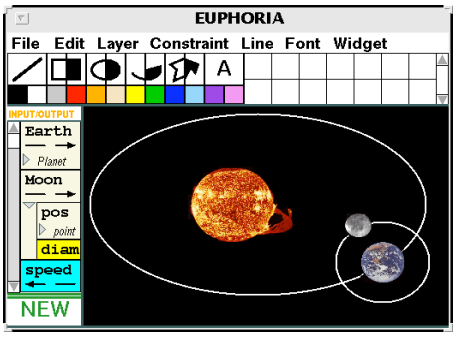
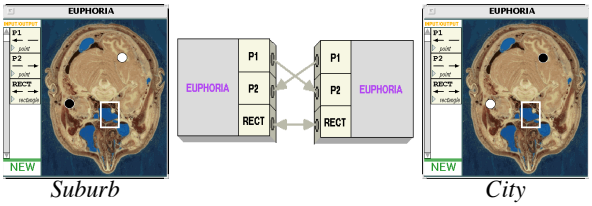
PROCESS CONTROL SIMULATION

The process control application consists of three modules controlling production of maple syrup in a factory (see previous figures). The Sensor module monitors conditions in the factory such as the syrup temperature. The Control module makes decisions, such as adjusting the incoming sap flow, based on sensor values. A display shows an interactive, animated display of the factory. The display is drawn in EUPHORIA, utilizing user-defined widgets for each display component. End-users configure the communication among these modules at runtime by drawing connections between the published variables. Note that the FIRE and FLOW connections between the Control and EUPHORIA modules are bidirectional, allowing user interaction in the display to override decisions of the control module.



PLANETARY ORBITS ANIMATION

The planetary orbits application consists of three communicating modules: Earth, Moon, and an animated graphical display. The display is created in EUPHORIA with user-defined types (e.g., *Planet* type above). Constraints link a speed variable to the Sun's size, enabling users to interactively adjust the simulation rate.



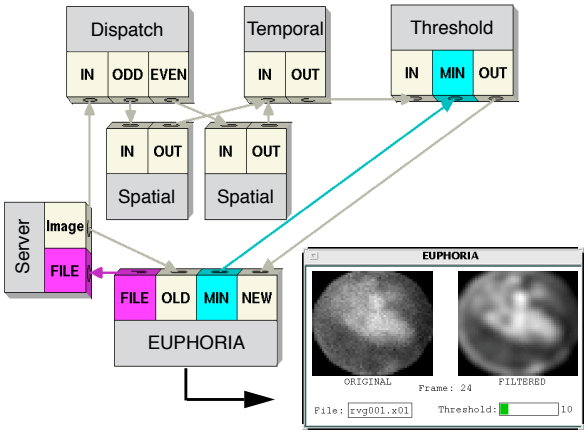
Suburb *City*

SHARED CURSORS

EUPHORIA can be used as a synchronized image display with shared cursors. A typical use of shared cursors is medical imaging, where physicians at different locations (e.g., a city and its suburb) want to discuss a medical image. Each physician has a graphical cursor which is moved over the image while talking about areas of interest. This multi-user GUI requires no programming; cursors are drawn in EUPHORIA and are linked through the use of graphical constraints and module connections. A rectangular "region of interest" can also be drawn and synchronized interactively.

VIDEO TELECONFERENCING

An ATM-based video conferencing application featuring 30 fps NTSC-quality video, simultaneous multiparty conferences, voice activated audio bridging, and "video follows voice" switching has been constructed.



MEDICAL IMAGE PROCESSING PIPELINE

A nuclear medicine radioactive blood pool study is used to create movies of the human heart for diagnostic purposes. Each movie consists of a series of pixmap images. The images suffer from noise caused by ambient radiation, making them difficult to read. This noise can be reduced by digitally filtering the images.

In this example, the digital filtering operation has three stages. The images of the movie are filtered *spatially* (each pixel is processed in terms of surrounding pixels) and *temporally* (each pixel is processed in terms of corresponding pixels in previous and following images). A *threshold* operation then zeroes pixels below a given value.

An interactive filtering application is created through the use of a number of communicating modules (see above). Spatial and Temporal filter modules are created independently, each having input and output image published variables. However, the spatial filtering operation is computationally intensive, requiring 61,236 floating point operations per frame. For this reason, a Dispatch module is used to distribute the images between two identical Spatial modules running concurrently on separate processors. The spatially filtered images are merged at the Temporal module, filtered, and sent to the Threshold module. EUPHORIA is used to control the filtering operation; users specify the movie name and threshold value. The original and filtered movies are also displayed within EUPHORIA.

Future work includes support for module encapsulation. For example, this will allow a user to create a single "filter" module containing the Dispatch, Temporal, and Spatial modules.

ACKNOWLEDGMENTS

We thank Jyoti Parwatikar and John DeHart for their teleconferencing work. This research was supported in part by NSF grants CCR-91-10029 and CCR-94-12711.

REFERENCES

1. Kenneth J. Goldman, Bala Swaminathan, T. Paul McCartney, Michael D. Anderson, and Ram Sethuraman, The Programmers' Playground: I/O Abstraction for User-Configurable Distributed Applications. *IEEE Transactions on Software Engineering*. To appear.

2. T. Paul McCartney and Kenneth J. Goldman. Visual Specification of Interprocess and Intraprocess Communication. In *Proceedings of the 10th International Symposium on Visual Languages*, Oct. 1994, pp. 80-87.

The Programmers' Playground: A Demonstration

Kenneth J. Goldman, T. Paul McCartney, Ram Sethuraman, Bala Swaminathan
Department of Computer Science
Washington University
St. Louis, Missouri, 63130
{kjg, paul, ram, bs}@cs.wustl.edu
http://www.cs.wustl.edu/cs/playground

ABSTRACT

We demonstrate The Programmers' Playground, a programming environment that supports end-user construction of distributed multimedia applications. Features include separation of communication and computation, dynamic end-user configuration of distributed applications, separation of active & reactive control, and migration of running modules. Applications include a process control simulation, planetary orbits visualization, medical image processing pipeline, a multimedia conferencing application, and a user interface management system for constructing customized graphical user interfaces (GUIs).

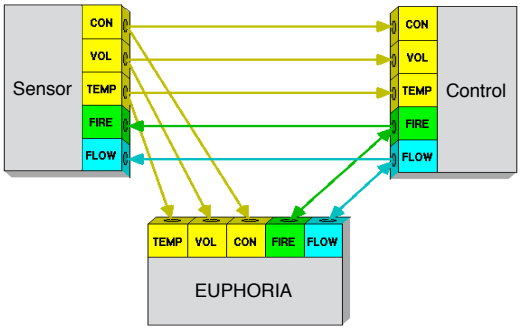
KEYWORDS: direct manipulation, distributed computing, graphical user interfaces, multimedia, reconfiguration, user interface management system

INTRODUCTION

The Programmers' Playground [1] is a software library and runtime system for creating distributed multimedia applications. Playground is based on *I/O abstraction*, a new programming model for distributed systems that provides a separation of communication and computation. A distributed application consists of a set of communicating *modules*, written as C++ programs, using special data types from the Playground library. Variables declared using these data types can be *published*, making information available to external modules. The communication structure among the modules of a Playground application is defined by a set of logical connections among published variables of the modules. This communication structure is configured dynamically at runtime through a graphical user interface for the connection management system.

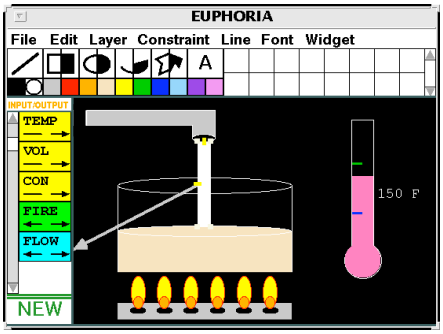
Communication among modules of a distributed system occurs implicitly; when a published variable is modified, the new value of the variable is automatically sent to its connected variables. In this way, the programmer does not need to know the low level details of how the communication is actually achieved. Each module can be created independently of the modules with which it communicates.

This paper describes Playground's graphical tools and typical applications constructed using Playground.



GRAPHICAL CONFIGURATION OF MODULES

Active software modules of a distributed system are viewed through the use of a connection manager graphical front-end. Modules are represented as boxes with each published variable represented as a color-coded "plug." End-users configure the module communication graphically by drawing connection arrows between published variables.



END-USER CONSTRUCTION OF GUIs

EUPHORIA, Playground's user interface management system [2], is a specialized module for creating customized direct manipulation GUIs without the need to write user interface source code. In EUPHORIA, end-users simply draw GUIs using a graphics editor. Attributes of a user GUI are selectively published, to expose their state to other Playground modules. For example, in the figure above, the height of the thermometer's mercury (i.e., temperature) may be published. Interaction with the GUI causes attribute information to be sent out to external modules. Similarly, changes to the states of modules connected to a user GUI can cause its appearance to change.

A number of advanced features are supported in EUPHORIA including constraint-based editing, imaginary objects, widgets, alternatives, and user definable data types. Aggregate mappings are planned, allowing visualization of aggregates through mapping rules.