EvoBoids: Co-Design of a Physical and Virtual Game using Artificial Evolution

David C. Alvarez-Charris
University Autónoma de Occidente
Cali, Colombia
david13.ing@gmail.com

Héctor F. Satizábal
University of Applied Sciences Western Switzerland
Yverdon-les-Bains, Switzerland
hector-fabio.satizabal-mejia@heig-vd.ch

Andrés Pérez-Uribe
University of Applied Sciences Western Switzerland
Yverdon-les-Bains, Switzerland
andres.perez-uribe@heig-vd.ch

Jesús A. López
University Autónoma de Occidente
Cali, Colombia
jalopez@uao.edu.co

***Abstract*— We present the co–design of a gaming scenario between an Artificial Evolution algorithm and a human designer. Such co–design is twofold, consisting of an initial stage in which a genetic algorithm is used to evolve the control parameters that define the behavior of a group of virtual agents. This produces interesting and unexpected results not only creating differentiated behaviors but also increasing the flexibility of the character to adapt to a given objective. In the second stage of the game design a human interplay was introduced in adding other elements to the game, such as other characters and new game dynamics. In this paper we introduce a game that integrates virtual and physical characters while taking advantage of such co-design approach. The physical character consists of a robot which controlled through a Natural User Interface can be part of the game by interacting with other characters in the virtual environment.**

Keywords— Virtual agents, Artificial Evolution, Physical/Virtual Interaction, Natural User Interface

I. INTRODUCTION

Computational Intelligence (CI) techniques have been broadly used in Video Games, in great part because of the interesting scenarios that video games can offer to test the performance of algorithms. These scenarios are of great use, since they can capture the complex dynamics and competitive elements of the real world, thus offering challenging tasks, and at the same time maintain the controllability and traceability of a computer simulation were all the variables, scores, feedback and simulation parameters can be easily accessed and configured. In addition to this, not only does CI benefit of using games as a test bed, but also, video games can obtain advantages by using this type of techniques. The application of CI into video games can provide solutions to create more realistic scenarios, intelligent characters with appealing interactions, or the adaptation of the game according to the user's level of motivation [1]. CI has been used in video games in a vast amount of applications. For example, in Procedural Content Generation [2], algorithms with limited or indirect user input are used for the creation of game content, such as characters, levels, items, game rules, etc. An approach used to generate such content is based on search, where by using evolutionary algorithm, random search, or any other stochastic optimization method, content with certain desired properties is generated. This is an incremental process in which the search space is explored producing changes on content, by keeping changes which make the solution better and discarding those that are harmful, the desired properties are eventually achieved [2]. Another application is the creation of characters that learn how to play a game well. This is the case of Non-Player Characters (NPC) [3], which use reinforcement learning or evolutionary algorithms to learn behaviors that enable them to win a game. In this type of applications, the game is seen as a reinforcement learning problem, in which policies are constructed based on the score of the game.

The aim of this project is to use artificial evolution as a tool that helps the designer of a game, by determining the set of control parameters that enable a group of virtual agents to exhibit a swarm behavior, coordinating their actions to collectively achieve the completion of a given task. In addition to this, we explore the use of evolutionary techniques to allow the group of agents to adapt to different objectives, by modifying the individual and the overall swarm behavior. To illustrate this co-design approach, we present a game integrating both, physical and virtual characters, and a human player that interacts as the leader of a virtual swarm of agents.

This paper is organized as follows: Section II presents the proposed methodology. The model used to achieve a swarm behavior is detailed, the genetic encoding and fitness function used in the evolutionary algorithm are described, and the human design process and physical-virtual interaction are described. Section III describes the results showing how the boids model achieved a cohesive behavior, the Genetic Algorithm (GA) generated highly fit solutions, and the different gameplays that where constructed along with the interaction of the user and physical characters with the virtual elements Finally, in Section IV a discussion summarizing this work and its result is done.

II. METHODS

To start, we identified the need to endow the main character of our game, herein referred as a casual agent, with a cooperative behavior and with the capability of adapting to different game objectives. The proposed methodology is based on the use of Genetic Algorithms (GA) to achieve the adaptability of the swarm and to modify the control parameters, aiding in the design of a video game. GA have the capacity to explore wide search areas using a population of candidates and because it does not require any previous information about the fitness landscape [4], yet generating efficient and powerful results. The GA was chosen based on the need of solving the engineering problem of optimizing the boids behavior. It is beyond the scope of this work to evaluate and compare multiple evolutionary algorithms. Future work described in Section IV shed light over possible improvements for the evolutionary algorithm. The approach used towards a collective behavior was to implement a model in which through local interactions a group of agents exhibits swarming behaviors. The model is known as Boids [5].

A. *Boids Model and Simulation Environment*

The first step was to construct our own simulation scenario to emulate each boid (virtual agent) using the Python programming language. Each of the virtual agents (referred to as A) was created as a single geometrical object [5] consisting of a circle with a special set of properties: a two dimensional position in the simulation environment $P^A = X^A, Y^A$ and a velocity in both directions $V^A = V_x^A$, V_y^A providing the agent an orientation. The casual agents (also known as boid) are agents whose behavior is determined by the boids model Fig.1, a. Each of the boids has a simulated perception of the other agents, limited by a local circular zone of vision defined by the neighborhood radius R_n (Light ring, Fig.1, a). The agent has access to the position and velocity of the M flock mates (smaller dark agents positioned inside light ring, Fig.1, a) within its neighborhood, allowing it to apply the boids set of rule and compute its new state at each time step of the simulation. It is important to notice that the membership to the neighborhood is obtained by computing the Euclidian distance between the two agents. The set which includes all the N casual agents in the virtual environment is denoted by $S_{ca.}$

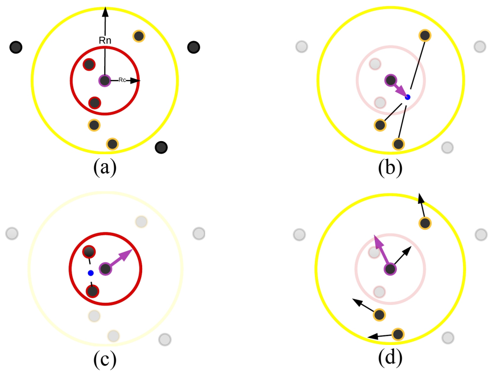


Fig. 1. (a) Boids model and radii, (b) Cohesion rule, (c) Repulsion rule, (d) Alignment rule

The three rules that determine the agent's behavior can be described as follows. The cohesion rule, mathematically described by (1), consists in modifying the agent's velocity (arrow pointing out of dark agent at the center, Fig.1, b) such that it produces a shift in its position that moves him closer to the center of mass of the local flock (small dot, Fig.1, b). In contrast, the repulsion rule described by (2) changes the velocity in a way that the agent moves away from the center of mass of its local collision group. This group is made out of the H agents (dark agents, Fig.1, c) lying inside the collision radius R_c. The alignment rule (3) adjusts the velocity in a way that on the next time step the agent will steer towards the average heading of the local flock (Fig.1, d)

Cohesion:

$$V_x^A = V_x^A + \left(\frac{1}{M} \sum_{j=0}^{M-1} X^j - X^A \right) . W_C , \forall \left\{ j \mid R_c < \left\| P^A P^j \right\| < R_n \right\} \quad (1)$$

Repulsion:

$$V_x^A = V_x^A + \left(\frac{1}{H} \sum_{j=0}^{M-1} X^A - X^j \right) . W_R , \forall \left\{ j \mid \left\| P^A P^j \right\| < R_c \right\} \quad (2)$$

Alignment:

$$V_x^A = V_x^A + \left(\frac{1}{M} \sum_{j=0}^{M-1} X^j \right) . W_A , \forall \left\{ j \mid R_c < \left\| P^A P^j \right\| < R_n \right\} \quad (3)$$

Each rule produces a non-correlated modification to the agent's velocity. These modifications are merged into a single value using a linear combination, where each of the parts is multiplied by a weight W_C, W_R, W_A.

B. *Evolution of Controllers*

As mentioned before, the main ideas underlying the use of Artificial Evolution are to enhance the process of the games' design. This, by autonomously generating control parameters that produce flocking behaviors that through manual adjustments would not have been easily achieved. Such behavior must respond to the need of the swarm to adapt to a given objective in the game. For this reason, in the initial stage of this work we added three additional agents: A predator agent, a leader agent and an obstacle agent. Each of these characters had a parameter which enabled modifying the reaction a casual boid had towards a specific character.

As a case study for the scope of this project the leader character was used as a mean to create a new objective in the game. The new objective was to achieve a moderately compact swarm exhibiting a flocking behavior that could use the leader agent as the head of the swarm guiding its movement trajectory. By "moderately compact swarm" we refer to the type of grouping behavior present in schools of fish or flock of birds [6], where even though individuals exhibit the capacity to move at impressive speeds in a uniform group, they can also alter the shape and level of compactness of the flock to avoid an obstacle or re allocate members within the group. To reach this objective we decided to construct each potential individual (solution) with a genetic representation encoding the parameters which produced the most significant changes on the swarm's behavior. That is, both vision radius (R_n and R_c) which drastically affect how global the communication between agents is, and the group of principal and auxiliary weights W_C , W_R , W_A and W_L (leaders weight). That have the highest impact in the swarms' behavior greatly controlling in a complex manner the inter agent interaction. A summary of the genotype is presented in Table I:

Table I. Genetic Representation of Individuals (Solutions) to Evolve

Gene	R_n	R_c	W_C	W_R	W_A	W_L
Value Range	0 - 150	0 - 150	0 - 1	0 - 1	0 - 1	0 - 10

The fitness function evaluating the performance of each individual (solution), consisted of a linear combination of four components (all normalized), making this search a multi-objective optimization problem. The first, the inter agent collision f_{ac}, measures the average amount of casual – casual agent collisions that occurred during the evaluation of an individual (i.e. the total run time of the simulation). This parameter was measured by computing if the Euclidian distance between agent A and its surrounding neighbors was less than or equal to twice the agents size. If such evaluation was true, it indicated that two (or more) agents had partially or totally overlapped, implying that a collision had taken place.

The second parameter is the number of agents at goal f_{ag}, showing how close the casual agents are from the leader during the run time of the simulation. In order to have a heuristic which indicated how likely were the casual agents to follow the leader, we created a circular area (termed goal area) centered in the leader's position. The proposed heuristic establishes that each time a casual agent stands within this area; the agent is currently following the leader. Consequently, by obtaining the amount of agents that stand inside the area and averaging it for the total simulation time, we are indirectly measuring the rate at which the leader following behavior was manifested.

As described above by using the heuristic implemented with *fag* we can estimate when a casual agent is close to the leader. However, being able to measure how many agents are inside the goal area is not enough. In addition we must be able to determine the way in which such agents arrived. Namely, we must measure if the agents arrived as a group, or if each one of them independently reached the goal. To suffice this, a third parameter delta time of arrival $f_{\Delta T}$ was introduced indirectly measuring the frequency at which the agents arrived to the goal. Specifically, what we measured was the elapsed time between the arrival of one agent and the other.

The fourth parameter is the median group radius f_{mr}. This parameter indicates how disperse or compact is the swarm of boids by measuring the average mean distance between the casual boids position and the center of the group, constituted by agents of the set S_{ca}. The measurement of f_{mr} is as follows. The center of the group C_{ca} is extracted by using the position of all the agents belonging to S_{ca} and computing the average as showed in (4). Then the Euclidian distance between the center of the swarm and each of the members of the set is computed and stored in a list. Finally, by knowing which is the mean value of this list for each time step of the simulation, an average value over the total number of iterations T_T can be calculated, obtaining f_{mr} (5). The values produced by f_{mr} were used as the input argument to a Gaussian Function as showed in (6), producing the modified parameter f_{Gmr}. This function was constructed such that individuals with a phenotype producing either heavily disperse or over compacted group of agents where strongly penalized. In contrast, agents exhibiting an intermediate compactness between the above mentioned extreme situations, were rewarded with a higher f_{Gmr} value. The Gaussian's peak r_{ideal}(mean or expectation) was set to an "ideal" mean group radius which was experimentally established following the guidelines previously described. Similarly, a standard deviation σ was also experimentally determined; controlling how acceptable are the described extreme situations.

Average Median Group Radius

$$C_{ca} = \left[\sum_{j=0}^{N-1} X^j , \sum_{j=0}^{N-1} Y^j \right], \forall \{ j \in S_{ca} \} \tag{4}$$

$$f_{mr} = \frac{1}{T_T} \sum_{0}^{T_T - 1} median \left\{ \left\| C_{ca} P^j \right\| \forall \{ j \in S_{ca} \} \right\} \tag{5}$$

$$f_{Gmr} = e^{\frac{(f_{mr} - r_{ideal})^2}{2.\sigma^2}}, r_{ideal} = 100, \sigma = 40 \tag{6}$$

Due to the stochastic behavior and short term predictability of the boids model [7] in which a controller could by chance, converge to a different behavior of the swarm. A single evaluation of an individual would lead to a noisy fitness functions [8]. To lessen the stochastic nature of the agents and compute a more stable value of fitness, each of the individuals was evaluated several times generating a set of fitness values which were averaged to produce a final decisive fitness. Fig.2 clearly depicts many of the previously described features of the simulation environment and the evolutionary parameters.

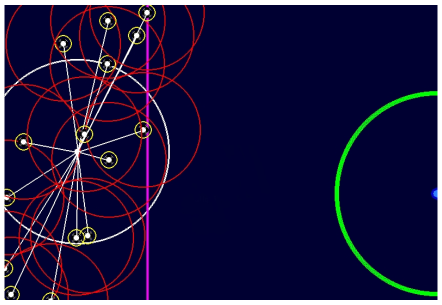


Fig. 2. Simulation environment during the Evolutionary Process. R_n (neighborhood radius) and R_c (collision radius) can respectively be identified as the big and small circles centered on each of the agent. As a white dot, the center of the group C_{ca} is presented and the f_{mr} (Median Group Radius) is represented as the white circle centered on this dot. The leader agent is represented as the dot on the far right of the scenario, it is surrounded by a circle of radius r_G which encompasses the goal area.

C. Human Crafted Gameplays

In a second stage of the design of the gaming environment a human user stepped in by carrying out two main tasks: the fine tuning of the controller and the creation of new gameplays. The fine tuning process was mainly carried out experimentally by trial and error. However, the time consumption was minimum given the fact that the adjustments that had to be performed where almost negligible and could be done by a quick visual inspection of the simulation. With respect to the creation of the new gameplays, a number of new features such as game characters, dynamics and graphical features were added, enhancing the games playability and making it more interesting.

The first feature added consisted in transforming the Predator Agent into a Non Player Character (NPC), exhibiting to a certain degree an autonomous behavior. The predator was endowed with a simulated perception limited by a radius which defined how “sharp” (big) was the predator’s sight. Any casual agent inside the sight of the predator would become a potential prey for him to hunt. Depending on the presence or absence of agents within the predator’s sight, it could exhibit to types of behaviors: Random Foraging or Prey Hunting. The Prey Hunting behavior is triggered whenever a casual agent lies within the predator’s sight; enabling the predator to follow and eventually hunt the preys that surround him. The speed at which a predator agent hunted its preys was initialized at a base speed V_p, and during the game the predators speed linearly decreases by a factor ΔV_p according to the number of preys within its sight. This mechanism translates in increasing the predator’s ability to hunt individual agents and decreasing its possibilities of capturing boids which remain in a swarm. The Random Foraging behavior, triggered when no casual agents lay within the predator’s sight. Is a simple behavior in which the velocity is modified by adding a real number sampled from a uniform distribution. In this way the predator randomly wonders around the scenario searching for preys.

Under certain circumstances in which the preys moved in a specific configuration, the predator agent started oscillating back and forth around the preys’, endlessly chasing the same group of preys but never successfully hunting them. To avoid this problem that perpetually locks the NPC in a loop, we introduced the surrender and precision actions. The surrender action is a mechanism that allows the predator to eventually abandon the hunting of a group of prey and restart exploring the scenario in look for new ones. Complementarily, the precision mechanism allowed the predator to reduce its hunting speed in order to achieve more accurate movements while approaching a prey. The speed reduction is directly proportional to the level of persistence. This allowed the predator to not only consider surrendering when no preys had been hunted for a certain amount of time steps, but to first consider adopting the strategy of gradually reducing its hunting speed to increase the probability of hunting a prey.

A second feature added to the gameplay is a character referred to as the Assistant Agent. This character consisted of an autonomous player (i.e. not controlled by the games user) whose main role was to search for dead casual agents and revive them. The assistant is able to localize any dead casual agents that are inside its vision radius and gradually approach to revive him. Similar to the predator agent, the assistant exhibits the same type of behavior by randomly exploring the scenario in look for dead agents, then targeting a specific agent to revive, and eventually abandoning this task if no successful revival action is performed within a limited time frame.

A third feature named Guider Agent, is another NPC whose function is to take the swarm of casual boids from an initial position A (left side of the scenario) to a final position B (right side of the scenario, target point). To achieve this conduct the guider agent has the capacity of attracting the casual boids and is equipped with a global vision of both, a target point and the complete set of casual boids. In this manner this character managed to produce a shift in its position that slowly moved him right, and at the same time it remained close to the group of casual agents to effectively take them to the target point. Whenever a casual boid was left behind, the guider agent went back, getting close to the casual boids and re uniting them in a single group. A final feature added, is the ability of the user to create obstacle objects which the autonomous characters try to avoid colliding against.

D. *Physical Character and Interfaces*

We decided to add a new component to the game which resides out of the virtual environment in which all the other agents where created: an E-puck mobile robot [8][9]. This robot served as a physical character which could directly interact with other virtual elements creating an interesting link between the physical and virtual world. Two main elements allowed us to create this physical – virtual interaction: The robot along with an overhead tracking system and a Natural User Interface (NUI).

The Natural User Interface consisted of a Kinect that kept track of each of the user’s joints, allowing him to command through gestures the mobile robot. The implemented gesture control algorithm required of capturing the position of the user’s head, hands and shoulders. Whenever the user raised its left hand above its head and the difference between the positions of both joints in the Y-direction was greater than a threshold distance, the control of the mobile robot was enabled. Once enabled the user could control the direction of the heading of the robot, between a straight motion and angular rotation. When the relative distance in the X-direction between the right shoulder and the right hand was zero or less than a small epsilon value the command sent to the robot was to establish the same speed in both wheels producing a forward motion. In comparison, when this distance was greater than a threshold, the users pose was considered as a left or right steering command, thus setting different speeds in the robots left and right wheels.

On the other end of the control loop the E-puck robot was programmed using ASEBA [10], an event-based programming language for real-time distributed control of mobile robots. The code stored inside the robot was trivial, it only had one event in which two variables corresponding to the speed of both wheels was established. The bridge between both programming languages (i.e. Kinect and E-puck robot) was established using an available API provided by ASEBA called ASEBA CMD. The wheel’s speeds where sent via Bluetooth to the robot’s microprocessor by using ASEBA Switch, a utility that can connect to specific targets differentiated by an identifier (each E-puck robot has a unique identifier).

In addition to the above, a feedback mechanism which connected the physical character to the virtual environment was created. Such feedback was achieved using computer vision techniques which enabled performing a visual tracking of the robot’s position. The tracking algorithm used the standard image processing pipeline: capturing the RGB image, converting it to gray scale to reduce the dimensionality of the input image, thresholding the input data to produce a binary image, performing blob detection and finally using a nearest neighbor tracking algorithm to trace the position of the desired blob (particle). It is important to highlight that the blob detection algorithm used two constraints (area and compactness) enforcing a greater stability of the tracking system. The complete visual tracking algorithm was implemented using the open source software SwissTrack [11]. After obtaining the coordinates of the robot in the physical scenario, they are sent through a TCP/IP port to the python game engine, were Swiss Track serves as a server and the gaming engine is the client. In the python algorithms the position of the leader character is set to the robot's position in the physical scenario.

III. Results

A. Boids Swarm Behavior

Using the boids model resulted in a bioinspired way to achieve a flocking behavior among the group of agents, inspiring itself in the movement of biological swarms. By manually modifying the main parameters through a long time consuming process, we could achieve a behavior which mimics the flocking behavior of birds, or schooling conducts of fish. The Boids simulation environment is presented in Fig.4.

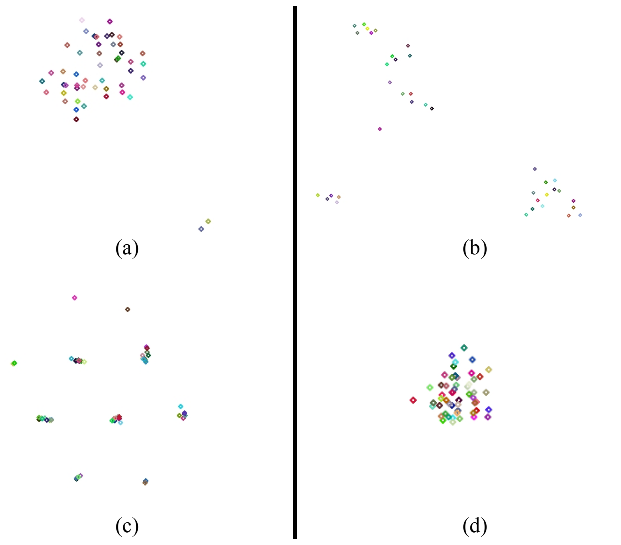
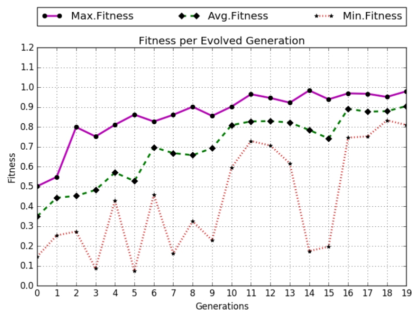


Fig. 4. Simulation environment showing the creation of flocks with different parameters. (a) Low W_c and high R_n, (b) High W_c and Low R_n, (c) Low W_R and high R_c, (d) High W_R and Low R_c,

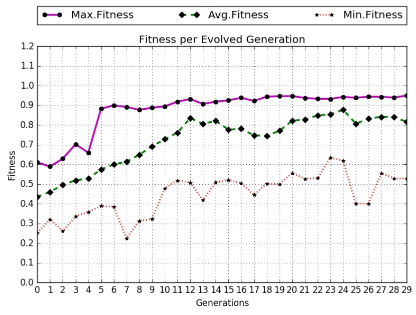
As depicted in the Fig.4, different flock behaviors could be achieved. For example, when we decreased W_c to a nearly null value but kept a moderately large radius R_n, the agents exhibited a highly compact behavior were a reduced number of small groups were formed and after some iterations they came together into a highly dense group (Fig.4, a). This showed an independence or low relevance of the cohesion weight when influencing the global behavior of the group and revealed the high impact that the level of communication has on the speed of convergence. This was further confirmed with the parameters used to produce the behavior depicted in Fig.4, b, in which a high cohesion weight with a moderately low Rn where established. Despite the high cohesion agents only formed multiple small groups sparse all over the scenario. By keeping a low W_R but increasing the collision radius R_c we achieved an unexpected behavior showed in Fig.4, c. In which numerous compact groups were formed, remaining heavily intertwined and never collapsing with the other neighboring clutters into a single flock. One interesting aspect of this behavior is that at the macroscopic scale the swarm formed a structured formation among clusters (hexagonal geometry) where each of the clusters positioned itself in a vertex and the "invisible" cohesion among clusters formed the hexagon's edges. This type of behavior is characteristic of Agent Ensembles, were permanent connections among the agent clusters cause the system to act as a single physical object [12], moving itself as a semi-rigid object along the scenario.

B. Adaptation Through Evolution

Manually tuning the parameters and achieving a desired behavior on purpose is not a straightforward process. Despite the boids model being solely composed of three interacting rules, it can be considered a complex system, having the unique property of short-term predictability, where the actions and movements of the agents can be predicted for a few couple of time steps, but beyond that, the deterministic aspect of boids fails. Short-term predictability is contrary to both, chaotic systems which are completely unpredictable, and periodic systems which are entirely predictable [6]. By using the Genetic Algorithm (GA) technique, we could overcome the gap that exists between mapping the individual boid behavior (micro) to the system/swarm behavior (macro).



(a)



(b)

Fig. 5. GA Fitness per Evolved Generation. Two solutions (many more solutions were generated, but only a subset of these are shown) gradually evolved generation by generation reaching acceptable fitness values. Each generation had population of 20 individuals.

Table II. Genetic Representation of Best Evolved Individuals (Solutions)

Gene	R_n	R_c	W_C	W_R	W_A	W_L^*
Ind1	150.00	30.29	2.71%	100.00%	71.16%	5.78
Ind2	150.00	32.07	0.67%	66.74%	10.99%	10.00

* W_L not showed in percentage since it is a scaling factor

The controller parameters of the best individuals within the generated solutions are presented in Table II. Based on the genotype of *Ind1* and *Ind2*, one can observe that both have a low cohesion and a very high repulsion value. Analyzing this results one could postulate that the GA found that this combination was ideal, since having a contrasting value to that found by the GA (i.e. A high W_C) would cause the boids from a neighborhood to converge to a small area (excessive attraction). In this manner increasing their chance of colliding and therefore raising the possibility of having agent inter collisions, decreasing the fitness parameter f_{ac}. By comparing the experimental fitness values showed in Table III for the swarms *Ind1* and *Ind2*, one can observe a direct relationship between having low cohesion and high repulsion values and getting a better f_{ac} (i.e. fewer agent collisions). Building upon this, a high W_R easily allows the agents to move close to each other at high velocity values and still prevent crashing against others. As achieved by *Ind1*, saturating this value to its maximum generates a policy which on average tries to totally avoid any collision and as a consequence the agents can get closer to each other. This is manifested as a lower collision radius R_c. The above insight can be clearly seen by doing a quick comparison between the values of W_R and R_c for the two satisfactory solutions found by the evolutionary algorithm. There certainly exist an inverse relationship between the distance at which two agents are comfortable of coming together and the level of rejection among agents. All of the individuals converged to having the largest allowed value for the neighborhood radius R_n. In this way they could increase the range of their simulated perception, more easily detecting the presence and position of the leader agent enabling them to approach him and increasing f_{ag}. In addition, a bigger amount of boids could be included within a single neighborhood radius, consequently creating more compact groups and increasing the value of the median group radius f_{Gmr}. At the same time if both, the compactness of the group and the chances of following the leader agent are increased, when a group of agents enters the goal area they will come in at more regular time intervals, also improving the fitness parameter corresponding to $f_{\Delta T}$.

Table III. Fitness Parameters Achieved by Best Evolved Individuals*

Parameter	f_{ac}	f_{ag}	$f_{\Delta T}$	f_{Gmr}	F
Ind1	0.95	0.99	0.97	0.75	0.93
Ind2	0.87	1.00	0.96	0.73	0.90

* Fitness values obtained after the end of evolution. Experimentally simulating the Swarms in a scenario in which the leader agent moved (not static)

Table IV. Priority Levels (Weights) for Fitness Components

Priority	W_{ac}	W_{ag}	$W_{\Delta T}$	W_{Gmr}
Value	0.30	0.35	0.15	0.20

The overall fitness values F presented in Table III were obtained through a linear combination of the fitness components using the weights presented in Table IV. The process of finding an ideal solution through the evolutionary algorithm did require the tuning of the priority levels (weights). By changing the values of the priority level we managed to exert different evolutionary pressures which guided the evolution towards each of its multiple objectives. Despite the fact that this tuning does require a manual trial and error approach, the process is remarkably simpler in comparison to tuning the six parameters that define the boids controller, that are highly dependent through complex inter relations and which have a nonlinear input–output correspondence. On the contrary, each of the priority levels is clearly defined in an intuitive and independent manner, where there is a direct association between the value of each component and a specific target behavior.

A numerical measurement of the emerging behavior of the evolved swarms is present in two boxplot contrasting characteristics of the best evolved individual (*Ind1*) versus an individual with a low fitness. The measured characteristic is the average between an agent's position and the group's centroid C_{ca} for each of the agents. The average is computed over all time steps of the simulation. As seen in Fig.6 the high fitness individual has a small mean value when compared to the low fitness individual, indicating that the swarm's level of compactness and communication for the former is much greater. This can be further evidenced by the reduced interquartile range of the high fitness individual.

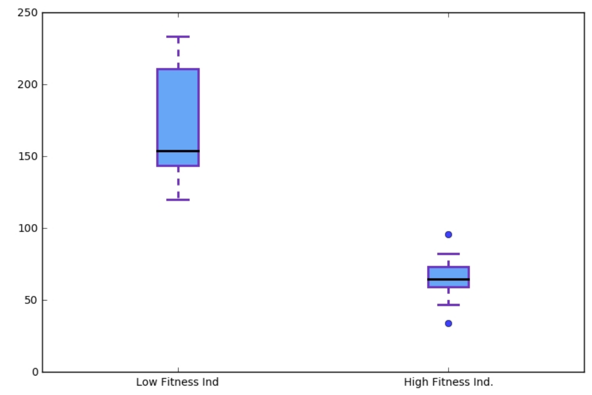


Fig. 6. Comparison of compactness values for high-fitness (right boxplot) and low-fitness (left boxplot) swarm's.

C. *Gameplays*

With the human crafted features, two types of games were created. The first gameplay consisted on an adversarial scenario where the two main characters, Leader and Predator, fought against each other. There are two teams: the leader's team constituted of the user controlled leader agent and the autonomous assistant agent, and the predators' team solely made out of various predator agents (Fig.7). The objective of the leader's team is to maximize the amount of casual agents that are alive by the end of the game. In contrast, the predator team has to hunt as many agents as possible. The assistant agent minimizes the amount of hunted agents by reviving them, and the role of the user is to attract and move the casual boids towards the safest positions, far from the random exploration areas of the predators. In addition, and with great importance towards a good user strategy, the game player should try to cluster the casual boids. Creating a compact group of agents and avoiding leaving individual boids wondering on their own, which for the predators are easy to hunt preys (because of its high speed against individual boids). When multiple predators where present during the gameplay we could observe a behavior similar to cooperative hunting, despite the fact that predators are not aware of each other and that they are not trying to explicitly cooperate, this result was possibly produced because individual predators had the same objectives. The underlying idea of the previously described game is that the swarm as a whole, cooperating among agents and aided by the leader, has a greater chance of survival upon an attack, in comparison to a behavior where each casual boid moves on its own without cooperation

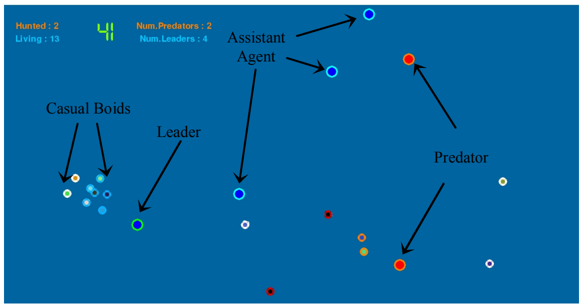
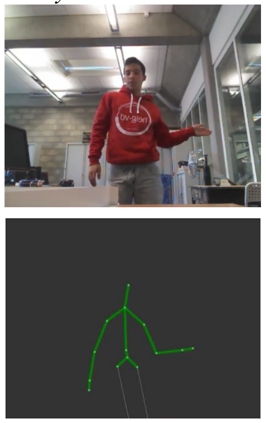


Fig. 7. Gameplay: Adversarial scenario, Leader's VS Predator's Team. User controlled Leader is currently wining the match by having 13 living Boids. The Predator Agents are losing the match by only hunting down two preys some of which have been revived by the Assistant Agent.

The guider agent which had the capability of moving the casual boids from one point of the gaming scenario to another, helped us to design a second gameplay, this time under the idea of having an adversarial scenario between the autonomously controlled guider agent and the user controlled predator. The guider character had to take the casual agents from their initial position on the left hand side of the scenario to the target point on the right side of the scenario. If the guider agent managed to at least safely position one casual boid within a small distance from the target point, the guider agent won. On the other hand, the user had to control the predator agent in order to either hunt down all of the casual agents and in this way prevent any boid from reaching the target point; or manage to keep the casual boids as far away of the target point until the game ended.

D. *Physical Virtual Interaction*



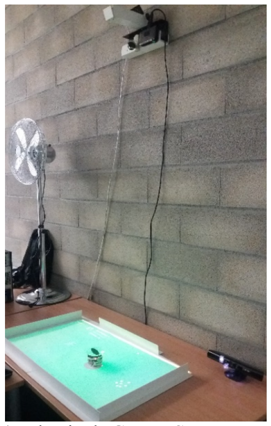
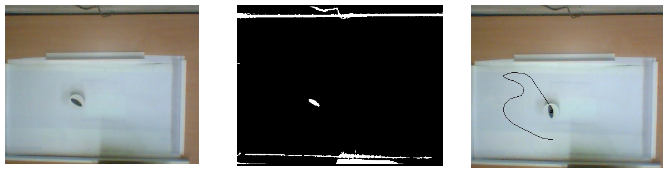


Fig. 8. (a) User's Joints Tracking (b) Physical Game Set Up: Kinect interface, E-puck Robot with black elliptical label, Overhead camera for visual tracking, Video beam for projecting virtual game in physical scenario.

Keeping track of the players joints (Fig.8) to capture the control commands, resulted in a process displaying a high stability, with a high performance, which was fairly independent of the light conditions in the room, the presence of other objects in the background of the scenario, and the rotation or translation of the user's body with respect to the Kinect. The interface stability was mainly due to the fact that we only used the data coming from the depth camera. The physical game setup is presented in Fig.8. By adding the tag to the robot we could make our tracking system more reliable (Fig.9). It provided the position feedback regardless of the robots speed even when other objects were present in the scene It is important to highlight that to increase the systems reliability it was also required to set the proper constraints on the blobs area and compactness, in addition to configuring the frequency at which the nearest neighbor tracking algorithm was executed.



(a) (b) (c)

Fig. 9. Overhead visual tracking system. (a) RGB Color Image. (b) Binary Image produced after thresholding. (c) Nearest Neighbor tracking algorithm

IV. Discussion

In this project we presented a bioinspired strategy of achieving a cooperative and coordinated behavior among a group of virtual agents, in which through local communication and the use of a set of three simple motion rules, naive micro behaviors at the individual level gave rise to an emergent complex behavior at the swarm's level. These complex interactions allowed the swarm to synchronize its actions and decisions to achieve different objectives. Furthermore, this project used evolutionary algorithms to ease the selection of the swarm parameters that cope with the different simulation objectives. Under the scope of this project the specific objective consisted in achieving a moderately compact swarm exhibiting a flocking behavior that could use a leader agent as the head of the swarm guiding its movement trajectory. Using a genetic representation which encoded the main control parameters of the swarm (simulated perceptions radii and rule pondering weights) along with a fitness function composed of four evolutionary pressures. We showed that the genetic algorithm was able to produce highly fit solutions which could find highly satisfactory behaviors, capable of balancing each rule's weight to obtain a fair tradeoff between a compact boid clustering, a reduction in the amount of inter agent collisions, and an adequate alignment among agents to promote the scenario exploration. In this way the controller managed to increase the swarm's probability to find and lock down its movement decisions towards following the leader agent.

The inclusion of the GA in the project, served as the initial stage of the co–design of a basic gaming scenario by automatically tuning the control parameters and helping the human game designer by generating a range of interesting and unexpected swarm behaviors. Some of which would have difficultly been achieved through the time consuming process of manually tuning parameters. In the second stage of the game design a human interaction was introduced in adding other elements to the game such as new characters (leader, predator, assistant and guider agent) and created new interactions and gameplays. In addition to the game design, we also enabled the games user to add a physical character which could be part of the game by interacting with some of the virtual characters. Such character consisted of an E-puck robot which was the physical representation of the leader character. The connection between the physical and virtual worlds was achieved by tracking the robots position inside a real arena. The position was obtained using an overhead camera which provided a live feedback to the virtual gaming scenario. The user could interact with the robot through gestures using his hands and shoulders, which were captured by a Natural User Interface (Kinect).

By using the GA as a character design tool, we left an open possibility that while searching for a strategically good behavior to achieve a specific objective, the GA could generate interesting and unexpected emergent behaviors [13]. By the simple fact that by an extremely large factor the search space of a GA is bigger to that of a human trial/error procedure. The GA could more easily generate boid controllers which exhibit certain sophisticated behaviors, from rapidly moving boids to naturally moving swarms with a more realistic aspect. An example of this can be seen in one of the individuals shown in Table II, in which the boids showed a duality, by always maintaining a single-group structure encompassing all of the casual agents (compactness), but at the same time allowing a great flexibility that enabled the boids to get significantly away from the center of the single-group (sparseness). This kind of behavior is really interesting since it achieves both, our desired flocking objective and at the same time a behavior that might be more appealing to a game user where the swarm members perform more dynamic movements, making the task of controlling and gathering the casual boids more difficult and thus interesting.

There are several future improvements that can be incorporated into the EvoBoids system in order to complete and enhance its capabilities. First, a more complex simulation and visualization environment can be created in order to not only model each of the agents and their intrinsic boids interactions, but also to incorporate interactions with the surrounding environment and ecosystem [14]. By adding these features one could achieve a spectrum of environmental conditions to which different types of agents could adapt their behaviors to, in this way, achieving a more detailed computational simulation resembling that of artificial life. Furthermore, an interesting approach would be to explore other evolutionary algorithms and use comparison techniques to evaluate the performance among algorithms. The work done in [15] presents more advanced evolutionary algorithms such as differential evolution (DE). DE could potentially incorporate several advantages towards optimizing the behavior of the swarm of agents because of its mechanisms of self-adaption of the control parameters, population size reduction, and mutation step size adaption. Moreover, DE is suitable for this problem since it has been demonstrated that it can be used for multi-objective optimization [16].

REFERENCES

[1] Thore Graepel, Ralf Herbrich and Adi Bot, "Video Games and Artificial Intelligence" [Online], Microsoft Research. Available: https://www.microsoft.com/en-us/research/project/video-games-and-artificial-intelligence/ [Accessed: May.29, 2016]

[2] J. Togelius, Julian and Shaker, Noor and Nelson, Mark J, "Procedural Content Generation in Games: A Textbook and an Overview of Current Research". Springer, 2016, pp. 01-04

[3] G. N. Yannakakis and J. Togelius (2015, Dec.), "A Panorama of Artificial and Computational Intelligence in Games," *IEEE Trans. on CIAIG*, 7 (4), pp. 317-335

[4] Melanie Mitchell, Stephanie Forrest et.al., "The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance", in *Proc 1st European Conf. on Artificial Life*, Cambridge, Aug 1, 1991

[5] J Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model", in *Proc. Computer Graphics Conf. SIGGRAPH*, 1987, pp 25-34.

[6] Ying Tan, Zhong-yang Zheng (2013, March). Research Advance in Swarm Robotics. *Defense Technology* [Online]. *9 (1),* pp. 18-39. Available: http://www.sciencedirect.com/science/article/pii/S221491471300024X

[7] Adam Birt, Samuel Shaw, "EvoBoids: Incorporating Machine Learning into Artificial Life", unpublished

[8] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," IEEE Symposium On CIG, 2008, pp 11-118.

[9] Cyberbotics, "e-puck robot" [Online]. Available: https://www.cyberbotics.com/item?id=8. [Accessed: June. 01, 2016]

[10] Stéphane Magnenat et.al., "Scripting the swarm: event-based control of microcontroller-based robots.", EPFL, unpublished

[11] Thomas Lochmatter et al. "SwisTrack - A Flexible Open Source Tracking Software for Multi-AgentSystems", EPFL, unpublished

[12] Eric Bonabeau, Marco Dorigo and GuyTheraulaz, "Nest Building and Self-Assembling" in *Swarm Intelligence*. New York: Oxford University Press, 1999, pp. 248-254

[13] M. Wagner, W. Cai and M. H. Lees, "Emergence by strategy: Flocking boids and their fitness in relation to model complexity," 2013 Winter Simulations Conference (WSC), Washington, DC, 2013, pp. 1479-1490.

[14] Aleš Zamuda, Janez Brest (2013, January), "Environmental framework to visualize emergent artificial forest ecosystems", *Information Sciences*, 220, pp. 522-540. Available: http://www.sciencedirect.com/science/article/pii/S0020025512005038

[15] Aleš Zamuda, José Daniel Hernández Sosa, Leonhard Adler (2016, May), "Constrained differential evolution optimization for underwater glider path planning in sub-mesoscale eddy sampling", *Applied Soft Computing*, 42, pp. 93-118. Available: http://www.sciencedirect.com/science/article/pii/S1568494616300266

[16] H.A. Abbass, R. Sarker, C. Newton, "PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems", in: *Proc. of the Congress on Evolutionary Computation*, Piscataway, New Jersey, 2001, pp. 971–978.