MOBILE VIRTUAL WORLDS: A PROXIMITY BASED EVOLUTION

Stefano Cacciaguerra, Gabriele D'Angelo
Department of Computer Science
University of Bologna
Via Sacchi 3, Cesena (FC), Italy
E-mail: {scacciag, gdangelo}@cs.unibo.it

KEYWORDS
Virtual Worlds, Pervasive Entertainment, Agent-based Entertainment, Game Design.

ABSTRACT

The wireless revolution has enabled a new generation of applications for nomadic users. In this work we propose a new paradigm for the creation of games based on virtual worlds that are hosted on mobile devices. Each time mobile devices *"get in touch"*, their virtual worlds have the opportunity to interact. This form of interaction is based on the remote control of a subset of the agents that populates the virtual world. In accord with this, it is possible to create games with an unpredicted and unforeseeable evolution. Finally, we introduce PReDA, a prototypal implementation of the proposed mechanism that is based on the Netlogo environment.

INTRODUCTION

The pervasive diffusion of the wireless technology has lead to wide effects on the ICT field. First of all the wireless access to the Internet telephony and, then, the use of wireless mobile devices to browse the Web, anytime and anywhere. Thanks to more capable hardware, pervasive networks and adaptive software protocols, the wireless technology is fostering a new generation of applications, as an example: sensors' networks, wearable computers, ubiquitous and context aware applications (Chen at al., 2003; Kanter, 2003).

In this scenario, it is easy to predict a future where mobile users will daily use many forms of Internet access (e.g. wireless hotspots, private networks, ad hoc networks etc.), in order to share contents and to take advantage of the resources offered by the broadband connectivity. As an example, the participative virtual worlds are gaining more and more popularity: they allow the users to keep in touch with friends and colleagues, to collaborate in the resolution of shared tasks, to run brainstorming meetings and to share common resources. It is worth noting that this kind of social environments supports both forms of collaboration and competition between users, building a new kind of interactive and immersive metaworld. To some extent, using these technologies is possible to build virtual worlds that mimic many of the daily activities (Linden, 2007). In this case, the world is virtual and under some viewpoints it is safe: it represents a sort of sandbox.

In this field, an important role has been played by the participative simulation (Colella et al., 1998; Wilensky et al., 2007) that is a gaming activity often used to explore complex systems. As an example (Terna, 2003), a virtual marketplace where users can be sellers and customers, at the same time. As another example, the road transportation: we could imagine a system where each user is in charge of managing a specific traffic light, with the capability to trigger the "green" and "red" lights. In this case the users would be able to choose a strategy based on collaboration or competition. For the sake of simplicity and clearness, all these examples are immediate and obvious, but it is worth noting that participative simulation has many fields of application in both scientific and technical areas (e.g. diffusion of viruses, vehicles behavior, modeling of molecules in a membrane and the prisoners' dilemma).

The main part of a participative simulation is the underlying Multi-Agent System (MAS). It offers a programmable system to model and simulate complex behaviors. In a MAS, the simulation developer can define groups of agents, to each group can be assigned autonomous or shared tasks, and can be instructed to achieve a specific goal. The global state of a multi-agent simulation is obtained as the result of a large set of interactions among the agents. During the simulation lifespan each agent will be part of many local interactions, data exchanges, cooperation and competition activities. Each agent (during its artificial life) can be driven by a form of artificial intelligence or by a human being. In both cases, the agents will play following their capabilities, in example their past knowledge, the ability to explore the environment and the available activities.

In this scenario, the wireless technologies enable the participation of nomadic users, extending the access to MASs also to users with mobile devices. The first and more direct consequence of the wireless technology is that human users can remotely control agents that are within a MAS, anytime and anywhere. Furthermore, less immediate and more complex consequences can be foreseen. Despite of a human being, the player could be a remote MAS that controls one or more external agents. In this case the interaction would be between MASs. Using wireless connectivity, would it be possible to imagine that a MAS running on a mobile device would connect to another MAS, to take control of a part of its agents?

The main goal of this work is to demonstrate that this scenario is realistic, and to propose a new framework based on the Netlogo environment. Following this approach, different MASs implemented by Netlogo will be able to interact together, based on their proximity, and to affect the evolution of the whole system. In this case, we are not proposing a new game based on MASs, but a new paradigm for the creation of games based on mobile virtual worlds.

Proceedings of the International Conference on Intelligent Games and Simulation (GAME-ON 2007).
Bologna, Italy, November 20-22, 2007.
ISBN 978-9-077-38137-3

Following this approach, the evolution of the game is determined by new factors, as the proximity of gamers and the consequent random interactions between MASs. In our vision, this work is the first step in the direction of a new class of mobile games.

The next Section explains why it is important to give the possibility to remotely drive agents. The third Section shows similarities and differences with computer entertainment applications. The fourth illustrates the system architecture. The fifth suggests a case study based on Netlogo. Finally, we conclude this paper with some final remarks and future works.

REMOTE DRIVEN AGENTS

A MAS allows to represent, mimic and study complex systems where different components interact among them in a cooperative or competitive way. It promotes the understanding of a complex system by means of the description of its rules and the representation of its evolution. Modern MASs are based on 2D raster graphical functionalities (North et al., 2006; SWARM, 2007; Wilensky, 2007) that, in many cases, are inadequate for the human perception. Only lately, the introduction of recent 3D rendering engines (Cacciaguerra et al., 2004; Wilensky, 2007) has lead to higher expressivity and a better representativeness of the system (see Fig. 1). Expressivity becomes either the capacity to mimic, with a higher detail of accuracy, a complex system (if this is necessary for the modeling effort) or to show to the viewer (i.e. the player) another dimension in order to enhance his comprehension. Essentially, we are introducing a new dimension to our discussion, referring either to another physical dimension (i.e. geometric plane) or an improvement of the representation (i.e. expressive power). In accordance with these considerations, we believe that enabling a MAS to control the agents of another one, would permit to add a new dimension in the evolution of complex systems. The idea behind this approach is not related on the opportunity to decrease the computational load, distributing the agents on other computers (that is a well known approach in literature, e.g. Riley, 2003). It refers to the opportunity that two or more MASs get in touch and interact when are close, that is, under the wireless coverage area of one or more network adapters. This translates to a system that can evolve in a “less deterministic” (i.e. unpredictable) mode: that is because the interaction due to the *proximity* with other systems would be able to change their evolution. This kind of system will be by far more unpredictable than a system where all agents are driven by a single piece of local software (i.e. the standard approach). In the first case, the movements of mobile devices are the basis for unplanned meetings and the availability of a wireless network is the media that allows the interaction. In this scenario, the unplanned meetings add a new degree of indeterminism to the whole system. Furthermore, following this approach, the game modelers can define different behaviors of agents when reacting to the same perception, implementing different course of action. In any case, all the implementations are bounded by a set of game-related roles. For example, in the wolf-sheep predation model, the sheeps should adopt different strategies to eat the cabbage and to flee from the wolves. In any case, it is not acceptable that a sheep eats a wolf! In other words, this means that the freedom in the implementation of a specific behavior have always to be coherent with the specific role of the agent. Following this approach, it is possible to mix the behaviors of agents that have been implemented by different parties. This will allow to generate combined actions that can lead to results that are unpredicted and unforeseeable in the original system. In a causal meeting, the exchange of behaviors among systems that are hosted on mobile devices, can be described as the spread of a virus in an epidemiologic scenario. In this metaphor, the remote system can affect the behavior of many agents due to the *contact* (i.e. the proximity). A posteriori, if the resulting effect is seen as interesting then it would be possible to analyze the log files, in order to trace the interactions and inspect step-by-step the evolution of the system. In most cases, videogames can be considered as complex systems. Therefore, very often the participatory simulation is seen as a form of game-based e-learning. In this sense, we think that our approach could be a first step in the direction of a new paradigm for mobile gaming.

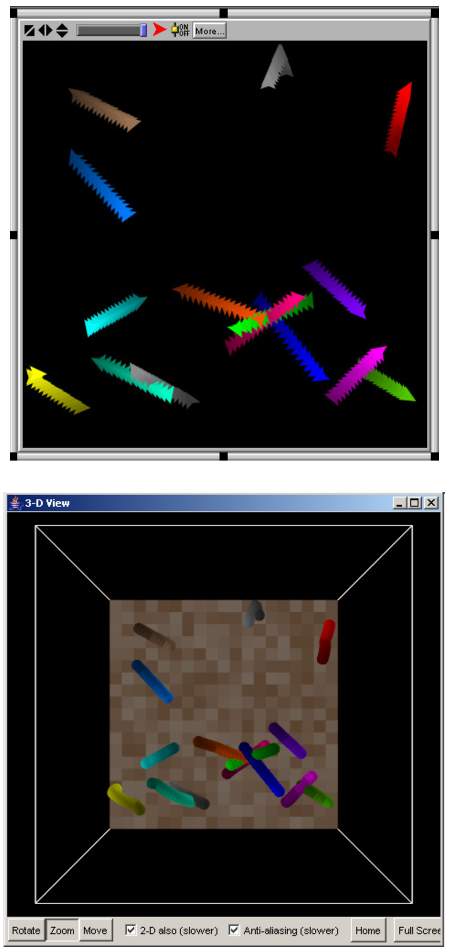

Fig. 1 Display of NetLogo bouncing balls model: 2D and 3D

RELATED WORK

In the state of art of computer entertainment there are some applications partially based on the proposed paradigm, such as: Ubiquitous Monster (Kawanishi, 2005), Insectopia

(Peitz et al., 2007), WSNMP (Liu et al., 2006) and Pirates! (Björk et al., 2001).

Ubiquitous Monster is a monster collection videogame where the players wander about the real world to collect monsters that will be used in the virtual world. In this case, the transfer of monsters is based on the RFID technology. Within the game, the behaviors and aptitudes of monsters are predefined, and they appear in the virtual world in relation with the geographical position of the player. The monsters, accommodated in the virtual world running on a mobile device, can: born, make friends, evolve, breed and die on the basis of the weather conditions (i.e. lightness, temperature, pressure, electric potential). All these conditions are detected by a sensors network in the area where the user is located. For example, given a monster that uses the light to obtain its vital energy, it is simpler to capture it in a sunny place instead of a dark zone. Further, the weather and light conditions change along the day: so, it is very difficult to find such kind of monster during the night! In the game, when two players meet, some monsters may migrate from one virtual world to another, in order to find more comfortable environments, and therefore to obtain as much energy as possible. This game promotes the movement of the players in the real world in order to collect different monsters and to exchange them, by means of the migration process. Since that the virtual world is an ecosystem with limited resources, it is not possible to capture a great number of monsters. Therefore, the aim of the game is to reach a sort of instable equilibrium in each local ecosystem, trying to support the higher possible number of monsters of different breeds.

Similarly, Insectopia is an insect collection video game running on mobile phones. Each player must collect and domesticate his insects. The lifespan is limited to a fixed amount of time. After a week, each insect dies and the player must capture a new one. The catch of a specific insect depends on the type of mobile devices that are in proximity of the player. This game adopts the Bluetooth technology in order to discover the different types of mobile devices.

Wireless Sensor Network based Mobile Pet game (WSNMP) is a game where the user must control domestic animal by means of a mobile device. In this case, the game is based on a wireless sensor network. The players can interact with virtual pets, feeding them, taking care of them and playing with them. And furthermore, they can share their pets with others, trade them, watch them compete against each other, become friends, create offspring and develop a virtual pet society. The players can also communicate each other through their shared pets. Each virtual pet is represented by a sensor node. The sensor nodes are composed by many sensors; each sensor is an organ of perception, such as, light detector, smoke detector and microphone for eye, nose, and ear, respectively.

Pirates! is a videogame where the player takes the role of a captain pirate sailing his ship in a fantasy archipelago. The ship permits to transport commodities from the different islands in order to be sold at markets. Each ship has a crew and is equipped with cannons. If the captain successfully completes the missions, then he can sturdier the ship thanks to the gained rewards. There are some dangers such as sinking in a battle, meeting cannibals or getting lost during the exploration of an island. The aim is to find treasures and commodities in each visited island. Islands are different and provide many kinds of merchandise and dangers. Further, at the free harbor it is possible to recruit new crew members, to repair a ship, to trade for goods and to obtain a new mission. Each ship is represented by a PDA equipped by an IEEE 802.11 WLAN card and a RF proximity sensor, while the islands are physical locations in the real world (e.g. different rooms in a building).

The similarities of these applications with our proposal are: i) the dynamism in the evolution of the ecosystems, and ii) the migration of agents among mobile devices. The above introduced applications use sensor networks to detect the ambient conditions and to get the geographical position. Differently in our approach, the possible evolutions of the virtual world are, *a priori*, less predictable. In fact in the other approach, the behavior of monsters, insects, pets or pirates does not change during the lifespan of the game: all of them are defined and implemented by the application developer and can not be changed at runtime.

To the contrary, in our approach, each player can modify the behavior of its agents (e.g. implementing new actions) in each moment, also if the game has already started. The only imposed limitation is to respect the general rules of the virtual world that, above, we have called roles. In this way, it will be impossible to define a priori the evolution of the simulated ecosystem. A posteriori, it will be very important to study the evolution of the system, inspecting the different phases of the evolution and taking care to study emergent patterns. This analysis will be possible using the log files that trace the evolution of the whole system.

SYSTEM ARCHITECTURE

In our vision, we have a set of virtual worlds, each one runs on a different mobile device. The virtual worlds should be able to interact together depending on their proximity. The position of a virtual world is due to the mobile devices that hosts it. Each virtual world is composed by a set of agents living in a defined environment. When a virtual world gets in touch with another one, it can take control of a sub-set of the agents in the other one.

In accord with this vision, we implemented the Proximate Remotely Driven Agents (PReDA) framework. PReDA is a prototypal communication framework, based on the proximity of mobile devices, that is in charge of: i) discovering devices that host a PReDA virtual world; ii) managing the communication among PReDA virtual worlds and iii) enabling the remote control of agents.

Given such requirements, the proposed architecture is based on ultra-portable notebooks, tablet PCs and PDAs, Java-enabled and with Bluetooth connectivity. The discovery phase of PReDA takes advantage of the Bluetooth discovery mechanism. Each device continuously searches other devices within its coverage area. Each time a new device is detected, an inquiry scan is performed to obtain the list of available services (i.e. a virtual world based on the PReDA framework). If the new device is running PReDA, then it is possible to start a direct communication between the local and the remote virtual worlds. The communication is implemented using the Bluetooth Logical Link Control and Adaptation Protocol (L2CAP). PReDA uses the L2CAP protocol to pair the local and the remote virtual worlds. Each instance of PReDA verifies if there are remotely controllable agents that are flagged as available. An agent is available if no other PReDA systems are now controlling it. If at least one available agent is found then the local virtual world will send a set of commands to a subset of them. Due to the nomadic nature of the hosting devices, only a limited amount of time (in order of a few seconds) would be available for the interactions among virtual worlds. Therefore, in a short time frame, it is possible to transfer a single command or a complex behavior (in form of a set of actions). In the following section we will introduce a prototypal implementation of PReDA based on Netlogo.

A CASE STUDY WITH NETLOGO

NetLogo (Wilensky, 2007) is a programmable tool that allows to simulate the evolution of complex systems. This tool permits to the modeler to give instructions to a high number of independent agents all operating concurrently, either in a cooperative way or in a competitive one. Therefore, it promotes the exploration of the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from the interaction of these individuals. Further, the users can "open simulations" (i.e. explore the internal state), can play with them, in order to explore their evolution under various conditions and can create their own models (i.e. implementing new functionalities). This tool is simple enough that any user can easily run simulations or even implement the behavior of his agents. The possibility to see the code of other models and to access an elevated number of high-level primitives promotes the reuse of the code, allowing everyone to implement its own routines. The simple approach, that can be used to program the tool, does not reduce the expressive power of the models that can be simulated, making it an interesting tool for many research fields (i.e. the simulation of many natural and social phenomena). Moreover, the Netlogo community is very active and has made a large number of models freely available, models that are related to many fields as: biology and medicine, physics and chemistry, mathematics and computer science, economics and social psychology. This wide adoption demonstrates that Netlogo is very easy to use and that can cope with many different topics and problems. One of the most interesting features (introduced in version 2.0) is the "extensions" module, it allows the developers to introduce new commands and reporters (Wilensky, 2007) that can be used inside the Netlogo environment. The idea behind this module is to extend the primitives by means of Java code, that is archived in a .jar file. In this way, it is possible to write high-level functions but also to integrate the Netlogo environment within other projects! In accord with this consideration, we have integrated the Netlogo environment within a new framework. The goal of this new framework is to support communication among many Netlogo environments that are executed on different devices. The framework exploits the Netlogo extensions module to obtain this result. The framework has to provide two main functionalities: i) it must supply an access point to each local instance of the environment by means of an agent discovery system; ii) it must support the exchange of commands among different mobile devices. The communication between different environments needs a protocol that permits to exchange commands (as string of characters) from different worlds (e.g. Netlogo instances). Given the "A world" and the "B world", that are different environments accommodated on two mobile devices, our mechanism provides a form of addressing (i.e. to make the environments reachable) and a communication protocol.

In detail, Netlogo classifies agents into two types: passive and active. The virtual world is divided in square pieces of ground, each piece is called "patch". Netlogo classifies the patches as passive agents. These agents can be affected only by active ones and by the Observer (that is the Demiurge of the world). The active agents (called turtles) can interact among them and with the patches.

In the following, we report some details about the implementation of our framework. Firstly, we report a piece of code from a Netlogo model. It is worth noting that the example contains an include of the *PReDAextension.jar* archive. This archive provides the basic functionalities for communication and discovery. In this way, the developer can directly use its own Java routines inside Netlogo. Each Netlogo model begins with the pressure of a button that starts the *setup* of the ecosystem, initializing the agents and the environment variables. In particular, in the code example reported below, after the initial setup, the setup bootstraps the discovery system specifying which agents can be remotely controlled. In detail, the *rmt-crt-turtles* and *rmt-crt-patches* routines report the set of turtles and patches that will be remotely controllable. By means of the definition of these routines, the user can determine which agents can be remotely controllable while leaving untouched the others. The *startCommFrmwrk* initializes the communication framework. The *GO* button runs the body of a Netlogo model. The *ask* construct is used to specify commands that are to be run by a set of agents. The *run* routine allows an agent to interpret the given string as a sequence of one or more NetLogo commands and runs them. The routine *recvmsg* that has been declared inside *PReDAextension.jar* receives a message, that is a string sent by a remote Netlogo environment accommodated on a mobile device. Obviously, this means that the remote Netlogo environment will use the *sendmsg* routing (also in this case defined in *PReDAextension.jar*) to send messages, that are strings dispatched to one or more remotely controlled agents.

In last part of the code, it is possible to analyze the approach used in the implementation of our framework. In the first case (see Example 1), the received string will contain the number of steps that a set of turtles must cover. In the second case (see Example 2) is defined the new color of a set of patches. For example, if the string returned by *recvmsg* is "3", then all turtles will be moved straight of 3 steps in their direction, while, if it is "yellow" then all patches will become yellow colored.

It is worth noting that in the third case (see Example 3 and Figure 2), the string directly reports a sequence of commands, respectively: i) "*ask turtles with color = red [fd 3]*" - all red turtles will be moved straight on their direction of 3 steps, while, ii) "*ask patches with pcolor = yellow [set energy 0]*" - the energy level of all yellow patches will be decreased to zero.

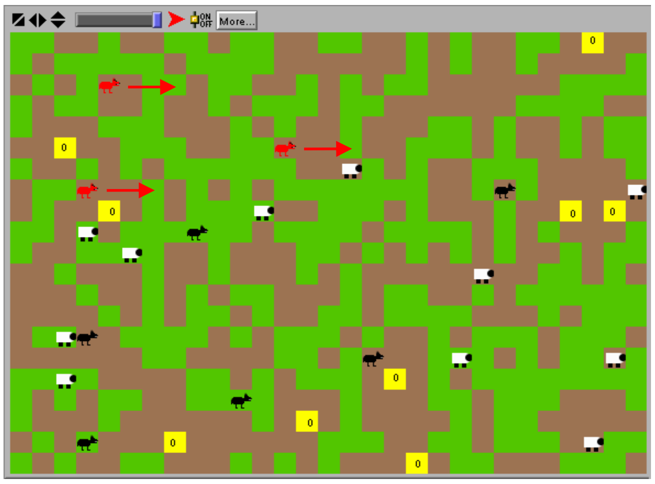

Fig. 2 NetLogo Wolf – Sheep Predation

```
__extensions["PReDAextension.jar"]

to setup
  …
  setup-turtles
  setup-patches
  set clock 0
  …
  Discovery(rmt-ctr-turtles, rmt-ctr-patches)
  startCommFrmwrk
end

to GO
  …
  ask turtles
  [
    …
      run fd recvmsg   ;; (Example 1)
    …
  ]
  ask patches
  [
    …
      run set pcolor recvmsg  ;; (Example 2)
    …
  ]
  …
  run recvmsg  ;; (Example 3)
  …
  set clock clock + 1
end
```

CONCLUSIONS AND FUTURE WORK

In this work, we have introduced the Proximate Remotely Driven Agents (PReDA) framework. PReDA is a prototypal framework based on the Netlogo environment and its "Extensions" module. PReDA exploits the Bluetooth connectivity in order to discover other PReDA copies running on top of mobile devices. Each mobile device runs a local virtual world composed by an environment and a set of agents. In this way, the PReDA-based virtual worlds running in each mobile device can interact together, taking control of a part of the remote agents. Since each player can easy modify the behavior of agents and since the game is subject to random interactions, the evolution of the game will result very unpredictable.

As a future work, we aim to build a real game based on the proposed paradigm. Furthermore, we plan to develop a light version of the PReDA framework that will run on Java-enabled mobile phones.

Another direction of this research will involve the study of the behavior of nomadic users and the impact of the proposed paradigm on social sciences.

REFERENCES

Björk S., Falk J., Hansson R., Ljungstrand P., 2001 Pirates! - Using the Physical World as a Game Board. *in proc. of Human-Computer Interaction conference* (July), Tokyo, Japan.

Cacciaguerra S., Mirri S., Pracucci M., Salomoni P., 2006. "Wandering about the City, Multi-Playing a Game" *in proc. of IEEE International Workshop on NIME* (January), Las Vegas (NV-USA).

Cacciaguerra S., Roccetti M., Roffilli M. Lomi, A, 2004. "Wireless Software Architecture for Fast 3D Rendering of Agent-Based Multimedia Simulations on Portable Devices" *in Proc. of the First Consumer Communications and Networking Conference (CCNC)*, IEEE Communications Society (January), Las Vegas (NV-USA)..

Chen H., Finin T., 2003 "An Ontology for a Context Aware Pervasive Computing Environment", *in Proc. of IJCAI Workshop on Ontologies and Distributed Systems*.

Colella V., Borovoy R., Resnick M., 1998 "Participatory Simulations: Using Computational Objects to Learn about Dynamic Systems" *in Proc. of Computer Human Interface Conference*, (April) Los Angeles (USA - CA).

Kanter T. G., 2003 "Attaching Context-Aware Services to Moving Locations" *in IEEE Internet Computing Magazine*,(March-April) Vol. 7, N. 2.

Kawanishi N, Kawahara Y., Morikawa H., Aoyama T., 2005 "Prototyping a Real-World-Oriented Monster-Collection Game" *in Proc. of 5th International Workshop on Smart Appliances and Wearable Computing*, (June) Columbus,(USA - OH).

Linden Lab 2007, "Secon Life", http://www.secondlife.com/.

Liu L., e Ma, H., 2006 "Wireless Sensor Network Based Mobile Pet Game", *In Proc. of NetGames*, (October), Singapore.

North M.J., Collier N.T., Vos J.R., 2006 "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit", in *ACM Transactions on Modeling and Computer Simulation*, Vol. 16, Issue 1, pp. 1-25, (Jannuary), New York (USA - NW).

Peitz J., Saarenpää H., Björk S., 2007 "Insectopia - Exploring Pervasive Games through Technology already Pervasively Available" *in Proc. of Advanced in Computer Entertainement Technology*, (June) Salisburg (Austria).

Riley P., 2003 "SPADES: System for Parallel Agent Discrete Event Simulation", in AI Magazine.

Terna P., 2003 "Decision making and enterprise simulation with jES and Swarm.", *in Proc. of the Seventh Annual Swarm Users/Researchers Conference (April)*, Notre Dame, Indiana (USA - IN).

Swarm Development Group. 2007 "Swarm" http://www.swarm.org Swarm Development Group, Santa Fe (USA – NM).

Wilensky U., 2007 "NetLogo" http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Wilensky U., Stroup W., 2007 "HubNet" http://ccl.northwestern.edu/netlogo/hubnet.html. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.