Motion Coordination in the Quake 3 Arena Environment: a Field-based Approach

Marco Mamei[1] and Franco Zambonelli[1]

Dipartimento di Scienze e Metodi dell'Ingegneria,
University of Modena and Reggio Emilia
Via Allegri 13, 42100 Reggio Emilia, Italy
{mamei.marco, franco.zambonelli}@unimo.it

Abstract. This paper focuses on the problem of orchestrating the movements of bot agents in the videogame Quake 3 Arena. Since the specific patterns of movement that one may wish to enforce may be various, and serve different purposes (have bots meet somewhere, move in formation, or surrounding human players), a general and flexible approach is required. In this paper we discuss how the Co-Fields coordination model can be effectively exploited to this purpose. The key idea in Co-Fields is to model the agents' environment by means of application-specific computational force fields, leading agents' activities to a globally coordinated and adaptive motion behavior. The Co-Fields model is described both in general terms and in the specific Quake 3 Arena implementation, and several application examples are presented to clarify it. Also, the paper outlines the general applicability of the approach besides the Quake scenario and in areas such as mobile computing and mobile robots.

1 Introduction

Quake 3 Arena (Q3A) [1] belongs to the kind of first-person shooter (FPS) computer games. The player controls a character (bot) fighting against other artificial bots (i.e., software agents). The most important tasks are staying alive and killing opponents. The game provides a first-person perspective on the current situation, see figures 4, 5, 7.

Bots in FPS have continually become more complex and more intelligent. The original bots were completely oblivious to their environment and used fixed scripts to attack the human player. Current bots, such as those found in Q3A and Unreal Tournament [2], are actually autonomous goal-oriented agents. They collect health and other power-ups, and they have a variety of tactics such as circle-strafing and popping in and out of doorways [3].

Although current bots perform very well as single players, the mechanisms to let them cooperate and coordinate each other activities (for example to surround an enemy) are still under-developed [4, 2].

The aim of this paper is to present an approach to the problem of coordinating the movements of a set of Q3A bots. The goals of bots' coordinated

movements can be various: letting them to meet somewhere, distribute themselves accordingly to specific spatial patterns, surround an enemy, or simply move in the environment without interfering with each other.

In our perspective, any type of coordination - there included in videogames - is built upon two main building blocks: *(i)* interaction mechanism *(ii)* context awareness. With regard to the former point, it is obvious that coordination requires some form of interaction (e.g. communication): bot agents need to communicate someway in order to decide/plan/synchronize their actions. With regard to the latter point a bot agent can meaningfully coordinate with other agents only if it is somehow aware of "what is around", i.e., its context (i.e. operational environment).

Starting from these considerations, we focus on the problem of dynamically providing bots with effective interaction mechanisms and with simple, easy to be obtained, and expressive contextual information.

To achieve our goal, we take inspiration from the physical world, i.e., from the way particles in our universe move and globally self-organize accordingly to that contextual information which is represented by potential fields. In particular, in our approach, the environment and contextual information are represented in the form of distributed computational fields (Co-Fields). Each agent of the system can generate and have propagated by the environment specific fields, conveying application-specific information about the local environment and/or about itself. Agents can locally perceive these fields and move accordingly, e.g. following the fields' gradient. The result is a globally coordinated and adaptive movement, achieved with very little efforts by agents.

The paper is organized as follows: Section 2 motivates and describes field-based coordination as realized by the Co-Fields model. Section 3 explains how the model has been implemented in Q3A. Section 4 presents some motion coordination examples in Q3A. Section 5 discusses how the ideas presented can be applied in a wide range of other scenarios and presents related work. Finally, Section 6 concludes.

2 The Co-Fields Approach

Let us consider the problem of letting a team of bots to meet somewhere in the Q3A dungeon. As anticipated in the introduction, to achieve this task, bots need some kind of interaction mechanism and some form of contextual information.

The mainstream solution for the above demands is to provide bots with a map of the dungeon and with some kind of communication channel that bots can use to agree on a specific location for the meeting. Although such solution may appear natural, it may require bots to execute complex algorithms to decide and negotiate where to meet and how to go there. This typically ends-up in brittle, static and not-adaptive solutions.

From a general perspective, the problem is that context-awareness is gathered by means of general purpose, not expressive and rather difficult to be processed description of the environment. The acquired information tends to be strongly

separated from its usage by the agents, typically forcing them to execute complex algorithms to elaborate, interpret and decide what to do with that information.

On the contrary, if the context would have been represented expressively to facilitate agents in the achievement of a specific task, agents would trivially use that information to decide what to do. For example, in the above meeting application, if the agents would be able to perceive in their environment something like a "red carpet" leading to the meeting room, it would be trivial for them to exploit the information: just walk on the red carpet!

So the point is: how can we create the "red carpet"? How can we effectively represent context for the sake of specific coordination problems?

An intriguing possibility is to take inspiration from the physical world, and in particular from the way masses and particles in our universe move and globally self-organize their movements accordingly to that local contextual information that is represented by gravitational and electro-magnetic fields. These fields are sorts of "red carpets": particles achieve their tasks simply by following the fields.

This idea is at the basis of field-based coordination models [5]. Following this approach, agents achieve their goals not because of their capabilities as single individuals, but because they are part of an auto-organized system that leads them to the goals achievement. Such characteristics also imply that the agents' activities are automatically adapted to the environmental dynamic, which is reflected in a changing field-based representation, without forcing agents to re-adapt themselves. More in detail, the Co-Fields approach is centered on a few key concepts:

1. Contextual information is represented by "computational fields", spread by agents and/or by the environment, diffused across the environment, and locally sensed by agents;
2. A motion coordination policy is realized by letting the agents move following the local shape of these fields, the same as a physical mass moves in accord to the locally sensed gravitational field;
3. Both environment dynamics and agents' movements may induce changes in the fields' surface, thus inducing a feedback cycle (point 2) that can be exploited to globally achieve global and adaptive motion coordination.

2.1 Computational Fields

A computational field is a distributed data structure characterized by a unique identifier, a location-dependent numeric value, and a propagation rule identifying how the field should distribute in the network and how its value should change during the distribution. Fields can be static or dynamic, basically a field is static if once propagated its magnitude does not change over time; it is dynamic if its magnitude does. A field can be dynamic because for example its source moves and the field, with some propagation delay, changes accordingly its magnitude, or because for example its propagation rule its designed to remain active and to change field value after some time. Fields are locally accessible by agents depending on their location, providing them a local perspective of the

global situation of the system. For instance, with reference to the case study, a Quake bot, call it the "prey" can spread in the map network infrastructure a computational field (let's call it PRESENCE field) whose value monotonically increases as it gets farther from the bot. Such field implicitly enables any other bots; call them the "predators", from wherever in the dungeon, of sensing the presence of the "prey" and its distance. Also, by sensing the local gradient of the PRESENCE field, the "predators" could also know in which direction the "prey" can be found (see figure 1).

2.2 Motion Coordination

In Co-Fields, the simple principle to enforce motion coordination is to have agents move following the local shape of specific fields. For instance, a "predator" bot looking for a "prey" bot can simply follow downhill the corresponding PRESENCE field. Dynamic changes in the environment and agents' movements induce changes in the fields' surface, producing a feedback cycle that consequently influences agents' movement. For instance, should the "prey" bot be moving around in the dungeon, the associated PRESENCE field would automatically update its shape to reflect the new situation. Consequently, any agent/bot looking for a "prey" would automatically re-adapt its movement accordingly. Should there be multiple "prey" bots, they could decide to sense each other's PRESENCE fields so as to stay as far as possible from each other to better escape from "predators".

In general, a Co-Fields based system can be considered as a simple dynamical system and can be effectively modeled as that (see [6] for details on Co-Fields modeling): agents are seen as balls rolling upon a surface, and complex adaptive movements are achieved not because of the agents wills and skills, but because of dynamic re-shaping of this surface. Of course, such a physical inspiration and the strictly local perspective in which agents act promote a strictly greedy approach in agents' movement: agents act on the basis of their local viewpoint, disregarding if a small sacrifice now (i.e., climbing a Co-Fields hill instead descending it) can possibly lead to greater advantages in the future. In a circular track for example, a "predator" looking for a "prey" that is moving clockwise, instead of greedily follow downhill the PRESENCE field (as the Co-Fields approach promotes), could better decide to move uphill to meet the "prey" counterclockwise. However, this is a general drawback of distributed problem solving, where efficiency reasons often rule out the possibility of globally informed decisions by distributed agents.

2.3 Application-Specific Coordination

The achievement of an application-specific coordination task is rarely relying on the evaluation, as it is, of an existing computational field (as in the case of a "predator" looking for a "prey" and simply following the specific PRESENCE field of that "prey". Rather, in most cases, an application-specific task relies on the evaluation of an application-specific coordination field, as a combination of some of the locally perceived fields. The coordination field is a new field in itself, and it is built with the goal of encoding in its shape the agent's coordination

task. Once a proper coordination field is computed, agents can achieve their coordination task by simply following (deterministically or with some probabilistic rule) the shape of their coordination field uphill or downhill (depending on the specific problem) as if they were walking upon the coordination field associated surface (see figure 1). For instance, in the case study, for "prey" bots to stay as far as possible from each other, they can follow uphill a coordination field *CF* resulting from the combination (the sum) of all the computational fields of each "prey":

$$CF = \sum_{i=1}^{n} PRESENCE_i$$

At the moment, we still have not identified a general methodology to help us identify, given a specific motion pattern to be enforced, which fields have to be defined, how they should be propagated, and how they should be combined in a coordination field. We are confident some methodology to help in that direction can be found in the future, and would possibly make Co-Fields applicable to a wider class of distributed coordination problems even beyond motion coordination. Nevertheless, the immediate applicability of Co-Fields is guaranteed by two important considerations:

1. It is possible to get inspiration and of reverse engineer a wide variety of motion patterns found in Nature. For example, swarm intelligence phenomena [7, 8] such as bird flocking, ant foraging and bee dances provide a variety of useful motion patterns and can all be modeled with Co-Fields [9].
2. Complex motion patterns can be divided into simpler movements (e.g. moving along a square can be divided into following sequentially the four lines composing the square edges). Thus, although it might be impossible to perform complex motion patterns by following only one field, such complex motions patterns could be easily achieved by following sequentially a number of different fields.

3 Implementation

As anticipated in the introduction, to test the validity of our approach, we implemented the Co-Fields model within the videogame Quake 3 Arena (Q3A).

There are mainly two options to exploit Q3A as an environment simulation for multi-agent system:

1. write an *external* program (client-bot) that connects to the game as a human player. This bot receives (via a suitable interface) a world representation (roughly similar to the 3D scene view a human player can see) and can perform actions by sending commands to the game engine (roughly similar to joystick commands). With this approach the goal is to create a bot that actually plays the game as a human player [10, 3].

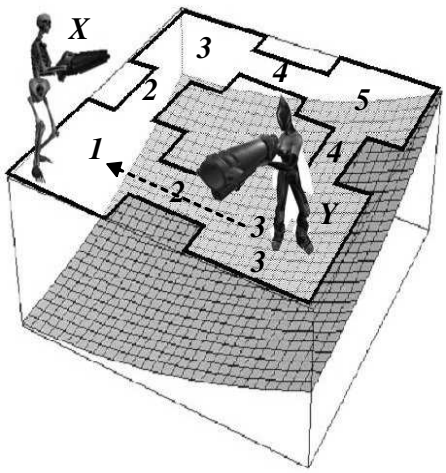


Fig. 1. Agent X propagates a PRESENCE field whose value has the minimum where Agent X is located. Agent Y senses that PRESENCE field to approach agent X by following downhill the PRESENCE field's gradient

2. write an *internal* modification (*mod*) of the game. A mod is a package of changes made to the game altering the way in which the game was designed to work. A mod can involve something as simple as changing the speed at which a rocket moves across a room or as complex as a complete overhaul of the look and feel and even the rules of the game [11].

In our work we ventured with the latter approach for a very specific reason. Since fields must be spread across the whole Q3A world, it appears difficult to realize such functionality with a client-bot that is functionally equivalent to a human player. Some kind of deeper modification appears to be required.

From a general perspective, this fact can be regarded as an example of the importance of explicitly modeling the environment in a multi-agent system. In fact, the power of the Co-Fields approach derives primarily from the fact that the field-based representation of the environment provides agents with handy information to achieve a specific task. This frees the agents from the burden of mutually interacting to acquire suitable context information.

3.1 Quake 3 Arena Internals

Q3A has been designed as two separate interlocking components: the 3D engine (proprietary) and the game logic (open-source). Mods are created modifying the open-source part of Q3A and recompiling it with the rest of the code. In this way, a mod programmer can disregard all the low-level details about the 3D display and concentrate on high-level topics like bots' behavior and game logic.

Before explaining how Q3A has been modified to implement Co-Fields, it is fundamental to give some remarks on the bots general architecture. A bot is build with a layered structure: the first layer manages the bot I/O operations, the second one manages the execution of complex actions (e.g. walking to a room, picking-up a weapon, etc.), the third layer is basically a finite state automaton governing high-level decisions (e.g. when to attack, when to withdraw, etc.). Eventually, the fourth one models the behavior of the commander when a squad of bots is involved (see figure 2). Q3A works as a time-based simulation. Each bot is given a 1/10 sec time-frame in which to run its code.

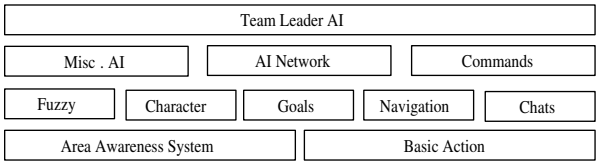


Fig. 2. bot layered architecture.

Although the purpose of the bot layered architecture is to mask to the programmer the intricacies of dealing with low-level details and to focus only on high-level artificial intelligence issues, for the purpose of implementing Co-Fields, there are two low-level parts worth to be considered: the Area Awareness System (AAS) and the bot's Goals.

The AAS is the component in charge of providing the bot with a representation of the Q3A world. Specifically, a bot perceives the world as a set of not-overlapping, adjacent, convex areas. These areas are mainly used for navigation purposes; routes in the Q3A world are basically a sequence of areas to be traversed.

A bot is a goal-oriented agent, whose execution is mainly intended to pursue some goals. There are either long-term goals (e.g. look for an enemy) and nearby goals (e.g. pick a power-up while looking for an enemy). The fundamental point to understand is that when, modifying the bots' behavior, we tend to remain on the high levels of its architecture. This means that bots are not controlled by means of an algorithm prescribing something low-level like (e.g. walk north 10m, then turn left, then fire). But with something high-level and goal-oriented as (e.g. look for heavy-weapons then hunt for enemies, until not injured).

3.2 Implementing Co-Fields

To implement Co-Fields one has basically to: *(i)* have fields spread in the Q3A world, *(ii)* change the way in which bots perceive their environment. Bots must be provided with a field-based representation of the environment, *(iii)* the goal of moving following fields' gradients must be introduced.

In our implementation, each bot propagates only one field (such a fixed number is imposed by Q3A that does not allow dynamic memory allocation). Each field is stored in an array having a number of cells equals to the number of areas composing the Q3A world. Each cell stores the value of a field in a specific area. In our implementation such a value is always a function of the distance between the area considered and the area in which the bot (source) is located. At every time-frame of the game, all the arrays are updated to reflect the current system situation (see code in figure 3).

In standard (i.e. not modified) Q3A, bots perceive the world by means of a set of areas as provided by the AAS. The AAS provides various information on the kind and content of the areas (e.g. the area is full of water, contains a power-up, etc.). In our modifications, we enriched such a model, to provide bots with also the information on the kind of fields being spread in the areas and their magnitudes.

Each bot is also associated with an array representing its coordination field. Such an array is evaluated by composing the above described fields' arrays. Specifically, each bot selects the area where to move by looking, in its coordination field array, for neighbor areas where the gradient goes e.g. downhill. Then, the bot is committed to the long term goal LTG_PATROL to the selected area. This brings the bot to go (and patrol) the selected area. The looping of these actions at each time-frame, lets the bot follow downhill its coordination field.

```
int client;
// these are pointers to some areas in Q3A world
aas_areainfo_t infoarea; aas_entityinfo_t ent_field;
// with the following command ent_field points
// to the area occupied by bot client
BotEntityInfo(client, &ent_field);
for (i=1;i<=numareas;i++)
{
 // The following command, within the for-cycle
 // lets infoarea points to every area in the world
 trap_AAS_AreaInfo( i, &infoarea);
 // The field value is equal to the distance between the
 // bot and the area. It is thus a field increasing its
 // magnitude as it gets farther from the source.
 dist = Distance(ent_field.origin,infoarea.center);
 FIELD[client][i]=dist;
}
```

Fig. 3. fragment of the Q3A modified file "ai_main.c".A PRESENCE field is spread in Q3A world.

4 Examples of the use of Co-Fields

To clarify the usage of Co-Fields in Q3A, we detail here some specific motion coordination problems that bots may be in need to face.

4.1 Exploiting the PRESENCE Fields: Meetings and Predations

We have previously shown how the PRESENCE field can be exploited to detect where a bot is located in the Q3A map and how it can be reached. Moreover, the PRESENCE field can be exploited to enforce a variety of other interesting motion patterns.

As a first example, let's consider a "meeting" application to help a team of bots to dynamically meet with each other. Although different policies can be enacted to let a team of bots to meet somewhere (e.g., a specific point or by a specific bot), here we concentrate on having the bots collaboratively walk towards each other and eventually meet in some dynamically determined intermediate point. If each member i of the group generate a PRESENCEi field, then each bot i of them can evaluate its coordination field by taking the maximum PRESENCE field of all the other bots:

$$CF_i = max(PRESENCE_{j \neq i}) j = 1, 2, ..., n$$

and then follow such coordination field downhill. While the coordination fields continuously change due to the concurrent movements of all the members of the team each following its own coordination fields, the result is that the bots gradually approach towards each other, until collapsing in a single point. In other words, because agents actually attract each other, the system naturally converge to the situation in which all the agents are in the same point.

Figure 4 shows a snapshot of Q3A with bots involved in the meeting process. The key point to emphasize is that the meeting process, and Co-Fields in general, is adaptive: it works well even independently of the characteristics of the environment (e.g., of the map), without having to change a bit in the code of agents or in the structure and propagation of computational fields.

As another example exploiting the PRESENCE fields, consider a team of bots ("predators") moving in an environment with the goal of surrounding and kill the human player ("prey"). By getting inspiration from the behavior of wolves, which succeed in collaboratively surround a prey using the simple strategy of approaching the prey while trying to stay as far as possible from each other, we can define the coordination field of a generic predator i in this way:

$$CF_i^{pred} = PRESENCE^{prey} - \sum_{j=1, j \neq i}^{n} PRESENCE_j^{pred}$$

expressing that the predators follow downhill the PRESENCE field of the prey (to reach it) and, at the same time, tries to stay far from all other predators. The result (see figure 5) is that predators, rather than simply approaching the prey, are able to surround it.

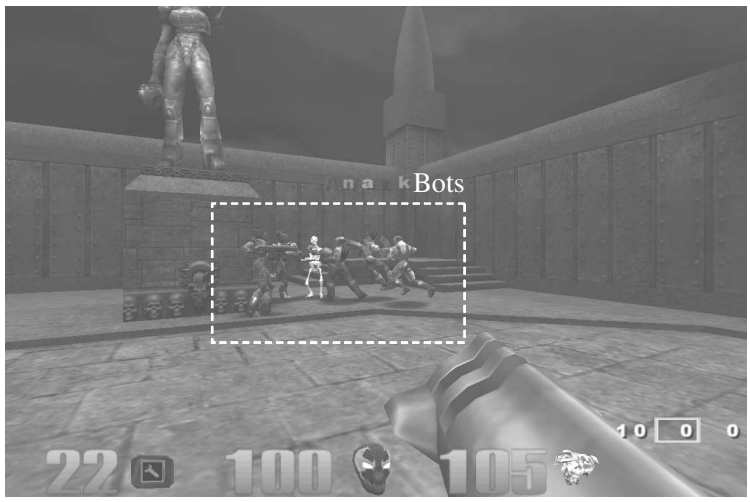


Fig. 4. A snapshot of the meeting process in Quake 3 Arena. All the bots have met.

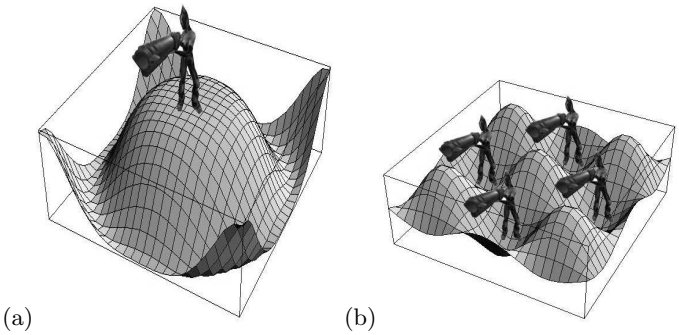
Fig. 5. snapshot of the surrounding process in Quake 3 Arena. The bots surround the player by closing all the escape doorways (remember that the game offers a first person view).

4.2 Flocking

As another example, let us consider the problem of having bots move in the Q3A map by maintaining an equal distance to each other, so as to form a regular formation. In an ideal, continuous case, a field that could be used to realize that purpose could be the one in Figure 6(a). Let us call this field FLOCK, to correctly attribute the fact that algorithms for moving in a formation has been inspired by the movements of birds' flocks [8]. Each agent has to generate a FLOCK field with a minimum at a distance a from itself, where a is the distance that must be preserved between agents to maintain the formation, and continuously increasing for higher distances. Then, to guarantee that each agents stay at a distance a from any other agents, the FLOCK fields generated by all the agents in the environment (say, a total of n) must be composed in the following coordination field:

$$CF(x, y, t) = min(FLOCK_i(x, y, t) : i = 1, 2, ..., n)$$

where is the FLOCK field of the agent i, and CF is the coordination field. to be followed downhill by agents. In other words, each agent i in the system can evaluate its coordination field CF as a minimum combination of all the other agents' fields and simply follow downhill the coordination field obtained. Globally, the system self-organizes in an almost regular grid, because all the agents try to remain in one of the minimum points of the next agents' fields and thus they tend to maintain a regular distance of a (see figure 6(b) and figure 7).



(a) (b)

Fig. 6. (a) Single flock field (b) Global composition of the flock fields: bots maintain an almost regular grid formation.

5 Other Application Scenarios And Related Work

Other than videogames, field-based coordination and the Co-Fields model well suit a wide array of application scenarios: from other videogames and movies to pervasive computing, robotics and material self-assembly.

Fig. 7. snapshot of the flocking process in Quake 3 Arena. Bots moving maintaining regular distances from each other.

5.1 Co-Fields In Other Scenarios

The problem of coordinating the movements of a group agents has lots of interesting applications in several scenarios. Let us consider the exemplary pervasive computing problem of tourists visiting a museum assisted by personal digital assistants. Specifically, we can focus on how tourists can be supported in coordinating their movements with other, possible unknown, tourists. Such coordination activities may include scheduling attendance at specific exhibitions, having a group of students split in the museum according to teacher-specific laws, helping a tourist to avoid crowd or queues, letting a group of tourist to meet together at a suitable location, and even helping to escape accordingly to specific emergency evacuation plans.

To apply the Co-Fields model to this scenario, we need a computer infrastructure suitable in supporting fields propagation across the museum. Specifically, we can suppose that the museum is provided with an adequate embedded computer network. In particular, embedded in the museum walls (either associated to each artistic items or to each museum room), there will be a network of computer hosts, each capable of communicating with each other and with the mobile devices located in its proximity via the use of a short-range wireless link. The number of the embedded hosts and the topology of the network may depend on the museum, but the basic requirement is that the network topology mimics the topology of the museum plan (i.e. no network links between physical barriers, like walls). On each host there will be a middleware providing basic support for data storing (to store field values) and communication mechanisms (to propagate fields) [6]. Moreover, we can suppose that tourists are provided with a

software agent running on some wireless handheld device, like a palm computer or a cellular phone, in charge of giving her/him suggestions on where to move.

Devices connect to nearby embedded hosts (to this end some kind of localization mechanism has to be enforced [12]). They can inject fields across the network and read field values in the neighborhood to enforce field-based coordination (e.g. move to the room where the gradient of a field goes downhill). With this regard, it is worth noting that the critical assumption made about the museum network topology is about not having people stumbling at walls while following gradients. More details on this Co-Fields application can be found in [13].

The above scenario and the associated motion coordination problems are of a very general nature, being isomorphic to a lot of scenarios ranging from other pervasive computing applications (such as, e.g., traffic management and forklifts activity in a warehouse, where navigators' equipped vehicles hint their pilots on where to go), to Internet-scale scenarios (e.g. in software agents exploring the Web, where mobile software agents coordinate distributed researches by moving on various Web-sites). Therefore, our Co-Fields model can be applied also in much diverse areas than the one considered [14, 13, 6].

5.2 Related Work

Several proposals, in the last few years, address the problem of supporting agents' activities with coordination approaches similar to Co-Fields.

In the videogame domain, one of the most remarkable examples is represented by the popular videogame "The Sims" [15]. "The Sims" exploits sorts of computational fields, called "happiness landscapes" and spread in the virtual city in which characters live, to drive the movements of non-player characters [16]. For instance, if a character is hungry, it perceives and follows a happiness landscape whose peaks correspond to places where food can be found, i.e., a fridge. After having eaten, a new landscape will be followed by the character depending on its needs. Although sharing the same inspiration, "Sims' happiness fields" are static and generated only by the environment. In Co-Fields, instead, fields are dynamic and can change over time, and agents themselves are able to generate fields to promote a stronger self-organization perspective.

Remaining in the entertainment domain, it is worth reporting that autonomous agents, coordinating their movements and their actions, have been employed in the recent movie "The Lord of the Rings, The Two Towers". In the Helm's deep battle, to enhance the scene realism, the 50000 fighting characters have been modeled by means of goal-oriented autonomous agents, developed within the Massive framework [17]. In this approach agents interact with each other on a strict local basis, without any long-range interactions. In our opinion, also this kind of approach could take advantage of integrating long-range, mediated interactions like those enabled by fields. These, in fact, would allow simulating large-scale tactics, like a global flanking or a global surrounding.

Also outside the entertainment domain, similar approaches can be conveniently used. Several projects in the last few years have worked to facilitate

distributed-motion coordination. In robotics, the idea of potential fields driving robotic movement is not new [5]. For instance, one of the most recent manifestations of this idea, the Electric Field Approach [18] was used to control a team of Sony Aibo legged robots in the RoboCup domain. Following the EFA approach, each Aibo robot builds a field-based representation of the environment from the images captured by its head-mounted camera and decides its movements by examining the fields' gradients of this representation. Although close in spirit, EFA and Co-Fields differ from the implementation point of view. In Co-Fields, fields are distributed data structures actually spread in the environment; in EFA, fields are just an agent's internal representation of the environment and do not actually exist. Co-Fields require a supporting infrastructure to host field data structures, but they completely avoid the complex algorithms involved in field representation and construction.

Field-based approaches are taking over also in futuristic and fascinating scenarios such as self-assembly of agent-based micro-particles. One of the most successful approaches in this scenario has been proposed within the amorphous computing research [19]. The particles constituting an amorphous computer have the basic capabilities of propagating sorts of abstract computational fields in the network, and to sense and react to such fields. In particular, particles can transfer an activity state towards directions described by fields' gradients, so as to make coordinated patterns of activities emerge in the system independently of the specific structure of the network. Such mechanism can be used, among other possibilities, to drive particles' movements and let the amorphous computer self-assemble in a specific shape. Although conceived for a very specific scenario, this approach shares with Co-Fields the idea of having a single, physically inspired, mechanism to both diffuse contextual information and to organize adaptive motion coordination patterns. Finally, we want to emphasize again that, in our opinion [9], a lot of related work form the swarm intelligence research [7], there included flocks of birds, schools of fishes and packs of wolves [8], can all be modeled in terms of Co-Fields.

6 Conclusions and Future Work

While both a preliminary prototype implementation and the outcomes of our simulation shows the feasibility of the approach, a number of research directions are still open to improve its generality and its practical applicability. In addition to the already identified need for general methodologies to help designing specific coordination patterns in terms of Co-Fields, it will be important to explore the potential of Co-Fields in encoding more general distributed coordination patterns, possibly not related to motion.

7 Acknowledgements

Work supported by the Italian MIUR and CNR in the "Progetto Strategico IS-MANET, Infrastructures for Mobile ad-hoc Networks".

References

1. Quake 3 Arena: (http://www.idsoftware.com/games/quake/quake3-arena)
2. Unreal Tournament: (http://www.unrealtournament.com)
3. Liard, J.: It knows what you're going to do: Adding anticipation to a quakebot. In: Intenational Conference on Autonomous Agents, Montreal, Canada (2001)
4. Quake 3 Team Arena: (http://www.idsoftware.com/games/quake/quake3-teamarena)
5. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research **5** (1986) 90 – 98
6. Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications with the tota middleware. In: IEEE International Conference On Pervasive Computing (Percom). IEEE CS Press, Orlando (FL), USA (2004)
7. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence. From Natural to Artificial Systems. Oxford University Press, Oxford (UK) (1999)
8. Parunak, H.V.: Go to the ant: Engineering principles from natural multi-agent systems. Annals of Operations Research **75** (1997) 69 – 101
9. Mamei, M., Leonardi, L., Zambonelli, F.: Co-fields: A unifying approach to swarm intelligence. In: Engineering Societies in the Agents World III: Third International Workshop, ESAW 2002. LNAI. Springer-Verlag, (Berlin, DE) 68 – 81
10. J. Liard, J.D.: Creating human-like synthetic characters with multiple skill levels: A case study using the soar quakebot. In: AAAI 2000 Fall Symposium Series: Simulating Human Agents. (2000)
11. Holmes, S.: Focus on MOD programming in Quake 3 Arena. Premier Press (2002)
12. Hightower, J., Borriello, G.: Location systems for ubiquitous computing. IEEE Computer **34** (2001) 57 – 66
13. Mamei, M., Zambonelli, F., Leonardi, L.: Co-fields: A physically inspired approach to distributed motion coordination. IEEE Pervasive Computing **3** (2004) 52 – 61
14. Mamei, M., Zambonelli, F., Leonardi, L.: Distributed motion coordination with co-fields: A case study in urban traffic management. In: 6th IEEE Symposium on Autonomous Decentralized Systems. IEEE CS Press, Pisa, Italy (2003) 63 – 70
15. The Sims: (http://thesims.ea.com)
16. Johnson, S.: Wild things. Wired **10** (2002)
17. Koeppel, D.: Massive attack. Popular Science (2002) 38 – 44
18. Johansson, S., Saffiotti, A.: Using the electric field approach in the RoboCup domain. In: RoboCup 2001: Robot Soccer World Cup V. LNAI. Springer-Verlag, (Berlin, DE) 399–404
19. Nagpal, R.: Programmable self-assembly using biologically-inspired multiagent control. In: Proceedings of the 1st Joint Conference on Autonomous Agents and Multi-Agent Systems. ACM Press, Bologna, Italy (2002) 418 – 425