A Design Approach to Decentralized Interactive Environments

Harm van Essen, Pepijn Rijnbout, and Mark de Graaf

Department of Industrial Design, Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{h.a.v.essen,p.rijnbout,m.j.d.graaf}@tue.nl

Abstract. We are exploring a design approach to the implementation of decentralized intelligent environments. We adopt the research through design process by creating an infrastructure of physical, interactive objects and explore the potential of a decentralized philosophy in four design iterations. Open-ended play serves as a fruitful context for design cases. Iterations of prototyping and user testing facilitate the exploration of emergence. One of the design outcomes is a simple decentralized system for soccer training which proved to be very successful on challenge and motivation, and inspired players to invent a range of games, both competitive and cooperative.

Keywords: open-ended play, decentralized systems, research-through-design, ubiquitous computing.

1 Introduction

In open-ended play, the goal of a game is not specified in detail. Tools and materials for open-ended play leave room for imagination, meaning giving, exploration and experimentation. A traditional example is construction material like Lego. SPORE [15] is a recent computer game that facilitates open-ended gaming. Open-endedness is an issue in designing intelligent play objects [1, 17]. If in a given physical context people start interacting with a number of such objects, the whole of objects and users forms an intelligent system, and the domain of open-ended play becomes connected to the domain of ubiquitous computing. This is the type of intelligent environments this paper is about.

The field of ubiquitous computing or ubicomp aims at a technology saturated future, in which the numerous computers that have invaded the environment we live in, are woven into the fabric of everyday life. They have become invisible to the users, they become connected in a ubiquitous network and will profoundly change the way we live [18]. In this paper, we are exploring the design of *decentralized* interactive environments. The potential of decentralized solutions has been explored successfully for a wide variety of contexts, such as sensor networks [10], P2P web crawlers [14], and robot swarms [13]. In these applications, the role of the user interacting with the objects is very limited. In the work presented here, the users are considered to be active, influential agents in the system.

A. Nijholt, D. Reidsma, and H. Hondorp (Eds.): INTETAIN 2009, LNICST 9, pp. 56–67, 2009.

In our view, entertainment and play is an excellent test ground for ubicomp research. It is often considered to be a good metaphor for "real" life: there are people with motivations and intentions, interactions, rules, and there is a physical environment. Compared to real life, all these aspects are strongly simplified. This is a favorable condition for the exploration of fundamental aspects of ubiquitous computing. Ubiquitous computing also offers excellent perspectives for open-ended play. In the dialogue with the environment, players can give new meaning to interactions taking place in the game, and new forms of play can be invented.

As a working method, we adopt the research through design process. In our interpretation, this encompasses an iterative approach based on extensive experiments with hardware objects interacting with real users. By designing objects and analyzing the interaction with users we investigate fundamental aspects of decentralized systems. Observing users' behavior and experience in different settings aids to an understanding how decentralized systems could be designed for applications in intelligent environments. Our final goal is to derive design heuristics.

Novel in this work is the combination of the important role of the users in the system, the decentralized implementation, and the design perspective. This paper aims to explore the potential of this focus on several levels. First, does the design approach work for decentralized interactive environments? Second, does it open new directions for interactive open-ended play? And finally, can the lessons learned be generalized to design heuristics for similar decentralized environments for open-ended play?

In the following section, we will go deeper into the roots and potential of the decentralized design method. Next, we will discuss the research through design approach method in more detail. Finally we will report and discuss three completed iterations of this process and formulate further steps.

2 Decentralized Systems

Decentralized systems consists of a number of (identical or different) components (objects, devices, or agents) that can have mutual interactions (i.e. communicate, coordinate, negotiate) with each other and with their common environment, according to basic (local) interaction rules. Grouped together these components form a system; this overall system demonstrates a global functionality (or reaches a goal) that single components are never able to accomplish. This behavior emerges from interaction dynamics and is called emergent behavior. Emergent behavior of a system is not explicitly described by the behavior/design of the individual components, and these components do not have knowledge about desired system behavior. The global behavior is not predictable as the sum or a function of the local behavior.

Well known and inspirational examples of Emergent Behavior are found in nature (ants [4], bee hive [12], flocks of birds and schools of fish [11], but also heart beat and brain rhythm [2]) but also in large cultural systems (traffic [7], economy [9], etc). In recent years, the principles of these multi agent systems are better understood. Many researchers like Steven Strogatz [16] and John Holland [8] have proved underlying mechanisms like self organization, clustering and coordination. For emergent behavior to occur, a relatively large number of more or less identical agents, gradient fields (like accumulation, pheromone traces etc.), multiple and intensive interactions, feedback loops, and finally, some amount of randomness are required.

A large number of interactions can be achieved by many agents having a few interactions, or few agents having many interactions, or anything in between. Ubicomp typically focuses on environments like a home or an office. The number of agents will be limited and the diversity will be high. In terms of a decentralized system this is characterized as small and heterogeneous, as opposed to the large and homogeneous systems known from nature. The low number of agents and the heterogeneity hampers the intensity of interactions. If such a system had to be self sustaining, there would be little chance of successful emergence. However, the active presence of (human) users makes the essential difference here: our decentralized system is sustained by user interactions.

The user input is a necessary condition to achieve emergence. The user introduces "order" into the system by his decisions and strategies. In a sense the decentralized systems that we strive for are no longer "closed", autonomous systems, and different conditions for the emergence of interesting behavior might be applicable. Moreover, the perception and experience of the users are more important than autonomy, stability, or extinction of interesting behavior; it is the user who drives the system. This opens perspectives for a designers approach.

Methodologies for decentralized systems' design have been developed [5, 6]. The designer of a system is interested in designing the global behavior of a system, in order to reach an overall design goal. For a decentralized system, the real challenge is how to define the (local) interaction/communication behavior ("the rules") between separate objects such that, when taken as a whole, they self-organize and demonstrate the desired global behavior.

When human users are involved, also the interaction between the users and the system, and the interaction between users needs to be designed. An interesting question is what the elementary requirements on the interactions rules of the agents are. A first, undisputable, requirement is that for determining the values of the output of a particular agent only information is used that is available within that agent at that time. This information can be based on (previous received) inputs or on autonomous change of internal variables. The update can be static, or dynamic, i.e. based on a history of previous input and conditions. Another requirement could be that that agents are not allowed to give direct instructions related to the output of other agents (in a hierarchical order). These basic requirements rule out that individual agents have a "global" overview of the status of other agents or the overall system (like a central database), and they also avoid properness errors, i.e. using information that is not available yet.

The expected advantages of a decentralized system are in:

- transparency (both from a user as well as from a designers' point of view, simple objects with limited functionality, simple local interaction rules instead of complex overall programs),
- scalability (add or remove objects without the need for reconfiguration),
- adaptivity (self organizing systems adapt easily to changing environments and different modes of use, also if they have not been foreseen by the designers),
- robustness (malfunction of one agent will not have large effects on the system behavior).

3 Research-through-Design Approach

Decentralized systems can and are explored from many different perspectives. These range from strictly mathematical, i.e. studying the dynamic behavior of coupled differential equations [16], via computational, e.g. verifying design goals by extensive simulations (e.g. traffic management [3]), to a merely observational approach, i.e. analyzing and mimicrying (biological) mechanisms and phenomena [4, 12]. Borrowing from all those disciplines, we choose to approach this problem from a design point of view.

Design refers to the process of creating and developing new objects or services. Designing requires considering aesthetic, usability, functional, and contextual aspects of a system or service. Working in a realistic context, relevant design goals and a specific user group needs to be formulated and evaluated. User needs and user capabilities are central in our view on design. Design research typically produces heuristics rather than mathematical relationships.

In our opinion, the research-through-design method is appropriate for the challenge at hand here. Research through design implies an explorative and iterative approach. Short cycles of design and reflection on design are used to generate research output. Reflection is driving the iterations. To investigate the (effect) of interaction, real objects are needed in a realistic use context. Concretely, in this project sets of physical agents are built and used in several consecutive design cases.

4 Design of the Objects

Before starting the actual design iterations we need an infrastructure of physical, communicating and interactive objects to work with. A set of requirements has been formulated, based on our vision on designing for interactive decentralized systems presented above.

The objects should be autonomous, it should be possible to spatially distribute the objects over a space, and the objects should be able to exhibit real interaction with each other and with human users. This means that agents have to be able to receive input from other agents and to communicate output with other agents, and that agents have to be able to receive input from users and to communicate output which is perceptible for users. The agents should be able to autonomously compute their current output based on (past) inputs and interaction rules.

Because we intend to explore the system in different contexts, the agents need to be easy to handle and have a neutral design. It should be possible to change the behavior of the individual agents by changing the interaction rules. Agents should have a flexible, easily adaptable functionality, while the interaction modalities should be abstract (for instance colored LED lighting instead of signals that are meaningful in a single context). Heterogeneity can be implemented by the flexible functionality: Objects could be “sensing” input in a subset of modalities, while being “active” in another subset of modalities.

Twenty agents have been built. This number of agents is a compromise between effort of building, interaction opportunities, assumed probability of emergent behavior, and perception of users. The number fits well in the range of small number decentralized

systems. The intensity of mutual interaction is of course influenced by software. In the perception of users, a set of 10 to 15 agents (when placed close enough together), is already experienced as 'a group' of agents. Interacting with a group, it is more likely that users perceive group output instead of output from individual objects. This perception obviously fits better the principles of emergent behavior.

As casing, standard, low cost containers (12x8x8 cm) are used in which a Plexiglas frame is fitted on which all components are mounted. All parts are shielded and protected by the box, including the battery pack. A robust on/off switch is positioned on the bottom of the agent. All agents are identical and the design is neutral. Power supply of the agents is arranged with batteries, which allows flexible positioning of the agents. As low battery voltage causes unpredictable behavior, a battery voltage monitor indicates if the voltage drops below a critical value.

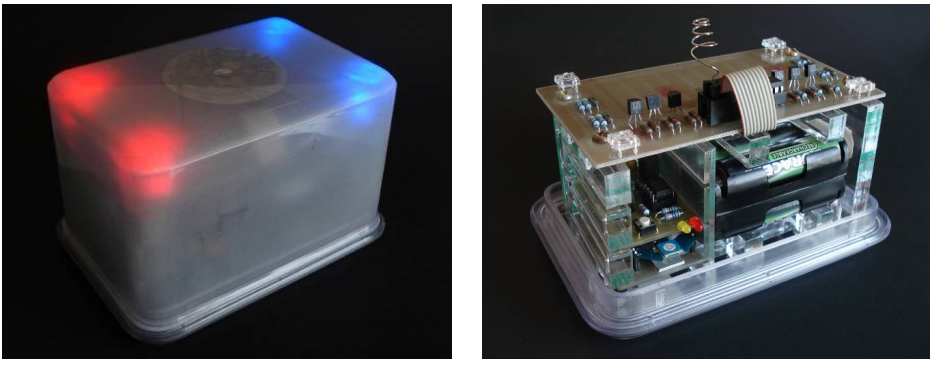

Fig. 1. Pictures of an agent, (left) closed with indicated LED lights and opened

In every agent the following modules are implemented on separate printed circuit boards: microcontroller module, input/output module, XBee module, battery voltage monitor, and a battery pack. The agents are equipped with a standard PIC microcontroller board. In most cases the interaction rules are simple and the required amounts of computing power and memory needed are limited. The opportunities for interaction with the user are implemented on a separate I/O PCB (which can be adapted to different contexts). The agents are able to give output which is perceptible for users. This is implemented with two sets of RGB LEDs. As different colors or patterns can have different meaning, this symbolic output provides sufficient output possibilities. Although the agents are physically identical, different input/output modes can simulate heterogeneity. The agents are able to receive input from users. A straightforward solution of one touch sensor (one bit output) on top of the object has been selected.

Interaction between agents is implemented with XBee wireless RF communication modules. The ZigBee protocol supports a peer-to-peer network configuration. When the module is configured correctly, communication is arranged by the module without interference of the microcontroller. In experiments with intensive data communication, occasional data loss was encountered. Although data loss can be part of a decentralized

system, it can be avoided by including a handshake procedure. Also more advanced communication protocols can be implemented.

As a matter of fact, the input of an agent is the output of another agent. In our case, agents should have light and color sensitive sensors. Instead of real sensors, the RF communication emulates input of agents, agents are connected by communication. As an example, when a red LED of a particular agent shines, this agent communicates that it is emitting red light to all other agents. Whether an agent is able to receive this signal is determined in software. An advantage of an emulated sensor is that the communication can be programmed freely; there is no measurement noise or problems with line of sight. It provides extra flexibility, because no sensors or different types of sensor are needed.

The microcontroller and communication software are written in C. In a discrete time frame, the output of the agent for the next sample time is computed according to the local interaction (or input/output) rules. Different methods are available for programming the local interaction rules of the agents: It is possible to program dedicated algorithms for learning or pattern recognition in the agents; it is also possible to select a generic structure for the software which can (easily) be adapted to different cases.

5 Research-through-Design: Case Studies

The above described abstract interactive objects were the starting point for a still ongoing iterative research-through-design process. The design cases represent simplified implementations of decentralized systems in a specific context. The first case, the coming home ritual, was chosen as a typical example of a repetitive, rather stable pattern in a relevant ubicomp context, home. Consecutive design cases were formulated based on the evaluation of the proceeding ones. This is the nature of the research-through-design process. The first three iterations have been finished and are concisely presented. The next planned iteration is described. As we found out with interactive simulations earlier, interactive decentralized systems are hard to imagine for people. Fast iterations of prototyping and testing allow people to experience such systems.

5.1 Coming Home Ritual

This design case is based on a simple scenario: a user arrives at home, opens the front door, puts his coat at the hall stands, switches on the lights in the living room, gets a soda from the fridge, and goes to the living room to watch TV. The scenario is simplified to a set of 6 simple agents, having binary output states. One agent represents the front door (open or closed), another the light switch (on or off position), and so on.

The agents are identical to the ones described above, and programmed with an associative memory: they can perceive short series of events, and in the case of repeating patterns in which they take part, memorize them. The concrete implementation is as follows. Users can toggle the state of the agents by touching the touch sensor of the agents. The toggle signal is broadcasted, so other agents can perceive it. Whenever an agent's state is toggled, it reinforces the pattern from its short-term

memory into its long term memory. This way, it gradually learns to recognize repeating patterns, in which it takes part. At a certain level of reinforcement, the agent will toggle itself when the recognized pattern comes along.

Agents can only learn short fragments of patterns: the long term memory can allocate memory chains of three transitions preceding their own transition. Longer patterns can develop in this system as a whole, but cannot be known to any of the agents. Users can train a pattern by just touching the objects repeatedly in a specific order and frequency and the system demonstrates to be capable to reproduce the pattern autonomously. As can be expected, when different patterns are trained simultaneously, or if the end of a certain pattern is coupled to the start of another, the system can exhibit unexpected interfering repeating patterns.

5.2 Smartgoals

The coming home case taught us that having embodied prototypes turned out to make all the difference to our test users. Suddenly the concept of a decentralized approach came to live. It also taught us that the context was abstracted too much to be experienced as realistic by test persons.

Therefore, for the second iteration, a realistic, but simple, context has been selected: sport, soccer training to be specific. Soccer training has all the aspects we have to deal with in a complex human environment like a home or an office: people with intentions and motivations, physical space, rules, interactions. The associative memory is modified: only the strongest ten patterns will be remembered. Patterns slowly fade unless reinforced. With these starting points, the Smartgoals have been developed.

Fig. 2. Smartgoals field test

The Smartgoals game is played with 6 interactive goals randomly positioned on a space of 50 to 100 square meters. At every moment, a fixed number of goals (usually one or two) are “active”. The lights on that goal are on, and if a ball passes an active goal it is counted as a score. Whenever a score occurs, the goal at which the score happened is deactivated and another goal is activated. A possible goal of the game is to score as much as possible in a set time. The game can be played alone, in couples or with teams. It can be cooperative or competitive, depending on the interpretation

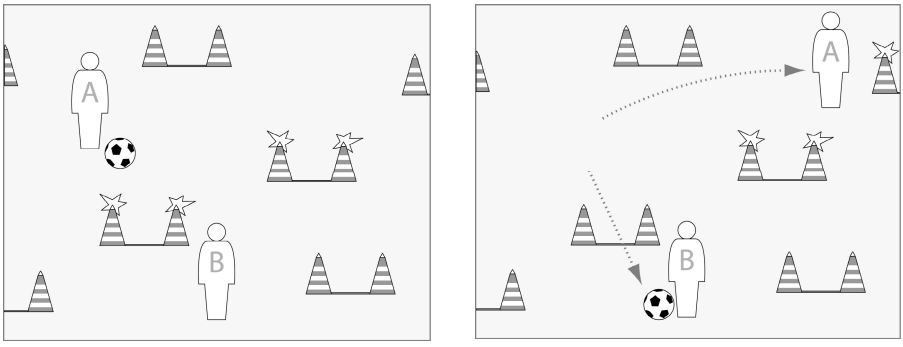

Fig. 3. A game variant with two players cooperating, and two active goals. In the left figure, the players choose the left active gate. Immediately after passing the ball, player A chooses position at the other active goal. When the ball passes the gate, it is deactivated and another goal becomes active.

trainers and players give to it. User can try to anticipate to the expected system behavior, the actual behavior strongly depends on the users actions.

A single Smartgoal has no knowledge of the intentions of the users, it does not know how many other Smartgoals are in the system, and it does not know its users. Yet, it turned out that even a system of only 6 goals can form a highly challenging, interactive training setup. With changes in the spatial configuration, the initial states of the individual agents, the number of players, or small changes in rules for agent behavior, a wide range of exercise forms could be generated. Currently, this concept is developed further for a startup company.

5.3 Synchronizing Fireflies

Reflecting on the Smartgoals, we conclude that this game clearly demonstrates the potential of the simplicity of a decentralized system, once it is brought alive by interaction with users. Moreover, the context of sports and games seems very appropriate for involving users in the research process. The decentralized implementation adds challenge, surprise, and user anticipation to a “traditional” game. A shortcoming of this iteration is that our approach to programming the behavior of the agents has been rather ad hoc. As our goal is to obtain understanding and derive heuristics for the design of decentralized systems also in other applications and contexts, we need knowledge that can be generalized. Therefore more transparent descriptions of the relations between the agent’s input and output are required, which can be mapped with observed (emergent) system behavior.

We decided to start exploring more formal descriptions of the agents’ interaction rules. In particular we apply generic state descriptions. Apart from its flexibility, the use of state descriptions in the agents is also useful because it corresponds to approaches in different fields of research. As a design case, we have implemented the synchronizing behavior of fireflies in order to demonstrate emergent behavior in small sets of physical agents. The relatively simple system dynamics of synchronizing

fireflies and the clearly recognizable emergent behavior makes it a suitable case. In this particular exploration interactions with users are not yet implemented.

A firefly is a bug that can emit periodic light flashes using a chemical process called bioluminescence. Large groups of fireflies have the property that they can synchronize their flashes. Strogatz [16] analyzed that an individual fly can be seen as an agent. The input of this agent is the detection of light emission by other fireflies. Output is the light emission of the fly itself. The system is formed by the population of fireflies reacting on each other. The (simplified) individual firefly can be modeled as a pulse generating oscillator.

The state description is one of the most general and transparent ways of describing interaction rules. States allow dynamical updates, which imply that agents can have memory. Memory is locally stored in state variables. In a difference equation, the new values of the states are computed based on the current values of the states and new values of the inputs (which are related to outputs of other agents and direct input from users). The new outputs are a function of the updated states.

Many phenomena in decentralized systems are based nonlinear mechanisms, such as threshold values, discontinuities, local extremes etc. Therefore it is necessary to introduce nonlinear equations. To allow a generic and flexible state description, in the implementation a strict separation between non-linear functions X and linear state functions Q has been made:

$$X_q(T) = f\left(input_p(T), Q_n(T)\right)$$
$$Q_n(T+\Delta t) = A + B \cdot Q_n(T) + C \cdot input_p(T) + D \cdot X_q(T)$$
$$output_m(T+\Delta t) = E \cdot Q_n(T+\Delta t)$$

In which: q is the number of non-linear functions, n is the number of state variables, p is the number of inputs, m is the number of outputs, and the matrices A [1xn], B [nxn], C [pxn], D [qxn], and E [mxn] define the constant parameters in the linear state functions. Note that due to the time discretization, the value of Δt is encapsulated in the parameter values in the matrices.

Three states are used to describe the behavior of the firefly-agent in the generic state description. The first state variable Q_1, represents the firefly's building up of charge. A constant increase combined with a dissipation factor results in a basic exponential increase of charge. When the energy level reaches a certain threshold value, the fly will flash, and the charge level drops down. When an agent 'detects' the light flash of another agent, it can make a little extra step (effort) in charging. This results in a tiny extra increase in charge. The interaction with other agents in the form of an extra charge pulse eventually results in a synchronization of the light pulses of a group of agents.

The discharge threshold level is represented by the second state variable Q_2. In case the initial value (or the natural frequency) of this threshold varies over the separate objects, a dynamic adaptation mechanism of the value of Q_2 allows for synchronization. The threshold is a non-linear component creating a periodical discharge. The nonlinear threshold function X_1 indicates the timeslot in which the fly

fires (the int operator truncates the sub decimal values variable to the nearest integer value below).

$$X_1(T) = Q_1(T) \cdot \text{int}\left[\frac{Q_1(T)}{Q_2(T)}\right]$$

The third state variable Q_3, is the flash indicator. In fact Q_3 equals the nonlinear state function and is only required to link the flash indicator to the linear output equation.

In order to test the implementation and demonstrate the opportunities of the agents for emergent behavior, three different firefly models are implemented in the agents. The models differ in synchronization mechanism (adaption speed, adaptation of capacitor charge and adaption of pulse frequency) resulting in different nonlinear functions and parameter sets in the matrices. Every model has a slightly different approach on achieving synchronization. The experimental results are compared with computer simulation results, based on virtual agents with a similar state description.

Experiments proved that it is possible to construct this self-synchronizing system with a set of general state descriptions in the hardware set of agents. The use of state description proved to be generic and flexible. The transparent description allows for a better insight in the mapping between model parameters and the resulting dynamic behavior of the complete decentralized system. Preliminary comparisons between computer simulations and the hardware agent experiments proved similar results.

5.4 Next Iteration: Interactive Objects for Open-Ended Play

The Smartgoals design case proved that the context of games and play provides great opportunities to explore decentralized interactive concepts with users. The system is open-ended to a certain extent: players or trainer might define the rules and the goals to be achieved in the game in many different ways. However, once defined, rules and the goals were not likely to evolve during play.

In a broader interpretation of open-ended play, exploration and discovery are more important than reaching a specific goal. The next design case aims at this broader interpretation by designing interactive play objects which, in interaction with playing users, demonstrates these opportunities of open-ended play. The decentralized systems approach seems especially appropriate to develop open-endedness. The assumption is that the dynamically changing behavior enabled by the interaction with the objects creates inspiring and unpredictable open-ended play opportunities. We will use the framework for successful design of an intelligent, interactive playground from Sturm [17]. An important aspect will be the balance between predictability and surprise. To deal with the expected increase in complexity, we build upon the work on formal descriptions of interaction rules. This will enable better mapping between parameter settings, system behavior, and experience of the user.

The expectation is that interactive objects, designed to facilitate emergent behavior, will be more exciting and fun to play with. We also expect that open-ended play stimulates children to develop their social skills, because they need to communicate and negotiate about the goals and rules of the game to play. Although we have just started this design case by creating scenarios and context studies, the interesting combination between new implementation methods and social, interactive opportunities of game

design to conquer societal problems like loneliness, isolation and obesity offers great opportunities.

6 Conclusions

The research-through-design approach works. The first iteration helped to understand and communicate the concept of a decentralized approach to intelligent interactive environments. The approach allowed developing new directions and it enabled other researchers to imagine what is possible with decentralized systems, and thus opened opportunities for inspiring cooperation. The second iteration, the Smartgoals, demonstrates the surprisingly high potential to motivate and engage users, it also demonstrates the opportunities of realizing emergent behavior brought alive by interactions with users. It demonstrates that the user is an unpredictable, but essential element in our approach which cannot be captured with standard frameworks for multi agent systems. The third iteration proved to be a step in demonstrating emergence related to more formal and theoretical frameworks. In general, the context, the prototyping, and solving practical problems in all iterations contribute not only to our insight, but also allow users and colleagues to experience such systems.

Does the decentralized approach open new directions for open-ended play? The answer to this question is positive. The Smartgoals were a convincing demonstrator of the potential of even a simple system consisting of a handful of identical agents, one or a few users and a simplified use context. Still, a big challenge remains, when a broader interpretation (exploration and discovery) of open-endedness is chosen. The next design case aims to address this issue.

The research goal is to derive design heuristics for the design of decentralized systems. Although the presented design cases are just a first step, it is possible to make some interesting observations. The exploration of memory in the agents indicates that memory of agents may be a key to stabilizing a decentralized system. A more structured use of memory forces the system to converge to a limited set of repeating patterns, as we explored in the first and second design case. This implies that factors like surprise and predictability can be tuned by balancing randomness and the strength of the associative memory. Challenge, an important factor in games, appears to be related to surprise.

Another issue explored is choice and (perceived) user control. In the soccer game, if only one goal at a time is active, players have no choice and the game basically is about skills in passing and stopping. If two goals at a time are active, players start communicating about possible strategies. The nature of the interactions between players changes with this simple parameter. Especially the notion of user control in relation to system behavior will be explored in the next iterations on open-ended play.

After three research-through-design iterations we feel we are on a challenging, engaging, and definitely open-ended road, which is worth to be explored for new intelligent play solutions. Eventually, insights developed in sports and play concepts also contribute to intelligent interactive environments in other contexts.

Acknowledgments. We thank Maarten de Jager, Chris Heger, Sjef Fransen and Vic Hensberg for their substantial contributions to the design iterations. Tilde Bekker and Janienke Sturm have inspired us with their ideas on open-ended play.

References

1. Bekker, M.M., Sturm, J., Wesselink, R., Groenendaal, B., Eggen, J.H.: Play Objects and the effects of open-ended play on social interaction and fun. In: Proceedings of ACE, International Conference on Advances in Computer Entertainment Technologies (2008)
2. Camazine, S., et al.: Self-Organisation in Biological Systems. Princeton University Press, Princeton (2003)
3. Dresner, K., Stone, P.: Multiagent Traffic Management: An Improved Intersection Control Mechanism. In: AAMAS 2005, Utrecht, Netherlands (2005)
4. Franks, N.R.: Army Ants: A Collective Intelligence. American Scientist 77(2)
5. Fromm, J.: The Emergence of Complexity. Kassel University Press (2004)
6. Fromm, J.: On Engineering and Emergence (2006), http://arxiv.org/ftp/nlin/papers/0601/0601002.pdf
7. Herman, R., Gardels, K.: Vehicular Traffic Flow. Scientific American 209, 6 (1963)
8. Holland, J.H.: Emergence, From chaos to order. Perseus Books (1998)
9. Krugman, P.: The Self-organizing Economy. Blackwell, Malden (1996)
10. Mainland, G., Parkes, D.C., Welsh, M.: Decentralized, Adaptive Resource Allocation for Sensor Networks. In: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, vol. 2 (2005)
11. Reynolds, C.W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics 21(4) (1987) (SIGGRAPH 1987 Conference Proceedings)
12. Seeley, T.D.: The Honey Bee Colony as a Super Organism. American Scientist 77 (1989)
13. Settembre, G., Scerri, P., Farinelli, A., Sycara, K., Nardi, D.: A Decentralized Approach to Cooperative Situation Assessment in Multi-Robot Systems. In: Proceedings of International Conference on Autonomous Agents and Multiagent Systems, Portugal (2008)
14. Singh, A., Srivatsa, M., Liu, L., Miller, T.: Apoidea: A Decentralized Peer-to-Peer Architecture for Crawling the World Wide Web. In: Callan, J., Crestani, F., Sanderson, M. (eds.) SIGIR 2003 Ws Distributed IR 2003. LNCS, vol. 2924, pp. 126–142. Springer, Heidelberg (2004)
15. SPORE, Electronic Arts Inc (2007), http://www.spore.com
16. Strogatz, S.: Sync, the emerging science of spontaneous order. Penguin books (2003)
17. Sturm, J., Bekker, M.M., Wesselink, R., Groenendaal, B., Eggen, J.H.: Key issues for the successful design of an intelligent, interactive playground. In: Proceedings of the Interaction Design and Children Conference (IDC 2008), Chicago, IL, USA, June 10-13 (2008)
18. Weiser, M.: The Computer for the 21st Century. Mobile Computing and Communications Review 3(3) (1991)