheme allows the production of interesting and complicated sonic patterns that follow the performance evolution in both objective and conceptual levels. The overall synthesis process is controlled by the conductor, a virtual entity that determines the synthesis evolution in a way that is very similar to directing an ensemble performance in real world.

1. INTRODUCTION

The continuous evolution of music synthesis techniques using automatic / algorithmic means has yield into well-formalized, perceptually–acceptable music compositions with rich characteristics. Multiple approaches for automated music composition exist, originating from a variety of scientific fields and disciplines. For example, natural phenomena represent an interesting alternative for formulating techniques for music synthesis, taking into account the diversity and complexity of bio–inspired algorithms that can lead to rich geometric and dynamic properties using rather simple rules [1]. Interactive evolution [2] on the other hand is a different approach for realizing music composition, that aims to specify and apply sets of objective aesthetic criteria (i.e. measures), with a parallel and particular focus on the development of music raters that will produce human–like evaluations of the derived music, using perceptual rules incorporated into the rather unexplored level of human noesis. Generally speaking, the state-of-the-art in the aforementioned music synthesis methodologies primarily comprises algorithms that adapt to the circumscribed musical directions by utilizing computationally intelligent strategies, including genetic programming [3].

Clearly, in order to allow the development of such algorithms, a connection is required to be established between mathematics and music, frequently under a generic framework of human–machine interaction. The direct involvement of human interactivity extends the capacity of the overall music synthesis algorithm that is typically following (and limited by) rules defined in the domain of computational intelligence and creativity [4]. Thus, state-of-the-art methodologies in all aspects of music composition (employing for example tone, rhythm, or even orchestration) can be re-formulated by incorporating direct human intervention, providing the potentiality to interdisciplinary explore the entire space of musical possibilities and allowing defying musical norms of human music perception.

Practical realizations of the above concept have been already reported in the literature. Interactive intelligent composition represents a typical example, where the music synthesis system tracks the human preference and is adapted to it in real-time, using intelligent algorithms. Although the definition of human preference and the lucid demarcation of the relative aesthetic features are per se subjects of open debate [5], usually it is expressed a) in terms of a selection-rating scheme, an approach that inherently suffers from user fatigue [6] or b) by letting the human user perform tasks in a direct fashion (i.e. playing an instrument) or indirectly (through appropriately adjusting target parameters in real time) [7]. Following the latter option, Cyber Composer was recently proposed [8], aiming to offer a gesture-driven interface for controlling the tonality and melody of the music synthesized. More advanced music interfaces have been also introduced, in an attempt to render collaborative music synthesis a reality. In [9], the authors presented ChoirMob, a collaborative singing-synthesis environment that is empowered by an interactive score-writing display [10] that coordinates participating performers, allowing for expressive and engaging music making using mobile equipment.

In this work we combine bio-inspired intelligence with a collaborative music synthesis interface, which provides flexible and easy to learn interaction paths with the performers, since it is defined under the scope of a game. The overall system encapsulates complex mapping structures between the intelligent algorithm and the game player / performer preferences in a way that is transparent to the user and the audience, allowing its perceptual transformation to an experimental game environment rather than a dedicated music synthesis platform. More specifically, our implementation relies on the well-known and widely employed swarm intelligence concept that defines the collective behavior of decentralized, self-organized systems [11].

The swarm populations are organized in different tribes that live within the game virtual world called ”the Lake”, thus deriving the system's name: Swarm Lake. The behavior of each tribe, as it is formed by the partial decisions made by each of its members is finally transformed to sound. The autonomous nature of the swarms behavior takes into account a number of physical parameters met in biological systems: it can be performed provided that there is enough energy, while it is motivated by primitive social behaviors and rules, including decisions for attacks to enemy populations. User engagement can affect the autonomous behavior of the wandering swarms, in terms of high level (strategic) decisions that are intended to achieve the ultimate game aim: tribe survival. This user intervention actually changes the autonomous behavior of the swarm agents rendering it forced, while incorporating additional human-oriented intelligence to the game. Finally, the fundamental concept for sound synthesis in swarm Lake is the creation of virtual instruments that are assigned to each tribe (and obviously controlled by the corresponding performer). This approach, as it will be explained next in Section 4, establishes a robust and flexible interactive sonic design environment, able to produce various forms of sound content.

The rest of the paper is organized as following: Section 2 contains an overview of the Swarm Lake environment, followed by the analysis of the autonomous agents' movement and the human interaction mechanism provided in Section 3. This analysis is necessary for discussing the details of the algorithmic framework used for composing the Swarm Lake sonic output. This discussion is the subject of Section 4. In Section 5, a brief evaluation of the overall synthesis approach is presented. Finally, Section 6 concludes the work and highlights specific issues that can be considered in the future for evolving the creative capacity of the Swarm Lake project.

2. SWARM LAKE OVERVIEW

Swarm lake, at its present prototype version, is a four player game which incorporates human interaction with swarm intelligence through handheld mobile devices. Furthermore, the game events are mapped to 4–voice music, providing the players with an audio spatial interpretation of moving objects, their states and actions. Specifically, the game is based on the movement of agents which belong to different "*tribes*" (one tribe for each player) and move autonomously by interacting with each other under simple social rules, if there is no human intervention. These rules are thoroughly explained in Section 3 and roughly incorporate attraction among same–tribe agents and repulsion between different–tribe agents. These social rules result into the formulation of several same–tribe agent clusters, called the "*herds*". Players may intervene to the agents' autonomous movement by assigning special movements to the herds of their tribe. These movements apply on a herd of the respective player's tribe and incorporate simple transportation to a specified point in the game's available space, capture of food, splitting the herd's agents apart and attacking an enemy herd. A graphical overview of the Swarm Lake architecture is provided in Figure 1.

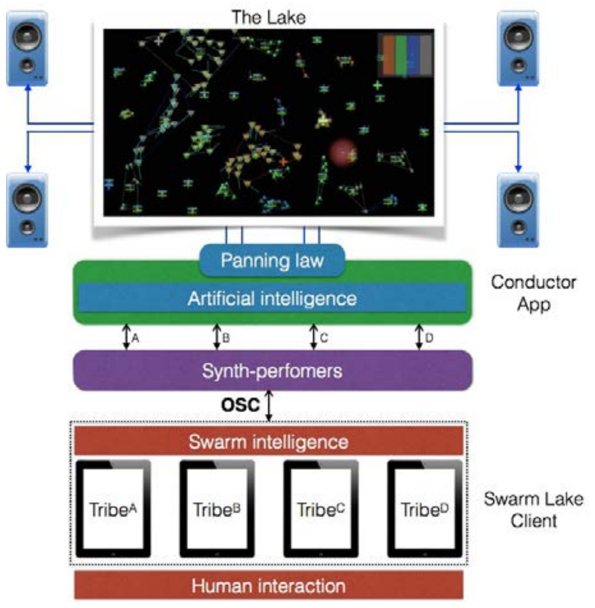


Figure 1. Swarm Lake architecture overview

Goal of the game is the survival of each player's agents for as long as possible. Each agent has an *energy level* which is depleted as the agent moves, or if the agent's herd looses a battle (following a procedure described thorougher in Section 3.3). An agent may regain some or all of its energy by being "healed" by its controlling player, as described in Section 3.2. Each player has an "energy repository" which may be used to heal the agents of a herd, or "resurrect" a number of previously diminished agents. The player's energy repository increases through proper herd manipulation, that is by winning battles, or ordering the herds to capture "food" packages offered by the Swarm Lake conductor. Therefore, each player's interaction with the game is generally realized through ordering herds of hers/his tribe.

If a parallelism with physical sciences may apply, the "atom" of the Swarm Lake is the agent, while the "molecule" is the herd. The agents encompass a set of simple social rules of interaction, which allow them to roam the available space in search of other agents of their tribe. When same-tribe agents "meet", they are able to form larger entities, the *herds*, which actually function as an individual mechanism. The herds however, do not have any kind of intelligence of their own, they simply *inherit* the collective social behavior that emerges from its constituent agents. Each agent has a "radius of perception", which signifies the distance below which it perceives objects and events. It is not necessary for two agents to perceive each other in order to pertain to the same herd. Two agents may be members of the same herd if there is a "chain" of agents that perceives each other, which links the aforementioned two agents. An illustrational example of the presence and the absence of this chain is given in Figure 2 (a) and (b) respectively. A herd may be constituting of an arbitrarily large set of agents, ranging from one to every agent of the tribe.

3. AGENTS' MOVEMENT AND HUMAN INTERACTION

Movement is separated in two parts, where the agents move in "*roaming*" mode or in "*forced*" mode.

Each agent has a particular position in the available game space (i.e. the Lake), which is updated taking into account the agents velocity. The latter in turn is adjusted by an acceleration coefficient which is determined by the perception of the agent and the interactions of the player.

3.1 Autonomous agent movement

The *roaming* agent movement is based on the Raynold's "*boids*" [12, 13] algorithms, with an adjustment for repulsion among agents of different tribes. The roaming movement of each agent is based on its perception of the world, i.e. objects and events outside its radius of perception do not affect its roaming mode move. There are four roaming movement rules, the first three of which are borrowed from the boids algorithm.

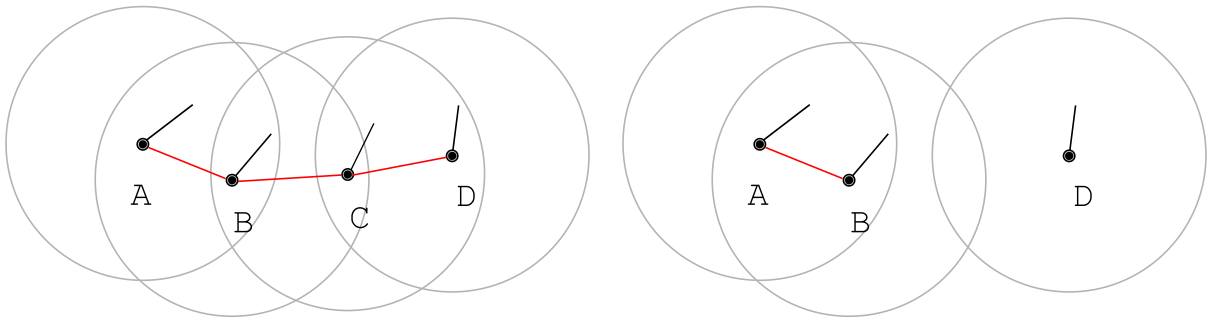
1. *Shoaling*: If an agent perceives a group of agents that belongs to its tribe (i.e. the herd it belongs to), then it moves towards the center of mass of the group that these agents form.
2. *Collision avoidance*: If two same–tribe agents are too close (below a predefined distance threshold) each agent moves away from each other.
3. *Schooling*: Each agent adjusts its velocity in accordance to the velocities of the same–tribe agents it perceives, i.e. it aligns its velocity with its herd mean velocity.
4. *Enemy repulsion*: Each agent moves away from the center of mass of the different–tribe agents it perceives (i.e. it is repealed by the enemy herds).

These rules define the roaming movement of each agent and result into the separation of agents to several clusters (i.e. herds) according to tribe. Each rule provides an acceleration coefficient defined for a certain direction and magnitude. All these acceleration parameters, together with a random coefficient of small magnitude, are then summed to produce the overall agents acceleration. Afterwards, the velocity is modified accordingly, and "trimmed" to a maximum velocity limit, which depends to the agent's tribe characteristics. Finally, the new position of the agent is defined by the updated velocity. However, the roaming movement rules are not considered likewise for some agents, when their controlling player interacts with the herd they belong to by providing orders. Another situation where these rules are not applicable as they appear above is when a herd is under attack by an enemy herd. These conditions constitute the *forced* movement which is analyzed in the following paragraphs.

3.2 Human interaction

The *forced* movement incorporates player interaction by providing orders to a herd she/he controls. When a particular herd movement is forced, the higher velocity limit of its comprising agents increases, in accordance to the characteristics of their tribe. Therefore, extensive utilization of forced movement dispenses the agents' energy, potentially leading them to faster extinction. Player intervention me be caused by giving the following orders to a herd of hers/his tribe:

1. *Move*: The herd moves to the specified location. To this end, only the roaming movement forces 2 and 4 are maintained, while an acceleration coefficient with direction towards the target is provided. The movement is completed if one of the herd's agents reaches the location specified by the player.
2. *Get Food*: This is equivalent to the "move" order, except from the fact that the target location is a food package. The herd captures the food if one of its agents approaches below a predefined threshold.
3. *Attack*: The selected herd becomes a *predator* and moves towards a selected enemy herd which becomes the *prey*. Only the roaming movement force 2 is maintained and an acceleration coefficient which is continuously pointing to the current position of the prey herd is given. If the predator reaches the prey, then a battle is taking place, the rules of which are described in Section 3.3. The attack is terminated without a battle, if the prey herd splits, or if it merges with an other herd.
4. *Split*: The agents pertaining to the selected herd move away from the center of mass of the set of agents that



(a) single herd (b) two herds

Figure 2. (a) Agents A, B, C and D belong to the same herd even though not all agents perceive the presence of the others (e.g. agent B and D). However, for every agent pair there is a chain of agents that perceive each other (for example D is connected to B via C). (b) If agent C is removed, then the chain is broken and thus agent D does not belong to the same herd as A and B do.

previously formed its herd, until it perceives none of them. As previously, during this movement, only the roaming movement acceleration 2 is maintained. The splitting actions may by utilized by the player as a defense mechanism, since such an action terminates potential attacks by enemy predators.

Each move may be terminated before it is executed if the controlling player provides a new order. While a herd is performing a certain move, it may encounter another herd of the same tribe. This will eventually merge the two herds in a unique one, which carries the players order. If two herds of the same tribe that have been given different orders meet, then the merged herd follows the orders of the most populated herd. If the herds are equally populated, then order to be followed by the merged herd is randomly selected from the targets of the previous two herds.

The player can also perform two additional tasks which do not incorporate herd movement. The first action is the herd *healing*, where the player heals all the agents in a herd. With this action, all the herd's agents obtain their initial energy level, while the energy "cost" required is subtracted from the player's repository. The second action is the agent *resurrection*, which incorporates the reappearance of agents that have previously demised. The energy "cost" for the resurrection of an agent equals to the total energy level of the agent that is about to enter the game. It has to be noted that the player is not allowed to "create" an agent, thus no resurrection can be performed if none of the player's agents has demised.

3.3 Battle rules

The battle rules apply when a predator herd, following the order of its controlling player, reaches the target prey herd. The winner herd is the one with the greater *fight strength*. The fight strength of a herd is determined by the energy level of the agents in the herd and by their relative positioning in space. The energy level of the herd, denoted by h_{nrg}, is the energy level sum of all its comprising agents. The positioning of the agents is measured with the *herd compactness* and is calculated by

$$h_{cmp} = \frac{1}{h_N^2} \sum_{i=1}^{h_N} \sum_{j=1}^{f_i} \left(\frac{s_i - d(i,j)}{s_i} \right)^k \tag{1}$$

where h_N is the number of the herd's agents, f_i is the number of agents that agent i perceives, s_i is the sight radius of agent i and $d(i,j)$ is the distance between agents i and j. The constant k can be characterized as an "overpopulation penalty", because large values of k result into lower compactness for herds with more agents. For the game setup as presented in this paper, a value $k = 2$ was utilized.

The energy level difference of the two herds that take part in the battle is stored in the energy repository of the player who controls the winning herd. The total energy level difference is divided with the number of agents in the loosing herd and the resulting number is the energy portion that each agent in the loosing herd has to "give off". This energy level is subtracted from each agent in the loosing herd, leading to demise of agents whose energy levels are vanished by this subtraction.

3.4 The Swarm Lake conductor

The Swarm Lake conductor represents a virtual software entity that affects the Lake and its enclosed population horizontally; that is, a decision made by the conductor potentially refers to all competitive tribes and within a larger application scope and affects the progress of the performance. More specifically, the conductor role as it is outlined by the game scenario is twofold: the application decides when and how much food is going to be offered in the Lake, also defining the specific place where it is going to be dropped. The final decision is made through a complex decision scheme, the analytic description of which is beyond the scope this work. In summary, it considers the mean and latest competitive activity of the tribes, as well as the energy values appeared in the local tribes repositories.

Clearly, the volume and the time intervals density of the offered food is a way for controlling the evolution of the music performance. For example, provided that the continuous absence of food will result into agents starving and a respective lack of energy, the game will soon come to an end and the music synthesis will stop. Thus, the decisions of the conductor can be regarded as a means for coding music score instructions that defines the length of the music track composed. In a future Swarm Lake version, this artificial entity can obviously be replaced by a human conductor. Moreover, the second conductor responsibility is to apply the 4-channel mixing of the synthesized content by considering the instantaneous spatial distribution of the autonomous agents within the Lake limits. More details on this issue are provided next in Section 4.3.

4. MUSIC COMPOSITION ALGORITHM

As it was mentioned at the beginning of the paper, sound synthesis is performed by assigning virtual instruments to each tribe. Each tribe-specific instrument is realized as a software SuperCollider[1] synthesizer ($Synth$), in terms of unit generators ($UGens$). This pure sound synthesis approach allows the production of the final sonic output, without the need of buffering prerecorded sound samples. Apart from the four $Synths$ that are mapped to each tribe, Swarm Lake also employs a fifth semi-independent $Synth$ element in order to represent the Swarm Lake world action and to fulfill sound synthesis conceptually, spatially and spectrally. We hereby refer to this synth as the *ghost* $Synth$. The rationale behind the definition of the above synthesizers is the realization of tribe-specific instruments with no actual reference to any real ones. All $Synths$ are designed with different parameters, in order to achieve unique sonic characteristics per tribe, thus achieving an interesting mix of sounds. This rather arbitrary design process is necessary for creating meaningful and organized tribe-specific sonic patterns, which fulfill fundamental human subjective expectations through providing an overall, robust and aesthetic sonic output.

4.1 Primary synthesis modules

Each tribe–specific synthesizer (denoted here as $Synth^A$, $Synth^B$, $Synth^C$ and $Synth^D$ respectively) is programmed to produce simple but distinguishable timbres that recall the epoch of the early generations of video games. The implementation of each of the $Synths$ included:

1. The definition of a function that analytically describes the $UGens$ employed within the specific $Synth$, as well as their interconnections (such as nesting, addition or multiplication).

2. The realization of multiple signal processing units (i.e. compressors, limiters and expanders) which provide more balanced sound under unpredictable or extreme amplitude changes that may occur during the game play.

3. The definition of specific $UGen$ arguments as interaction parameters, aiming to offer access and control to the respective generator tones (pitch), gain and ADSR envelopes duration, harmonics and phase offsets. These parameters depend on the particular type of a specific $UGen$.

[1] http://sourceforge.net/projects/supercollider/?source=directory

Following are the details of each of the $Synths$ realized for the purposes of this work.

$Synth^A$ consists of a series (i.e. an array) of sinus oscillators for creating a plurality of harmonic content. Signal amplitude is modulated using a saw oscillator combined with a low–pass filtering unit. A second sinusoidal signal is added to the previous one, which is phase–modulated using a saw oscillator too. Thus, the saw oscillator frequency becomes the frequency of the phase modulation, and it's amplitude the range of modulation. The above $Synth^A$ functionality can be mathematically modeled as:

$$\begin{aligned} Synth^A = &(Sin_1 + Sin_2 \ldots + Sin_V) * Saw \\ &+ Sin(phase : Saw) + Sin \end{aligned} \quad (2)$$

where the last term is added in order to control the plurality of the produced sound harmonic content following the variations of the compactness defined in Eq.(1) (see also Section 4.2 for more details). It should be noted that the same control task is performed by the last term of the equations following that describe the rest $Synths$.

$Synth^B$ incorporates a band limited impulse oscillator, whose harmonics are modulated by a sinus modulator. The derived signal is added with a sinusoidal and a sawtooth one as:

$$Synth^B = bl\_Imp(harms : Sin) + Sin + Saw \quad (3)$$

$Synth^C$ is defined by a low–pass filter pulse oscillator, a low–pass filter saw wave oscillator with a sinus oscillator being it's initial phase offset modulator, a sinusoidal oscillator as a phase modulator, a quadratic noise generator (QNG) and finally an extra generator being a bank of resonators fed by an impulse generator:

$$\begin{aligned} Synth^C = &(LFPulse + LFSaw(iphase : Sin) \\ &+ Sin(phase : sin) + QNG + Reson(Impulse) \end{aligned} \quad (4)$$

On the other hand, $Synth^D$ is modeled as

$$\begin{aligned} Synth^D = &RLPF(pulse) + RHPF(sin) \\ &+ Ring(impulse) + impulse(phase : sin) \end{aligned} \quad (5)$$

being the combination of a low–pass filter resonator (with cut–off frequency at $tone * 1$) with input a pulse wave generator, a high–pass resonant filter (with cut–off frequency at $tone * 2$) fed by a sinus oscillator, a ring modulator with input an impulse generator and finally, an impulse oscillator modulated by phase using a sinusoidal oscillator.

Finally, the $GhostSynth$ consists of an 8-input resonant filter fed by an equal number of sinusoidal oscillators, a ring modulator driven by a low–pass frequency saw wave oscillator and a chaos generator (i.e. a feedback sinus oscillator with chaotic phase indexing). The output of the chaos generator is particularly amplified when the tribes are in war. This $Synth$ is modeled as:

$$\begin{aligned} GhostSynth = &Reson(Sin_1 + Sin_2 \ldots + Sin_8) \\ &+ Ring(Sin(LFSaw)) + Chaos \end{aligned} \quad (6)$$

In order to efficiently follow the performance progress, each $Synth$ is controlled by one *synth–performer*, a software routine that triggers the initiation of sound events with specific sound parameters, following a pre-defined map of interactions. Thus, these *synth–performers* represent the application interface between the Swarm Lake world events and the sound engine. A detailed description of the sound parameters and the considered map of interactions is provided next.

4.2 Music parameters configuration

Tempo is pre-defined and fixed during a performance session. On the other hand, *rhythm* is defined in real–time, following the evolution of the game. Each *synth–performer* handles a sequence of notes with different durations. For example, when a tribe dies, a different global rhythm is applied, aiming to deliver a dramatic character on the game play, mainly through dividing the music measure into shorter note durations. During a game session, three different rhythmic changes can occur:

1. Initial rhythmic pattern applied upon starting a new game: the music measure M is divided in four beats ($[\tau_1, \tau_2, \tau_3, \tau_4]$), where τ denotes single note durations.

2. As it was mentioned previously, upon death of a tribe the music measure M is further divided into two more beats (i.e. $[\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6]$ for the first dead tribe, $[\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8]$ for the second one, etc).

3. When the next to last tribe dies, the game ends and music synthesis stops (i.e. $M = 0$).

Every time a new rhythmic section is about to be applied, a new fixed array of durations is sent to each *synth–performer*. All arrays have the same sum of durations that follows the applied rhythmic pattern; however, each *synth–performer* plays a different sequence of the globally defined durations, for example:

- $Synth^A \rightarrow [\tau_1, \tau_2, \tau_3, \tau_4]$
- $Synth^B \rightarrow [\tau_3, \tau_1, \tau_4, \tau_3]$
- $Synth^C \rightarrow [\tau_1, \tau_3, \tau_2, \tau_4]$
- $Synth^D \rightarrow [\tau_4, \tau_2, \tau_1, \tau_3]$

while the $GhostSynth$ always plays one note within a single music measure (i.e. $GhostSynth \rightarrow M$). Following the above rhythm description scheme, we ensure that all *synth–performers* are synchronized at the end of each rhythmic pattern.

Focusing on the produced sound *dynamics*, these are defined and controlled in a per tribe basis through appropriately shaped ADSR envelopes. Specifically, each *synth–performer* is assigned a specific ADSR curve, with A, D, S and R durations and gain values being varied in a range that is unique for the specific tribe, thus providing a distinct dynamic character of the sonic output produced by each one. The ADSR gains are analogously mapped to the average herd strength value for a specific tribe (as it was defined previously in Section 3.3), while the corresponding ADSR durations are directly associated with the mean herds energy h_{nrg}. By following this mapping approach, high sound dynamics are produced by tribes with excellent physical condition. It should be also noted that the ADSR envelope for the $GhostSynth$ remains constant during a specific game period. Moreover, the overall $GhostSynth$ reproduction level depends on the number of the dead tribes. Initially, the $GhostSynth$ is not audible, since all tribes are alive, but its level is increasing as the competition between the game tribes evolves.

The *melodic content* of the produced sound is continuously determined by the tribe that is dominant in terms of strength. Each time a tribe becomes a dominant one, all the other ones are enforced to follow the melodic character assigned to the leader tribe, as it is summarized in Table 1. Technically, this enforcement is applied by sending specific arrays of notes to the particular *synth–performers*. The array of the leader tribe has always double length. For example, if tribe D is assumed to be dominant, music synthesis will be based on the harmonic minor: tribe D *synth–performer* will play the complete musical scale, while the rest will be randomly playing the first, third, fifth and seventh scale note. Hence, the leader tribe is perceived as if it executes a music solo session, while the rest bondservant tribes are musically following it. A specific note (e.g. C) is defined per game session as the root of the reproduced scale. Obviously, if there is no leading tribe alternation, then the scale of notes no longer changes and no new arrays of notes are generated. This condition results into a lack of melodic changes under decreased tribe competition conditions.

Tribe synth	Assigned melody type
$Synth^A$	Melodic Major
$Synth^B$	Major Pentatonic
$Synth^C$	Melodic Minor
$Synth^D$	Harmonic Minor

Table 1. Melodic assignments to the Swarm Lake tribes.

Changing note octaves depends on the average of the absolute velocity magnitude of each tribe. For example, if a specific herd is about to attack, it will increase the average speed magnitude for the specific tribe. In this case, the reproduced notes for this specific tribe will be selected from a higher octave. Instead, if the tribes are moving leisurely within the Lake, then the sonic output will have a low frequency character. Moreover, the harmonic content distribution of the sound produced by each tribe is controlled by its compactness h_{cmp}: the more compact is the population of one tribe, the richer is the harmonic content of the music outcome, thus partially affecting the corresponding timbre characteristics. This extent of harmonic content is programmatically controlled through directly mapping the measured compactness to the amplitude of the last terms of Eqs.(2)–(5) presented in Section 4.1.

4.3 Sonic output final rendering

As it was previously mentioned, sound reproduction is performed under a 4–channel loudspeaker setup. For producing this multichannel audio stream, we have used a four channel equal power panning law that is realized on the conductor application side. Taking into account that the conductor is also responsible for delivering energy through food parcels to the Lake world, it can be considered that the conductor application is acting as a real conductor does for adjusting the overall balance of the various instruments or voices within his music ensemble.

Technically speaking, the panning law applied follows the tribe position (x_{tribe}, y_{tribe}) calculated as the average position among the overall amount of a tribe agents (the center of the Lake world corresponds to the $(0, 0)$ reference point of the coordinate system employed for defining the positions of the agents within the game world). Only the $GhostSynth$ sonic outcome is mapped in a different way. It's position (x_{ghost}, y_{ghost}) is defined as the symmetrically opposite point that corresponds to the average position of the four tribes, that is:

$$x_{ghost} = -(x_{tribe^A} + x_{tribe^B} + x_{tribe^C} + x_{tribe^D})/4 \quad (7)$$

and

$$y_{ghost} = -(y_{tribe^A} + y_{tribe^B} + y_{tribe^C} + y_{tribe^D})/4 \quad (8)$$

Hence, in a conceptual level, the $GhostSynth$ acoustically exists where the evidences of life are limited (or *where no life dwells*), while it is strengthen by death.

5. DEMONSTRATION AND EVALUATION

An essential task in this work was to evaluate the Swarm Lake game in real conditions, mainly in terms of a music-making collaborative tool rather than a game. Hence, for this performance assessment we particularly focused on the ability of the game to foster meaningful, subjectively acceptable and rich musical experience for both players / music makers, as well as the audience. The evaluation was performed using the Swarm Lake pre-release prototype, aiming to disclose (a) potential failures of the application that would imply low perceptual and user engagement efficiency and (b) possible success of the design alternatives selected, as well as (c) to designate realization best-practices that will allow further improvement of the creative process of composing music.

The Swarm Lake ensemble was assembled by four players, each one being the human *controller* of one of the existing swarm tribes. All players / music makers had formal music training and skills, while three of them were additionally involved in at least one algorithmic music synthesis project. Moreover, they all belonged to the Swarm Lake developing team, being aware of the scope of the game. Hence, no player training session was performed prior to the demonstration. The overall evaluation task included a complete game session, terminated when three of the four participating tribes were eliminated. For monitoring purposes, the particular agent/herd units positions were communicated to the conductor application through OSC messages. The conductor was also responsible for synthesizing the instantaneous graphical representation of the game world in real time. This video stream was projected on a wall, being available to all the music makers, as well as the audience which attended this pilot performance. The Swarm Lake client application that was available to each player was developed on the iOS mobile platform and was executed on a 3^{rd} generation iPad. It encapsulated all the necessary functionality for supporting the game scenario and interaction as it is described in the previous Sections. Through this application, each player was able to visually monitor his own tribe spatial distribution and provide his orders. For the overall Swarm Lake world representation, he should follow the visual stream produced by the conductor application. Finally, a dedicated wireless network was available for establishing the OSC communication links between the Swarm Lake clients and the conductor.

The 4–channel sonic output derived during the performed game session was recorded for future reference and evaluation by third parties. A short stereo downmix example, as well as the respective separated $Synth$ tracks are available online [2]. The music track composed was found to exhibit rich and interesting musical characteristics, while a significant evaluation outcome was the distinct musical character of each tribe incorporated into the derived music experience by the tribe dominance-dependent mechanism employed. Moreover, conceptually, the produced music content was found to perfectly match the evolution of the game, with the chaotic influence of the *Ghost Synth* being prominent in cases were the competition among the tribes was high. Finally, the artificial conductor involvement achieved an overall performance progress in terms of dynamics evolution and music track ending, which was found to be very similar to the one achieved by typical music ensemble directing schemes.

6. CONCLUSIONS

The concept of collaborative music making through mobile platforms is well investigated and employed in modern sound synthesis applications targeted to multiple composers' ensembles that perform concurrently. This concept is usually realized under the scope of efficiently described instructions that substitute the necessity of following a well-defined, organized score form. In the work at hand we introduce an alternative approach that considers a multiplayer game environment for realizing the collaborative music making system. Numerous advantages originate from this approach, such as the fact that the performers' excessive training is not required, since it is likely that the majority of participating humans are used to play at least one kind of video games. Moreover, the employment of artificial (and specifically bio–inspired) intelligence combined with the ability of the performer to interact with the autonomous artificial entities provide a novel and promising means for further experimentations on algorithmic sonic

[2] https://dl.dropboxusercontent.com/u/3975478/SwarmLakeAudio.zip

creation.

We have developed a system prototype, namely the Swarm Lake project, that was demonstrated aiming to provide an initial assessment of the music–making approach, mainly in terms of the musical experience achieved and the subjective evaluation of the sonic outcome. It was found that the complexity of mapping between the social, swarm behavior rules followed by the autonomous, intelligent agents and the highly parameterized music synthesis engine results into perceptually consistent and interesting forms of sound, in both practical and conceptual levels. In the latter case, several correspondences between the game group of music–makers and a real-work music ensemble are highlighted, such as the impact of the artificial conductor that controls the music dynamics, the reproduction panning and the overall length of the synthesized sonic content. Moreover, increased tribe activity and competition towards the ultimate survival aim results into an evolutionary music structure free from repetition artifacts and static characteristics.

The preliminary demonstration of the Swarm Lake project provided useful insights that will be considered towards the evolution of the prototype into a fully functional performance system. For example, the substitution of the artificial conductor entity by a human player will definitely enhance the user interaction significance in the overall sonic synthesis. The participation of an arbitrary number of tribes represents also an attractive future enhancement, especially under the perspective that each performer will be allowed to re-define the inherent properties and parameters of its own $Synth$ engine. This feature can be easily incorporated into the present version of Swarm Lake, due to the modular, $Synth$-based design approach followed. Finally, it is in the authors near future intentions to incorporate emotionally driven behavioral and sound synthesis rules in the artificial populations, that will complicate the game progress and provide an affective-rich sonic outcome.

7. REFERENCES

[1] M. Kaliakatsos-Papakostas, A. Floros, N. G. Kanellopoulos, and M. Vrahatis, "Genetic evolution of l and fl–systems for the production of rhythmic sequences," in *Proceedings of the* 14^{th} *international conference on Genetic and evolutionary computation conference companion (GECCO12).* New York: ACM, 2012, pp. 461–468.

[2] M. Kaliakatsos-Papakostas, M. Epitropakis, A. Floros, and M. Vrahatis, "Controlling interactive evolution of 8–bit melodies with genetic programming," *Soft Computing*, vol. 16, no. 12, pp. 1997–2008, 2012.

[3] A. Burton and T. Vladimirova, "Generation of musical sequences with genetic techniques," *Computer Music Journal*, vol. 23, no. 4, pp. 59–73, 1999.

[4] B. Manaris, J. Romero, P. Machado, D. Krehbiel, T. Hirzel, W. Pharr, and R. Davis, "Zipfs law, music classification, and aesthetics," *Computer Music Journal*, vol. 29, no. 1, pp. 55–69, 2005.

[5] B. Manaris, P. Roos, P. Machado, D. Krehbiel, L. Pellicoro, and J. Romero, "A corpus-based hybrid approach to music analysis and composition," in *Proceedings of the 22nd national conference on Artificial intelligence.* Vancouver, British Columbia, Canada: AAAI Press, 2007, pp. 839–845.

[6] J. Yan and Y. Min, "User fatigue in interactive evolutionary computation," *Applied Mechanics and Materials*, vol. 48–49, pp. 1333–1336, 2011.

[7] M. Kaliakatsos-Papakostas, A. Floros, and M. Vrahatis, "Intelligent music composition," in *Swarm Intelligence and Bio-Inspired Computation*, Y. X, Z. Cui, R. Xiao, A. Gandomi, and M. Karamanoglu, Eds. ELSEVIER, 2013.

[8] H. Ip, K. Law, and B. Kwong, "Cyber composer: Hand gesture-driven intelligent music composition and generation," in *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International*, Jan 2005, pp. 46–52.

[9] N. dAlessandro, A. Pon, J. Wang, D. Eagle, E. Sharlin, and S. Fels, "A digital mobile choir: Joining two interfaces towards composing and performing collaborative mobile music," in *Proceedings of the 2012 New Interfaces for Musical Expression Conference (NIME 2012).* University of Michigan, Ann Arbor: University of Michigan, May 2012.

[10] A. Pon, J. Ichino, E. Sharlin, D. Eagle, N. d Alessandro, and S. Carpendale, "Vuzik: A painting graphic score interface for composing and control of sound generation," in *Proceedings of the 2012 International Computer Music Conference*, Ljubljana, Slovenia, September 2012.

[11] A. Engelbrecht, *Fundamentals of Computational Swarm Intelligence.* John Wiley & Sons, 2006.

[12] C. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ser. ACM SIGGRAPH, 1987, pp. 25–34.

[13] ——, "Not bumping into things," in *in the notes for the SIGGRAPH 88 course Developments in Physically-Based Modeling*, ser. ACM SIGGRAPH, 1988, pp. G1–G13.