Collective Programming: Making End-User Programming (more) Social

Alexander Repenning[1], Navid Ahmadi[2], Nadia Repenning[1], Andri Ioannidou[1], David Webb[3], Krista Marshall[3]
[1]AgentSheets Inc., Boulder, Colorado, [2]University of Lugano, Switzerland, [3]University of Colorado, Boulder, Colorado
{alexander@agentsheets.com, ahmadin@usi.ch, nadia@agentsheets.com, andri@agentsheets.com, dcwebb@colorado.edu, krista.marshall@colorado.edu}

Abstract. The do-it-yourself Web 2.0 culture is quickly creating and sharing more end-user produced content. Gradually moving from static content, such as pictures and text, to interactive content, such as end-user programmed games, the artifacts created and shared have become significantly more sophisticated. The next frontier to make end-user programming more social is to move beyond the current create, upload, share, download, and repeat Web 2.0 models. Collective Programming is a framework that fuses 100% Web-native end-user programming tools with real-time communication mechanisms into a cloud-based multi end-user programming environment. A prototype built, called CyberCollage, enables groups of students to work on game design projects together: they can play multi-user games, change game worlds in real-time, and engage in virtual pair programming.

Keywords: collective programming, end-user pair programming, computers and education.

1. Introduction

The 21st century do-it-yourself Web 2.0 culture is quickly creating more end-user produced content. From sharing static content such as pictures (e.g. Flickr), encyclopedic articles (e.g., Wikipedia) and dynamic content such as movies (e.g. YouTube), end-users are gradually progressing to *interactive* content such as end-user modded [1] games (e.g. LittleBigPlanet) and end-user created programs (e.g. Yahoo pipes). At the same time, improved infrastructure including faster networks, ubiquitous internet connectivity, audio and video capabilities built into basic computers enables *real-time* communication of potentially large numbers of end-users to create sophisticated computational artifacts such as games and simulations. Already, the combination of tools such as screen sharing, chat, voice over IP, and shared white boards is facilitating new kinds of collaboration including training, design and research at great distances.

CyberCollage is a first-of-a-kind real-time end-user development environment for creating interactive content. Implemented as a cloud-based Web application,

CyberCollage enables a new form of social end-user development that we call *Collective Programming*. At the content level, CyberCollage allows end users to build sophisticated computation artifacts such as games and simulations using drag and drop visual programming approaches developed previously in AgentSheets [2], AgentCubes [3] and numerous other educational programming environments including Alice [4], Scratch [5], and Squeak Etoys [6]. An important contribution of CyberCollage over these previous systems is that the entire programming and runtime environment is 100% browser based and built with Web-native HTML 5 technologies. Even more important for Collective Programming, however, is the real-time communication framework built into CyberCollage allowing groups of end-user programmers to collaborate concurrently on projects shared in the cloud. Imagine, for instance, two end users collaborating on a Frogger-like video game (Fig. 1). They can simultaneously play the game, e.g., have a two-frog race, change the game world, or modify the game behavior.

The real-time communication aspect of Collective Programming is essential for enabling a new kind of social end-user programming that we believe to be especially useful in educational settings. The challenge is not just to send more information faster, but also to maintain the perception of simultaneity among a group of collaborators. For instance, if Tim and Victoria are using or modifying their game (Fig. 1) their perception of the interaction should be as though they were sitting next to each other in the physical world. Significant motivational and peer learning [7] benefits of pair programming [8] have been documented. CyberCollage could be considered a virtual pair-programming environment that enables end users to program together remotely.

2. CyberCollage: real-time collaborative end-user programming

CyberCollage enables real-time collaboration through a combination of formal and informal communication. At the formal level, participants share a common artifact (game or simulation) consisting of media (images), programs (agent behaviors), and game/simulation worlds (worksheets) and communicate through the exchange of the modified pieces of the artifact. Informally, participants communicate through online communication mechanisms such as chat, voice or video. They also communicate through awareness interfaces that keep them informed about other participant actions.

CyberCollage significantly advances the state of real-time social interfaces in ways that are probably best explained through a scenario. Imagine that two students, Tim and Victoria, are working on a joint Frogger-like game (Fig. 1). Tim wants to work on the frog whereas Victoria is eager to build the road and the truck. They start a new project and work on their respective game objects in real time. Each person has a *focus* defined by what they are working on and a *peripheral vision* providing social information about what others are doing. Even with Victoria's focus on programming the truck, a *presence (or awareness) interface* allows her to perceive that Tim is drawing the Frog. This presence information may trigger the need, or opportunity, to engage in additional communication. They will not only be able to see how the other

person doing, but will also be able to take control and contribute in real time. That is, Victoria can touch up Tim's frog image and Tim can help Victoria program the truck.

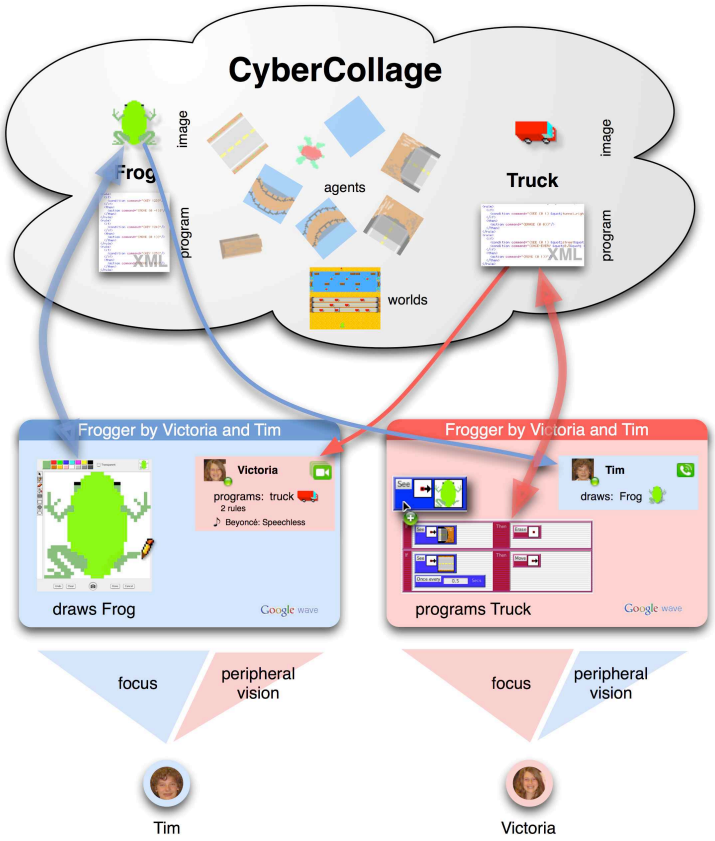

Figure 1: Collective Programming = Real-Time Communication + Rich Interactive Content + Web-based End-User Programming. Tim and Victoria collectively author a Frogger game. Tim draws the frog, Victoria programs the truck. Through peripheral vision implemented as presence interface they can track what the other is doing and interact in real time.

Technically speaking, CyberCollage is a cloud-based integrated development environment (IDE) with client and server side components. On the front-end, Javascript clients provide end users with an HTML 5-based visual programming environment, operating inside a Web browser. The IDE lets users create agents, draw depictions, program the agent behavior in a visual programming environment, and execute the agents in the worksheet. On the back-end, a communication component synchronizes multiple clients working on the same project. While users interact with their programming environment individually, the client sends updates to the server. The updates include changes in agent depictions, program updates, and the worksheet modifications. The server collects the updates from each client and broadcasts them to all other clients. Each client has a dispatching component that fetches the updates from the server and dispatches them to the IDE.

3. Related Work

Fig. 2 depicts a two dimensional space of collaboration models. Horizontally, the Use⇔Design continuum captures a number of points, including just using a finished artifact, changing it (modding), and programming one from scratch. Vertically, an essential distinction is made between real-time, synchronous collaboration and offline, asynchronous interaction.

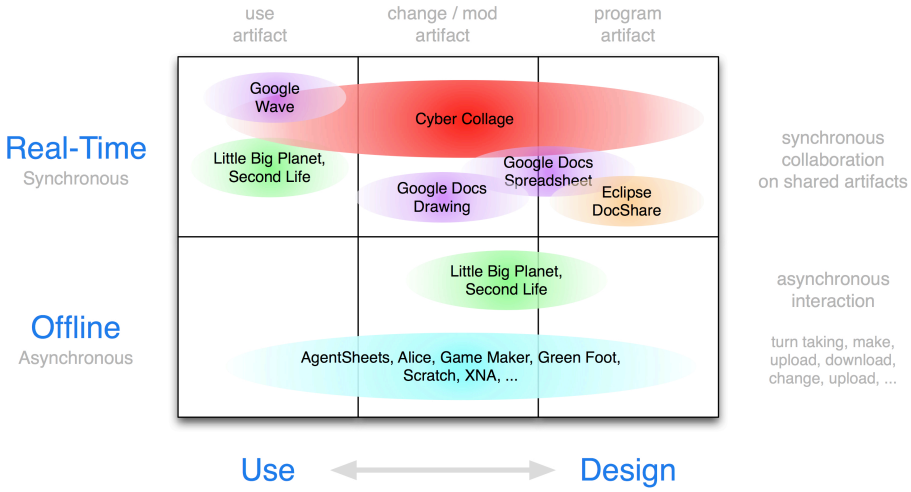

Figure 2: Collaboration Models

Offline collaboration models are based on the asynchronous interactions of participants to use, mod, or design games and simulations. A typical interaction at this level may start with one person making a game, uploading it to a repository and then having a different person play the game. This, in turn, may interest the second person to change the game world or even to change the game behavior. This would typically require downloading the game, assuming its source is available, modifying it, and then perhaps uploading the new version back to the repository.

The synchronous nature of real-time collaboration is radically different from the asynchronous type of offline interaction. In this model people are working together on shared artifacts. The real-time character of this approach requires more sophisticated client/server architectures providing collaborators a sense of presence. Who is online? Who is working on what and how? If one person is changing something then this change needs to be broadcasted to all other users as fast as possible. The enormous degree of interactivity may require access control, e.g., file locking, and interaction protocols such as turn taking to avoid potential conflict. Artifacts are shared.

CyberCollage is a real-time collaboration framework covering the entire Use⇔Design continuum. With it end users can use games and simulations, change them, and program new ones collectively. CyberCollage allows, theoretically, any number of users to participate in all aspects of the Use⇔Design continuum. Conceptually speaking, CyberCollage combines and extends ideas found in frameworks such as

Google Wave, Google Docs Drawing and Google Docs Spreadsheet. Combining the ideas from Google Wave and Google Docs Drawing two users could collaboratively build a chess game by creating all the pieces, drawing a chess board, placing the pieces onto the board and moving them on the board taking turns. However, the ability to program the game would allow them to quickly advance from the collaborative drawing to implementing multi-user games or simulations.

4. Evaluation

To evaluate CyberCollage as an environment for real-time collaborative design, education researchers from the University of Colorado conducted a pilot study. Three sessions involving middle grades students with prior experience with AgentSheets were organized to document student interaction and assess end-user feasibility.

In one activity, two boys started with a highly structured, competitive programming activity. However, they soon expressed that they did not like to compete but collaborate. Their interaction organically evolved into a side-by-side game design session. The pair immediately began adding, drawing, and programming agents, creating a sophisticated two-player 'good versus evil' game. The students became so engaged in their game creation that they continued working on it after class and at home. Once their game was functional, the communication between the pair shifted from collaborative design to negotiation of agent behaviors. Communication occurred through verbal interaction as well as through observing agent behavior in the programming environment and in successive rounds of game play. For example, when one student observed that his partner's agent was immune to his "laser beam", he adapted his agent's behavior to gain an advantage. Eventually, the collaborative design session morphed again into a competitive programming arms race in which two boys pitted strategic programming methods against each other and, at times, re-negotiated the rules of the game.

Another activity involved five students – two from Colorado and three from Wyoming – communicating over a Skype video connection while using CyberCollage. Students were encouraged to design a Frogger-like game. With five students involved, the group required leadership and some negotiation of roles [9]. Over a period of 90 minutes we observed an emergent participatory structure in which students communicated verbally, as needed, but found ample information available on their collective workspace as new agents were designed, positioned, and programmed by group members [10]. Tasks were initially assigned by one of the girls from Wyoming, but over time all students provided input on additions, deletions and adaptations of the game objects.

While space limitations preclude a more detailed summary of the analysis of student participation and collaboration, the initial pilot of CyberCollage shows promise as an environment for real-time design for both local and distant collaborators. Real-time, synchronous design and programming were accessible and engaging for middle grades students, demonstrating that they could collaboratively develop a working game in one to two hours.

5. Acknowledgements

This work was supported by the National Science Foundation (grant number IIP-1014249). Opinions expressed are those of the authors and not necessarily those of the National Science Foundation.

6. References

[1] Magy Seif El-Nasr and Brian K. Smith. 2006. Learning through game modding. Computers in Entertainment 4(1), Article 7 (January 2006).

[2] Repenning, A. and Ambach, J. "Tactile Programming: A Unified Manipulation Paradigm Supporting Program Comprehension, Composition and Sharing," in Proceedings of the 1996 IEEE Symposium of Visual Languages, Boulder, CO, 1996, pp. 102-109.

[3] Ioannidou, A., Repenning, A., and Webb, D. "AgentCubes: Incremental 3D End-User Development," Journal of Visual Languages and Computing, Special Issue on Best Papers from VL/HCC2008, 20, (4) pp. 236-251, 2009.

[4] Moskal, B., Lurie, D., and Cooper, S. "Evaluating the effectiveness of a new instructional approach," in Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Norfolk, VA, USA: ACM, 2004.

[5] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y. (2009). Scratch: Programming for All. Communications of the ACM 52(11).

[6] Squeakland, Home of Squeak Etoys: http://www.squeakland.org/

[7] Wills, C.E., Deremer, D., McCauley, R.A., & Null, L. (1999). Studying the use of peer learning in the introductory computer science curriculum. Computer Science Education, 9, 71–88.

[8] Williams et al. Strengthening the case for pair programming. Software, IEEE (2000) vol. 17 (4) pp. 19-25.

[9] Fuchs, L. S., Fuchs, D., Kazdan, S., Karns, K., Calhoon, M. B., Hamlett, C. L., & Hewlett, S. (2000) Effects of workgroup structure and size on student productivity during collaborative work on complex tasks. Elementary School Journal, 100(3), 183-212.

[10] De Laat, M., Lally, V., Lipponen, L., & Simons, R-J. (2007). Investigating patterns of interaction in networked learning and computer-supported collaborative learning: A role for Social Network Analysis. Computer-Supported Collaborative Learning, 2, 87-103.