Urbarium – A Socially - Based Game Platform

Robert G. Reynolds
Dept. of Computer Science
Wayne State University
Detroit, Michigan 48202
313-577-0726
reynolds@cs.wayne.edu

Farshad Fotouhi
Dept. of Computer Science
Wayne State University
Detroit, Michigan 48202
313-577-2478
Fotouhi@wayne.edu

ABSTRACT

In this paper we describe a socially-based game platform, the **Urbarium,** developed using XNA, Torque, and Muppets that will support learning of concepts for the laboratory components of the IEEE/ACM CS1 and CS2 programming courses. The **Urbarium**'s narrative will span three different stages of social development in a real world location, the Valley of Oaxaca, Mexico. These stages will correspond to hunter-gatherer economies, tribal economies, and state organizations. The students interact with each other over a local-area network. By solving problems, students collect code segments, and exchange these segments to complete components. Completed components provide enhanced rewards to students on future moves. The idea is to support a net-centric approach to software development and interaction among students in the lab. We feel that this will support student learning within a network environment by weaving a **social fabric** of cooperation into the code development process early on in a students' career.

1. INTRODUCTION

Recently a number of researchers have brought attention to the cognitive and social factors that impact the creation and communication of knowledge [1]. It is clear that the social context in which the creation and learning of knowledge takes place is an important factor in determining what is learned by a student and how it is integrated into the student's knowledge.

In particular, a number of practical techniques have been proposed for the creation and communication of knowledge. One approach is to develop tools to support dialogues between problem solvers. Fischer developed tools to help participants review their actions [2]. A second approach is the use of metaphor as a vehicle improving learning of design tasks. Gordon suggested that problem solving frameworks that invoke appropriate metaphors for certain design tasks can be useful in proving the effectiveness of the design process [3]. A third approach involves the collection of domain specific strategies for problem solving and interaction [4]. These strategies can be expressed within a standard framework for retrieval and use. A fourth approach is to use stories and storytelling to illustrate difficult points in the teaching of concepts, and to tie seemingly unrelated concepts together in a narrative framework [5].

All of these approaches intend to weave a social fabric into the design process which allows students/designers to more easily map their past experience into the task at hand. In this paper we describe an approach, **the Urbarium,** which uses a gaming environment to support these activities. The **Urbrarium** is a sequence of socially motivated games that can be used as part of the programming laboratory for CS1 and CS2 courses as specified by the ACM. It tells the story of the social evolution of a culture within a real region of the world as the social system evolves from a hunter-gatherer to a state organization. This story spans both classes and provides a social thread in which the various software development concepts can be integrated. Each of the games is based on existing prototypes of multi-agent system models of a particular region of the world, the Valley of Oaxaca, Mexico

The **Urbarium** runs in a peer to peer network laboratory with agents interacting with each other via PC's. Within the **Urbarium** framework, code components are distributed throughout the landscape, or can be generated by solving related problems. These code components can be exchanged in different ways among the student to build up a collection of software tools. These tools can be used to collect larger rewards in the current landscape.

There are three basic games which focus on hunter-gatherer, village formation, and state formation respectively. Students in CS1 begin as hunter-gatherers collecting simple code components with basic functionality. Later in CS1 village formation is introduced and students begin to acquire data structures and algorithms that are suitable for those tasks. Village formation carries over into the beginning of CS2. There, the students begin to focus more on object oriented design and problem solving in terms of the state formation process.

In section 2 we give the laboratory context in which the **Urbarium** will be used. Section three gives an overview of the **Urbarium** game structures for each of the 2 laboraory courses. Section 4 describes the first of the three games in more detail as it can be implemented using the MUPPETS game programming environments. The other games are developed using XNA and Torque. Section 5 gives our conclusions.

2. THE LEARNING ENVIRONMENT

Wayne State is Michigan's only urban research university. Located in downtown Detroit, Wayne State's 12 schools and colleges offer more than 350 major subject areas to our 33,000 graduate and undergraduate students. It has the most diverse student body in Michigan -- more than 35 percent, or more than 11,800 students are minorities (based on Fall 2004 enrollment statistics). Of these students, 8,780 were African-American, 2,134 were Asian, 137 were Native Americans, and 764 were Hispanics [6]. Minority students are attracted to schools with a racially diverse climate and an appreciation for their presence 3.

The presence of minority students on campus acts as an incentive for other minority students to pursue their education in that environment [7].

In order to retain and engage such a diverse undergraduate population we have revised our Computer Science undergraduate program to include a one-credit hour laboratory with each of our courses including the undergraduate programming classes (i.e., CSC 2110, and CSC 2200- See Table 1). It is our vision that these mandatory labs will provide students with a supervised one-on-one instructional format that will focus on cooperative problem solving in a computer laboratory setting. These new labs have been developed to align our program with the goals of ABET accreditation and the IEEE/ACM curriculum 2001. Therefore, these games can be used in any other CS program that is subject to the accreditation process. In conjunction with these changes to the core courses, we have instituted a set of four-course sequences that will allow undergraduates to focus their elective courses in particular sub-disciplines within Computer Science. These tracks include Software Engineering, Databases, Networking, and Computer Gaming to name a few. This, we feel, will be an opportunity to engage the student within a given sub-discipline.

TABLE 1: Descriptions for CSC 2110 and CSC 2200

CSC 2110 Course Description In this course the student is introduced to basic object-oriented programming including classes, subclasses, class hierarchies, polymorphism, inheritance, templates, and exceptions. The use of object-oriented methods in supporting event-driven programming with graphic APIs is highlighted.
CSC 2200 Course Description In this course the student develops fundamental data structures using object-oriented techniques. These structures include stacks, queues, linked lists, and hash tables. Also, control constructs that can be used for problem solving such as iteration and recursion are implemented. These will be applied to problem solving along with related concepts of backtracking, and divide and conquer.

As part of the undergraduate laboratory course sequence (i.e., CSC 2111 and CSC 2201) we propose to develop a subset of assignments that utilize "real world**" socially motivated** gaming applications to address fundamental concepts in the course. We call this the "**Urbarium**". The purpose of these game-motivated programming exercises is to weave a **social fabric** into the programming process. The idea is to emphasize both individual and group performance. Such an idea has recently been the focus in the development of software tools for on-line project management systems at companies such as IBM[8].

Each of the three courses will have a "theme" game that will serve as a foundation around which to build the exercises. These games, taken together, describe three increasingly more complex phases of the social evolution process; hunter-gatherers, tribal society, and state organizations. In the narrative that overarches the three games it is assumed that they represent three stages in the social evolution of a given region. The actual region of interest is the **Valley of Oaxaca, Mexico** for which Reynolds has developed a prototype for each of the three social organizations using real-world data.

Within each of the game frameworks, each student can **exchange control and data structures** that they have produced along with the simulated resources that they have collected. This will provide the basis for the **social fabric** behind the software development process. That is, no one student will be able to generate all of the useful control and data structures in the allotted time. Therefore, they will need to distribute the work among themselves and exchange the results through the laboratory network. Interaction between individuals can be in the form of balanced reciprocal exchange, auctions, and cooperative games. The interaction will take place in a networked lab environment with the games running on PCs and players connected over the local network. We now briefly describe the theme-game and platform for each of the two basic lab courses.

3. THE URBARIUM: AN OVERVIEW

In this section we overview the **Urbarium** games that will be applicable to each of the two courses. The key is to integrate the learning of design and problem solving concepts into games that are appropriate metaphors for their use. Also, the games are connected according to a over-arching narrative. So, one's accomplishment at the end of the previous game will be carried over to the next. This we feel will not only motivate the reuse of concepts, but support the desire for students to "**remain in the game**" and finish their requirement for graduation.

3.1 Games for the CS1 Laboratory

For this course, the "theme" application is hunting and collecting in a virtual world, the Valley of Oaxaca Mexico (we refer to this as game 1). A related application was developed as part of an NSF project, and was used also as a vehicle to introduce high school students to the information necessary for survival in Southwest Colorado between 500 A.D. and 1200 A.D. This has been reported in Scientific American [9] as well as in a special issue of IEEE Transactions on Evolutionary Computation on computer gaming applications [10]. Here the idea is that code segments from several related programs are distributed over the environment. Students take turns moving through the environment and collect different fragments. Since it is unlikely that any student will find all of the pieces themselves, they will need to barter, auction, or exchange pieces. When a student collects all of the code pieces and arrange them correctly then they can get extra code points for visiting cells associated with the program. Figure 1 gives a screen shot of a prototype developed for the NSF project as a tutorial for students taking an Archaeological laboratory at the Crow Canyon Archaeological center in Mesa Verde, Colorado. In section 4 we will discuss the prototype for this game using the MUPPETS framework in the **Urbarium** environment.

Figure 1. 3D snapshot of the game concept illustrating a simulated terrain and realistic visualization.

Another exercise for this course is to provide students with a virtual environment in which each of the students in the lab is a chief, in charge of a growing town or **simulated city** (we refer to this as game 2). The data and graphics used for this application will come from actual archaeological data that describes the emergence early villages in the Valley of Oaxaca Mexico. Students will need to collect information about their site, and use that information to plan for its future growth. Each student can develop and exchange functional classes and control with others in the lab in order to improve their own planning capabilities.

3.2 GAMES FOR THE CS2 LABORATORY

For this course, the **Urbarium** uses an approach inspired by Microsofts' "**Age of Empires**". Here the empire builders at each site now need to interact with competing chiefdoms in the valley by acquiring resources, deploying armies, and engaging in trade relationships. This will involve some basic planning, and problem solving activities. These computational components can be traded and exchanged along with goods and services produced by the modeled towns. A successful programmer will be able to build an empire in the valley and recreate history. A very successful programmer might be able to organize the empire so that they can rewrite history by defeating the Aztec armies. A prototype agent-base simulation for this application was already produced with support from an NSF grant to Reynolds.

All three "theme games" exist presently in prototype form and were initially developed as part of related NSF projects. We plan on developing these games within the context of Muppets, and XNA game studio express. Each game will use the Cultural Algorithm social engine. Cultural Algorithms are a computational mechanism developed by Reynolds to support the evolution of Cultural Systems [11]. Each places the programmer in a social context of increasing complexity, and requires them to use their programming skills to help advance the society. The three games will be tied together in terms of an overarching narrative which will help engage the student. That is, the student will want to take the next course to see how the story is completed, or how they can complete the story on their own. In addition, the gradual introduction of the student to Microsoft gaming software will help to encourage students to consider taking the gaming sequence as an elective. That sequence will also use the same Microsoft developed products, but they will be directed to develop their own extensions.

4. THE HUNTER-GATHERER CS1 GAME

The original prototype for the hunter-gatherer was designed to allow archaeological students to get the "look and feel" of the social system that utilized the area hundreds of years ago. Individuals looked over the landscape for resources. Some of the resources could be gathered by a single individual while other required cooperation and perhaps the exchange of collected resources.

In our case the students are seated at workstations connected via peer to peer network. They take turns moving through the environment. The resources that they are collecting here are syntactic and semantic code components. The current prototype is being developed using the MUPPETS gaming framework. Figure 2 gives a basic example.

The world map is used from the MUPPETS system. The rider object also was created there, in addition to the code objects that should be collected. The callouts show what information should be collected in each cell. These were added outside of the created system to complete the conceptual design. Each question mark in the function inside the belief space represents a statement that the player must find in the game world to complete the function and score more points. As functions are completed, the player can score more points for visiting squares associated with the code. In other cases completing the code allows the player to perform certain functionalities in a cell that can lead to more points.

Some of the syntactic element will have many copies spread throughout the environment, while other code segments may be uniquely placed. This requires the students to exchange collected code segments with each other for points or other code segments. Once a code segment is completed an individual can choose to exchange copies of a complete or partial function with others or keep it for themselves.

At each time step each player in the network moves to a cell, collects what is there. Then there is a pause in which individuals can engage in an exchange dialogue if desired. Otherwise they continue on. The laboratory instructor can keep a trace of the interactions between students in order to assess the degree of social interaction taking place.

In figure 2 just a single algorithm is given, the factorial function shown below:

```
double factorial(int a){
        fact=1;
for(int i=0; i < a)
        fact *= i;
return fact; }
```

It is up to the individual to put the collected components together in the correct fashion. If this is not done, then the code component will execute. Incorrect code segments can be exchanged between individuals, so that all players will need to validate the code that they receive from other. The extent of the validation process will reflect their "trust" of others.

During the course of the term additional relevant code segments that have been studied in lecture can be added to the environment, searched for and exchanged. Thus, the items to be searched for can change over time as the students understanding of the course concepts increases.

5. CONCLUSIONS

In this paper we have described our efforts at integrating a social fabric more explicitly into the design process. Here we use a gaming framework to support the social integration of knowledge in terms, of metaphors, and a continuing storyline that encourages the students to **"stay in the game"** and complete the course sequence. In addition, while individuals can still "win" the game they must cooperate during the code design process in order to do so. Also, since others can make mistakes, a student must be able to validate code that they or others generate.

ACKNOWLEDGMENTS
Our thanks to Mostafa Ali for help in developing the current design of the hunter-gatherer game.

6. REFERENCES

[1]. J.C Thomas, W.A. Kellogg, and T. Erickson, "The Knowledge Management Puzzle: Human and Social Factors in Knowledge Management", *IBM Systems Journal*, **40**, No. 4, 2001.

[2]. E. Arias, H. Eden, G. Fischer, A. Gorfman, and E. Scharff, "Transcending the Individual Human Mind—Creating Shared Understanding Through Collaborative Design," *ACM Transactions on Computer-Human Interaction* **7**, No. 1, 84–113 (2000).

[3]. W. Gordon, *Synectics*, Harper, New York (1961).

[4]. A. S. Gordon, "The Representational Requirements of Strategic Planning," Fifth Symposium on Logical Formalizations of Commonsense Reasoning, New York University (May 20–22, 2001).

[5]. D. Snowden, "The Paradox of Story," *Journal of Scenario and Strategy Planning* **1**, No. 5 (Dec. 1999/Jan. 2000).

[6]. Wayne State University, *Key WSU Facts*, http://www.wayne.edu/keyfacts.html, 2005.

[7]. Smith, D. G., The challenge of diversity: Alienation in the academy and its implications for faculty. Journal on Excellence in College Teaching, 2, 129-137, 1991.

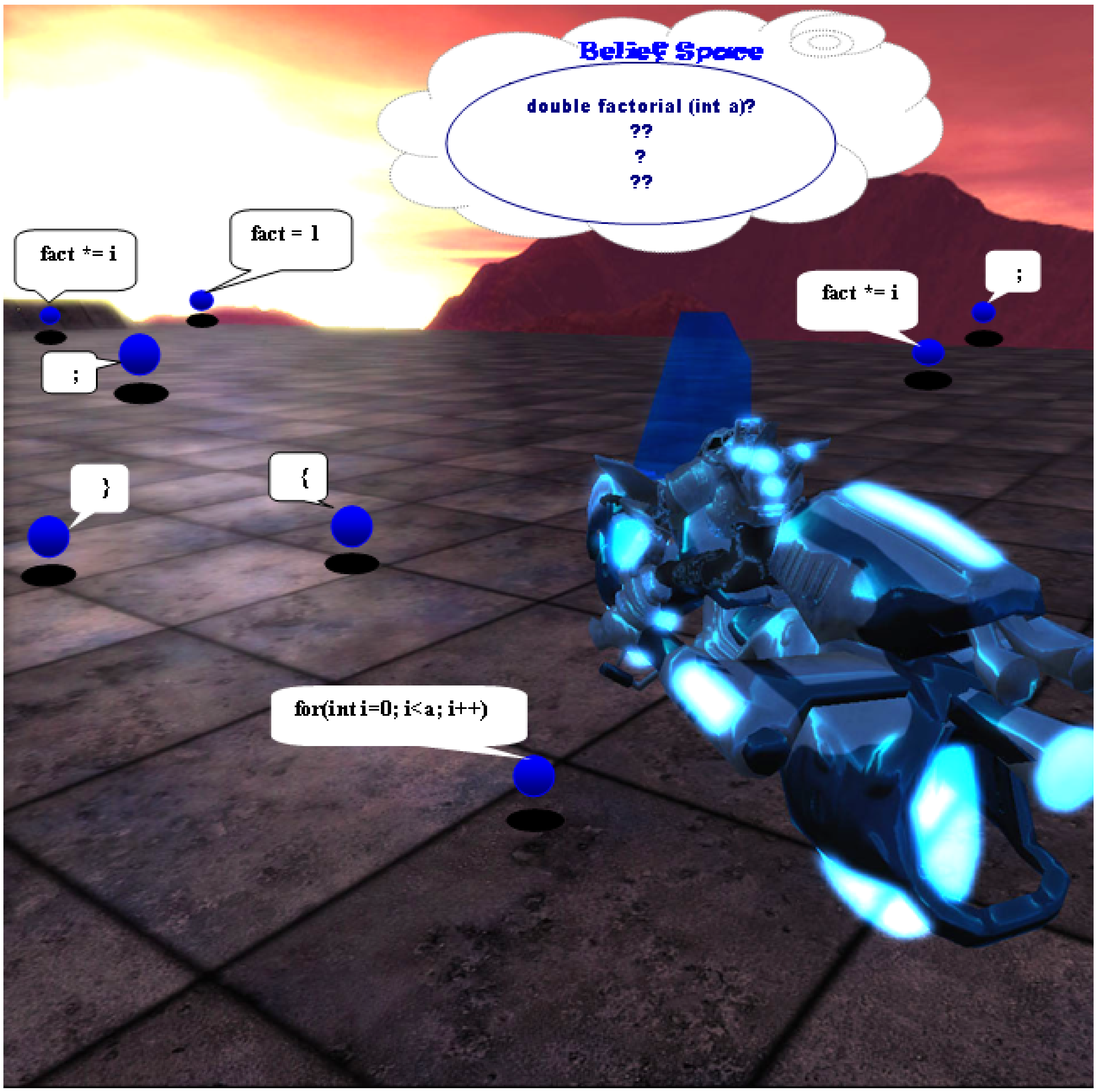
[8]. Cheng, L., Patterson, J., Rohall, S., Hupfer, S., and Ross, S., "Weaving a Social Fabric into Existing Software", IBM Research Report, RCS2385 (W0501-029), January 10, 2005.

[9]. Kohler, T., Gummerman, G., and Reynolds, R. G., "Virtual Archaeology", Scientific American, vol. 293, no. 1, pp: 76-84, July, 2005.

[10]. Reynolds, R.G., Kobti, Z., Kohler, T., and Yap, L., "Unraveling Ancient Mysteries: Re-imagining the Past Using Evolutionary Computation in a Computer Gaming Environment", IEEE Transactions on Evolutionary Computation, Vol. 9, No. 6, pp: 708-720, December, 2005.

[11]. Reynolds, R.G., "An Overview of Cultural Algorithms", New Ideas in Optimization, D. Corne, F. Glover, and M. Dorigo Ed., McGraw Hill Press, 1999, pp: 367-378.

Figure 2: The MUPPETS prototype for the hunter-gatherer game.


Belief Space
double factorial (int a)?
??
?
??
fact *= i
fact = 1
;
}
{
fact *= i
;
for(int i=0; i<a; i++)