

This application demonstrates the natural way in which event sharing (FIG. 7) and state sharing (FIG. 10) can be combined. Note that the bidding and notification teams 230 could have been implemented with shared object 650, by having a shared bid object and notification object between 5

Drawing Board

This application allows the users to share a drawing board on which they can draw and edit lines and curves. The lines and curves are drawn by pressing down on the left-mouse button, dragging it to the end point (or along a curve), and releasing the button. A shared object 650 is maintained for each line or curve, where the line is represented as two points and the curve as a list of points. The editing operations permitted on objects are translation, and changing the end-points (only for lines). 10 15

The application uses the conflict detection support (FIGS. 13, 14, 15, 16) provided by this invention to detect concurrent updates on the same object 650. It then allows the user to clone and export the complete set of objects to others (FIGS. 10–11). After deciding on the correct state, re-initialization of the complete set is done. 20

In addition, it uses Late Event Modification (FIG. 12) to give the user feedback while drawing or translating a line or curve. Consider the case of Alice drawing a line. The initial mouse press fixes the initial point of the line and adds the line object to the set. Though the final point is not fixed till Alice releases the button, she wants to see the current position of the line while dragging the mouse. However, others do not need to see each of these mouse drag events. LEM allows us to solve this problem by posting an updateEvent 810 with the current final point and then continuously modifying the final point value in the event object as Alice drags the mouse. The done method 1231 is called on the event when the mouse button is released. 30 35

This application can be enhanced in two ways. First, instead of cloning the complete set, it could clone the object 650 in error and the objects in its physical neighborhood. After users fix the errors, re-initialization is done for the single object in error. 40

Thus, while we have described our preferred embodiments of our invention, with alternatives, it will be understood that those skilled in the art, both now and in the future, may implement various equivalents, improvements and enhancements which fall within the scope of the claims which follow. These claims should be construed to maintain the proper protection for the invention first disclosed. 45

BIBLIOGRAPHY

The following background references are hereby incorporated herein by reference in their entirety: 50

- [1] D. Comer and D. Stevens. Internetworking with TCP/IP. Prentice Hall.
- [2] J. F. Patterson, R. D. Hill, S. L. Rohall, and W. S. Meeks, "Rendezvous: An architecture for Synchronous Multi-user Applications." CSCW 90: Proceedings of the Conference on Computer Supported Cooperative Work, ACM, 1990.
- [3] Sun Microsystems, Java Development Kit (JDK). <http://www.javasoft.com>.
- [4] K. Birman, A Schiper and P. Stephenson. "Lightweight causal and atomic group multicast". ACM TOCS, 9(3):272–314, August 1991.

Having thus described our invention, what we claim as new and desire to secure by Letters Patent is: 55 60

1. A system for updating replicated state in a distributed collaborative application, comprising:

a plurality of clients, each client comprising:

the distributed collaborative application comprising one or more distributable components, wherein each component may be executed independently and communicate with other components; each component including one or more sets of related shared objects which are replicated at collaborating applications; and

a collaborative client middleware, communicatively coupled to the application and to the network, adapted to communicate to and receive updates over the network to the state of a shared object;

conflict detection means for detecting conflicting updates to the shared object; and

conflict resolution means for resolving said conflicts, detachably coupled to the conflict detection means.

2. The system of claim 1, further comprising update event means for asynchronously and atomically updating one or more of said related shared objects.

3. The system of claim 1, further comprising add object means for adding an object to a local set replica of said shared objects and communicating an added object state to collaborating applications.

4. The system of claim 1, further comprising means for instantiating a collaborative team.

5. The system of claim 4 wherein the system includes a client-server system, further comprising:

an optimistic event execution model; and

team descriptor logic means for ordering events through the server and dispatching locally generated events immediately.

6. The system of claim 1, wherein the system is a client-server system, further comprising:

a server coupled to the network, the server comprising means for maintaining a collaborative activity for each active collaboration.

7. The system of claim 6, the server further comprising: team server means for dynamically creating, updating and destroying a collaborative team, wherein each team has a unique name space in a collaborative activity.

8. The system of claim 6, the server further comprising team policy means for defining and implementing a team behavior.

9. The system of claim 1, further comprising context-sensitive state marhalling means for propagating state to one or more new or updated shared objects in the shared object sets.

10. The system of claim 1, further comprising cloning and reinitialization means for correcting diverging state among the shared object sets.

11. The system of claim 10, further comprising: context-sensitive state marhalling means for marshaling and unmarshaling clones of one or more shared objects in the shared object sets.

12. The system of claim 10, wherein said cloning and reinitializing means further comprises:

clone subsetting means for making a clone of a subset of objects in the set and preserving pointers from inside the original subsets to objects outside the cloned subset; means for mapping references between objects inside the original subset into the cloned subset;

means for exporting the cloned subset to other clients; and means for reinitializing the subset of objects based one of the cloned subsets.

13. The system of claim 1, wherein the events are the units of communication in the collaborative client, and each event has a name, a type, data, and a source.