NeWS Applications

9/11/87

Agenda

NeWS Terminology and Concepts

Designing NeWS Applications

Application Case Studies

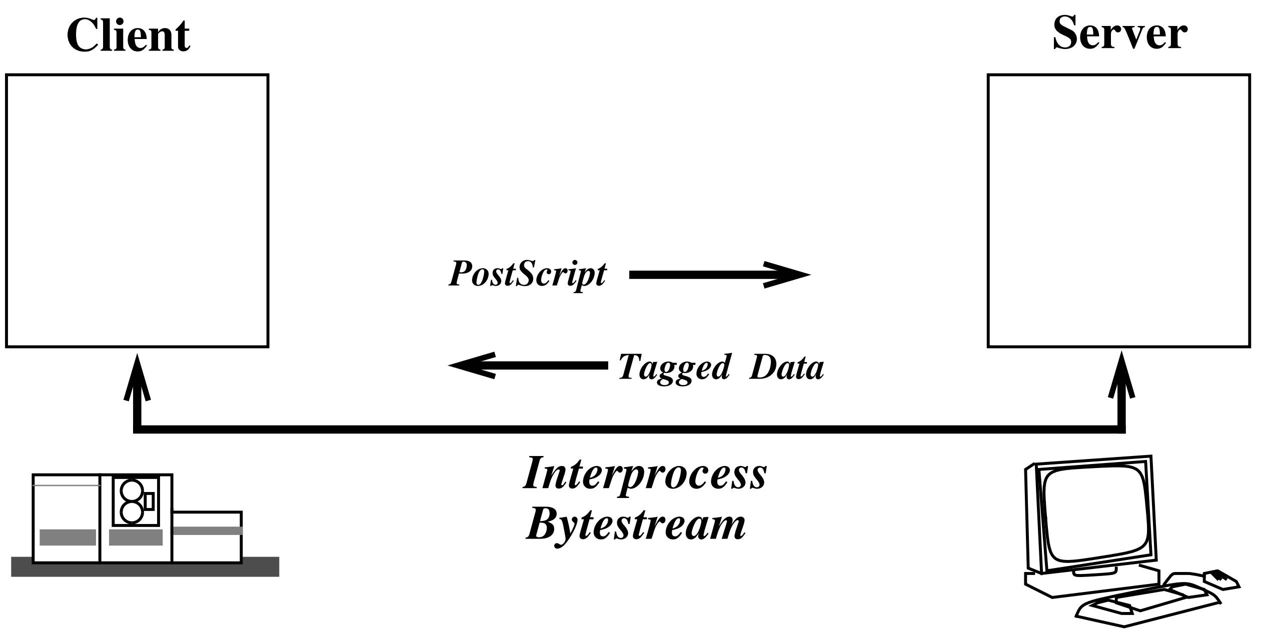
- **Existing Applications**
- **Emulating Other Window Systems**
- **Server-Based Applications**
- **New Applications**

Implementation Hints

- **Converting From A RasterOp Model**
- **Application Debugging and Performance Improvement**

NeWS Terminology and Concepts

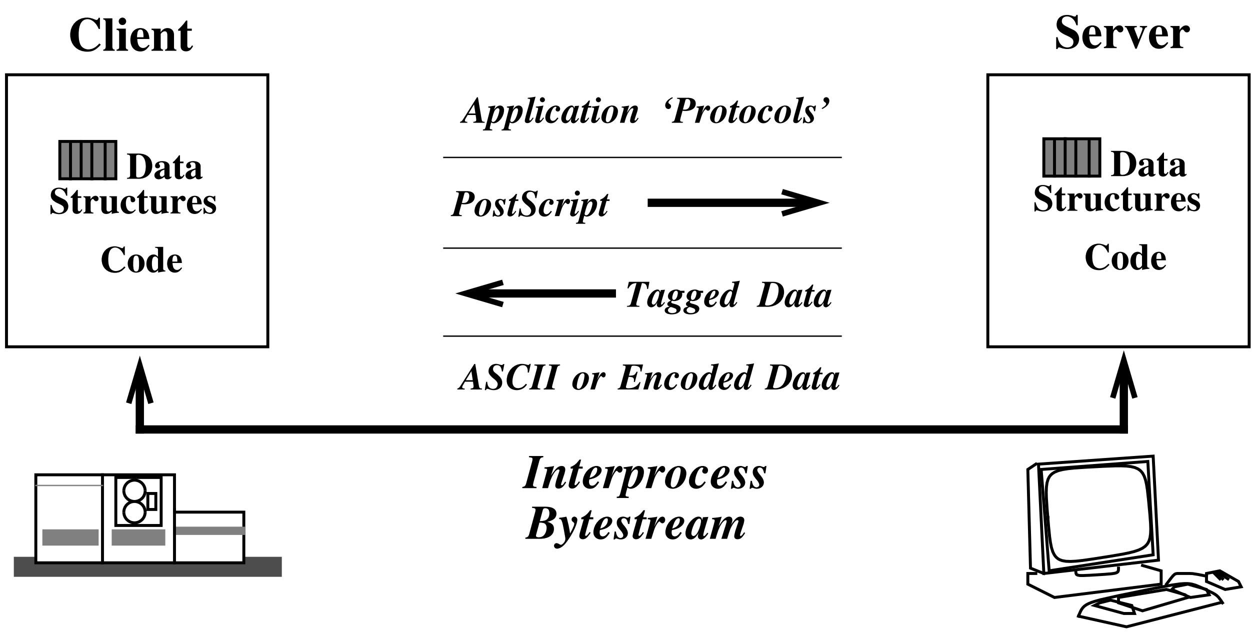
- **Network Window Service**
  - **'Server' process provides a display service**
    - **Controls the screen, keyboard, and mouse**
  - **'Client' process makes use of this service**
  - **Client and Server communicate over bidirectional bytestream**
    - **Use local IPC when on same machine**
    - **Use network when on different machines**
- **NeWS Server Understands PostScript**
  - **Language interpreter**
    - **Direct execution**
    - **Invocation of previously loaded subroutines**
- **Server Returns Tagged Data to Client**
  - **Message id tag**
  - **Data values**



Designing NeWS Applications

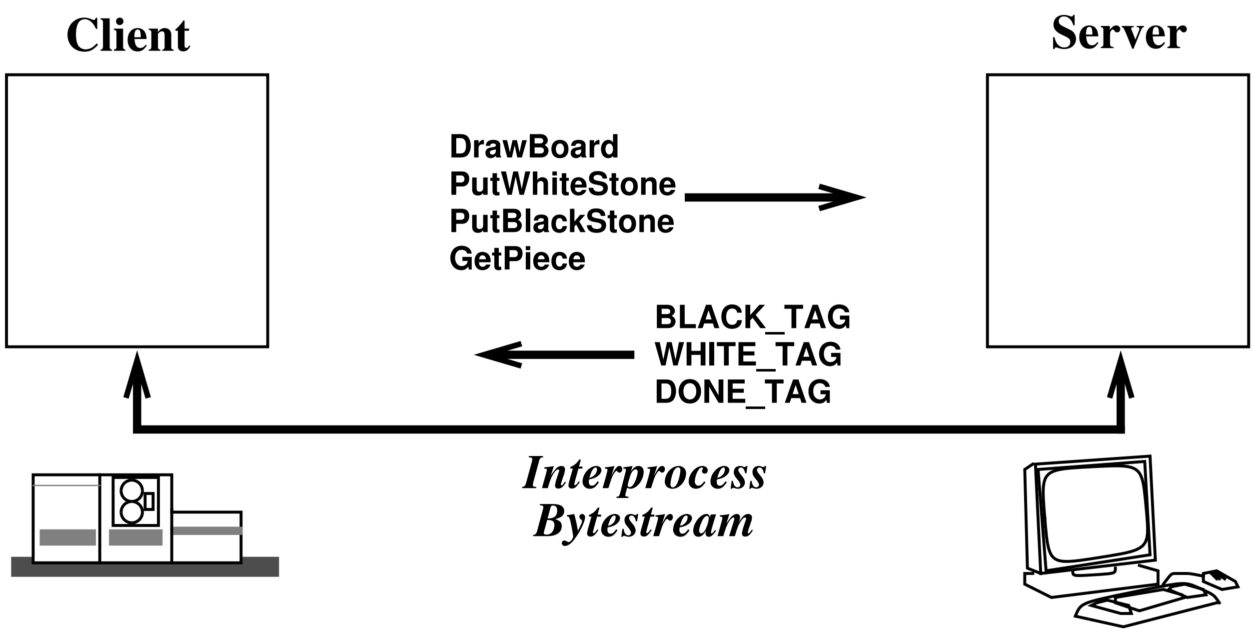
- **Application Partioning Options**
  - **Client side only**
  - **Server side only**
  - **Both Client and Server**

- **Factors in Designing NeWS Applications**
  - **Code/Data partioning between client and server**
    - **Interactive vs. Non-interactive parts of an application**
    - **Network bandwidth and limitations**
  - **Data structure synchronization**
  - **Bytestream encoding**
  - **Client/Server interface specification (Application ‘Protocols’)**
    - **Client → Server: PostScript subroutine names and parameters**
    - **Server → Client: Tag values and data format**



Application Protocol Example

- **Go Board**
- **Client → Server**
  - **DrawBoard()**
  - **PutWhiteStone(x, y)**
  - **PutBlackStone(x, y)**
  - **GetPiece(x,y)**
- **Server → Client**
  - **BLACK_TAG x y**
  - **WHITE_TAG x y**
  - **DONE_TAG**



Application Case Studies

Existing Applications

- **Many Existing Applications Can Take Advantage of NeWS With Little or No Modification**

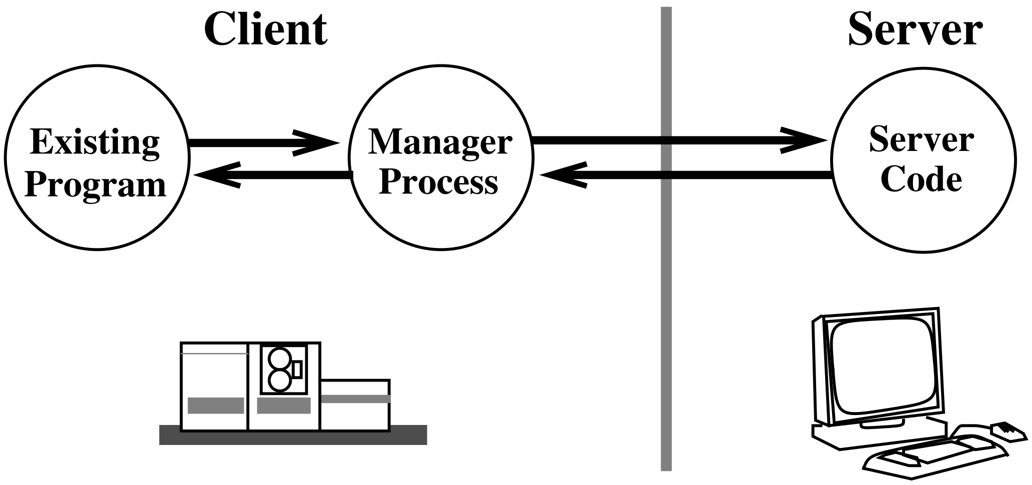
- **Output-Oriented Applications**

  (e.g. chart generation, performance meters, graph drawing)

  - **Those that already generate PostScript require no modification**
    - **Terminal emulator for input, NeWS for output**
    - **Result: application gets screen display for free!**
  - **Non-PostScript applications require an output translation filter**
    - **Filter could be client/server based, or entirely in the server**

- **Interactive Applications**

  - **NeWS handles both input and output**
  - **Client-side application-manager process is used**
    - **Manager acts as an intermediary between application and NeWS**
    - **Good performance because server handles low-level user interaction**



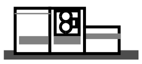
GNUPlot & SURFMODL

- **PostScript Output-Oriented Applications**
- **Public Domain Programs**
  - **GNUPlot - Interactive data plotting program**
  - **SURFMODL - 3-D Surface modeler**
- **Los Alamos PostScript Window Interface**
  - **Displays PostScript output stream in a NeWS window**
  - **Uses NeWS menus to provide:**
    - **Zoom/Restore**
    - **Printing**
    - **Overlays**
  - **Simple NeWS driver; 30 lines of C, 400 lines of PostScript**

Alliant Performance Monitor

- **Non-PostScript Output-Oriented Application**
- **Dynamic Color Display of Alliant Processor Utilization**
- **Written in Two Weeks, Including Learning PostScript!**
- **Implementation**
  - **Server Side**

    - **Processor objects which knows how to display themselves created**
    - **One processor object for each real Alliant processor**
    - **set_processor_value routine tells a processor to change value**
  - **Client Side**

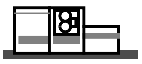


    - **Parses output of existing performance monitoring tool**
    - **Sends set_processor_value call to server for each CPU**
  - **All graphics scaling is performed automatically by NeWS**
  - **Server code moved to Silicon Graphics machine without recompiling**
  - **300 lines of C, 800 lines of PostScript**

Backgammon

- **Interactive Application**
- **Computerized Backgammon Game on NeWS**
  - **Fast interactive response**
  - **Animates computer moves along curved path**
  - **Looks and feels like the real thing**
- **Uses Existing Berkeley Backgammon Program**
  - **Now runs in the background; NeWS handles all user interface**
  - **Berkeley backgammon still thinks it is talking to human!**
- **Implementation**
  - **Server side**
    - **Board, dice, pieces, and point areas are separate canvases**

    - **Each piece has own event manager process to handle movement**
    - **Understands high-level messages (e.g. MovePiece, DrawDice)**



  - **Client Side Manager Process**
    - **Translates between what existing program expects and NeWS**
  - **1400 lines of C, 1800 lines of PostScript**

Existing Application Issues

- **Does the application currently output PostScript?**
- **Can the application be added to? Modified?**
- **What is the cost/benefit of using an existing application vs. creating a new one?**

Emulating Other Window Systems

"New Technology is First Applied To Emulate Current Technology"
Marshall McLuhan

- **Window System Emulation Strategy**
  - **Typical window system interface is set of library routines**
  - **Create new interface libraries that work with NeWS**
  - **Relink applications**
- **Window System Emulation Requirements**
  - **Programmatic interface to window system remains unchanged**
  - **Applications looks and feels the same**
  - **Performance is acceptable**
- **NeWS is an Excellent System For Emulation**
  - **Option of moving part of window system to server available**
  - **Flexible, programmable input processing**
  - **Powerful graphics model**

GEM Emulation

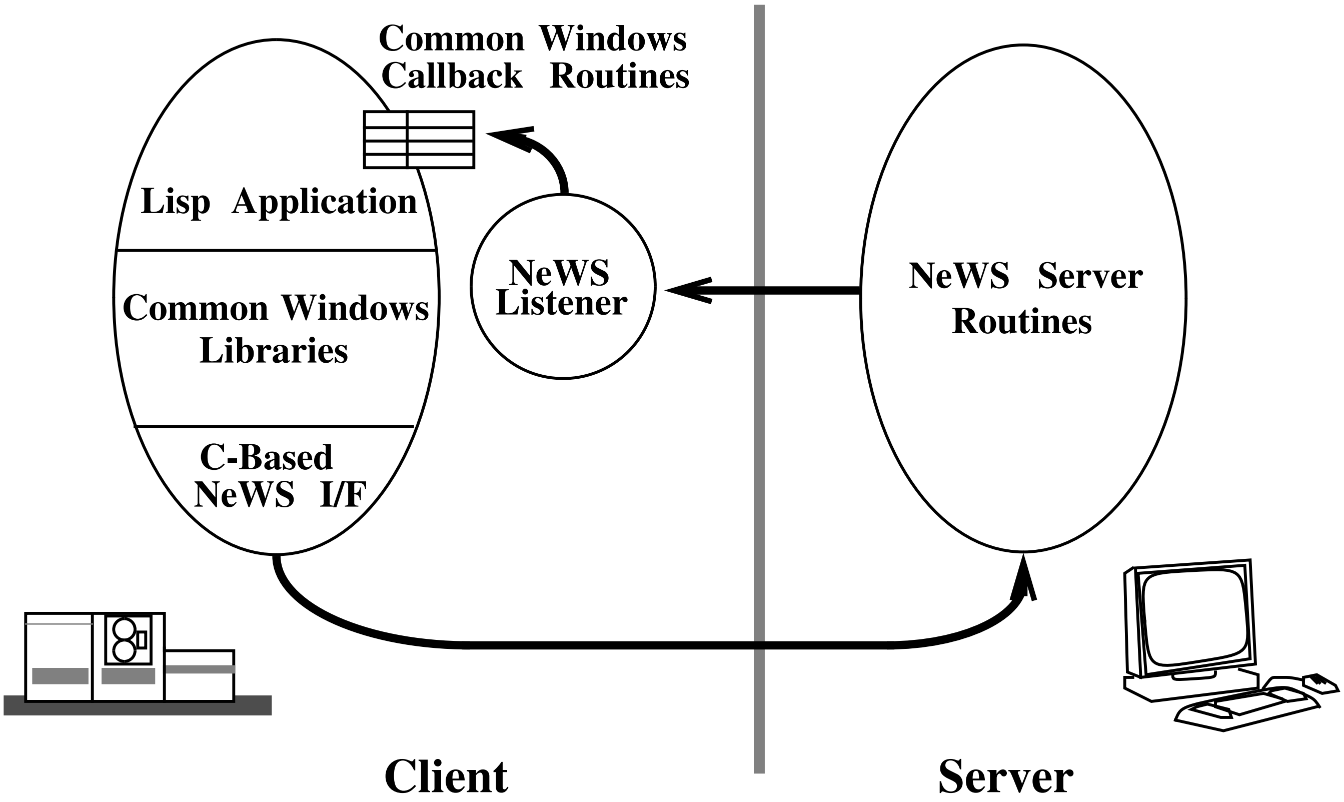
- **Gem is an Atari and IBM PC Window System**
  - **One application at a time**
  - **Pop-up desk accessory windows**
- **Programmer Interface**
  - **C libraries**
  - **AES — Window and menu manager**
  - **VDI — Drawing and graphics**
  - **Applications responsible for damage repair and window movement**
- **Implementation**
  - **PostScript routine in the server for each GEM library call**
  - **Window attributes stored on both client and server for performance**
  - **Client updates attributes on both sides**
  - **2900 lines of C, 300 lines of PostScript**

Symbolic Programming Environment (SPE)

- **SPE is a Window-Based AI Development System**
- **Common Windows: A Standard Lisp Window System**
  - **Common Windows is a proposed standard for Common Lisp**
  - **Specifies Lisp programmatic interface only**
  - **Provides high-level constructs (e.g. scrollbars, selection areas)**
- **SPE Implements Common Windows on NeWS**
  - **Network-based Common Windows big win for compute-bound AI**
  - **NeWS minimization of client/server traffic critical for SPE**
  - **Window system look and feel can be changed dynamically**
  - **Extensions to Common Windows:**
    - **LispScript: Lisp to PostScript package**
    - **NeWS functionality (e.g. color, graphics scaling and rotation)**

NeWS Common Windows Implementation

- **Server Side**
  - A PostScript routine for each Common Windows library routine
  - NeWS events passed to client with id of window were they occurred
- **Client Side**
  - NeWS Listener is interrupt-level code inside Lisp system
  - Listener decides which callback routine to invoke from window id
  - Identical window system data structures on both client and server; client makes changes and synchronously propagates them to server
- **Lines of Code**
  - 3700 lines of PostScript; 18,100 lines of Lisp; 1200 lines of C



Window System Emulation Issues

- **Is target system defined at a high enough level?**
- **Can the NeWS graphics emulate the target system?**
- **What is client/server functionality split?**
- **How should data be split and managed?**
- **How can performance be optimized?**

Server-Based Applications

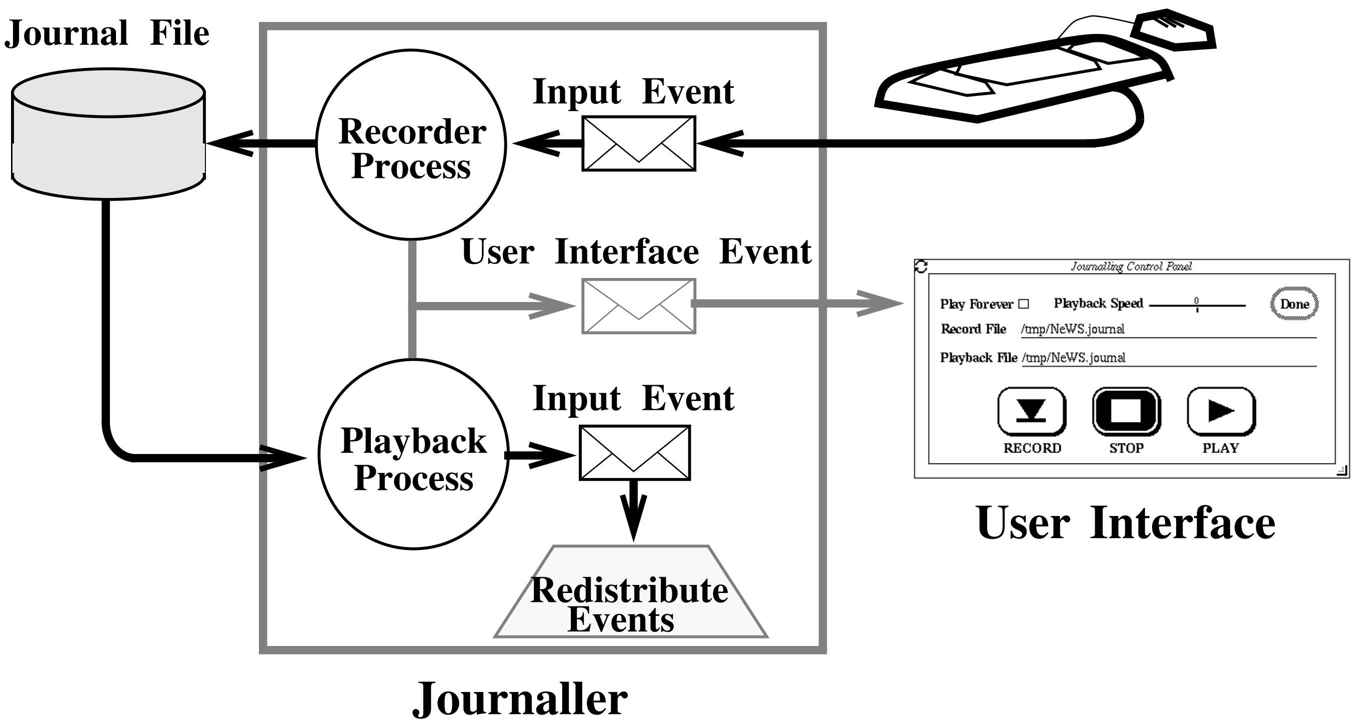
- **Application Written Entirely in PostScript**
- **Can Access All Server-Resident Toolkit Classes (Lite*)**
- **Can Make Full Use of NeWS Environment**
  - **Lightweight Processes**
  - **Events**
  - **Class mechanism**
  - **PostScript imaging**
- **Rapid Prototyping, Development and Debugging**
- **Enrich the Overall NeWS Environment**
- **'Desk Accessories' of the NeWS World**

NeWS Calculator

- **Full Function Calculator Written in PostScript**
- **Accepts Both Mouse and Keyboard Input**
- **Runs Entirely on the Server; No Client Support**
- **Uses Lite Window/Menu Package**
- **Scaling Handled Transparently By NeWS**
- **260 Lines of PostScript**

NeWS Journalling Program

- **Records All NeWS Mouse and Keyboard Events**
- **Playback: NeWS Acts Like Player-Piano**
- **Uses**
    - **Q/A and stress testing**
    - **Tutorials**
    - **Auto-running demos**
- **Implementation**
    - **Journal file is an ASCII file read and written from PostScript**
    - **Two parts: Journaller and User Interface.**
    - **Events used to communicate between Journaller and User Interface**
      - Different User Interface easily accommodated
    - **550 lines of PostScript (250 Journalling, 300 User Interface)**



Server-Based Application Issues

- **Can application be written without client-side support?**
- **How can the NeWS environment be best used?**

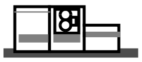
New Applications

- **Applications Designed to Use NeWS From the Start**
  - **Can take advantage of unique NeWS features**
  - **Client/Server partitioning can change during development**
  - **Can be designed for portability to a wide range of machines**
- **Some Ideas For New Applications**
  - **Dynamic network traffic display**
  - **PostScript-based Hypertext**
  - **NeWSDraw construction kit**
  - **Advanced terminal emulators**
- **At the Leading Edge**
  - **Full potential of NeWS is just beginning to be used**
  - **Join the fun!**

Stock Trader Workstation Demo

- **Joint Development Between Sun and Stratus**
- **Networked Stock Trading System**
  - **NeWS Server is front end**

    - **Provides user interface**
    - **Performs graphics display of information**
  - **Stratus Client is back end**



    - **Provides interface to wire services**
    - **Submits stock buy/sell transactions**
    - **Maintains stock and portfolio databases**
- **Implementation**
  - **Server caches recently used stocks and portfolios**
    - **Portfolio and Stock classes**
  - **Significant computation on the server side**
    - **Generalized chart display object**
    - **Database update decisions**

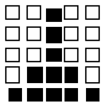
Implementation Hints

Converting From RasterOp To PostScript

- **Figure Out What Is Really Trying To Be Accomplished and Express It As A Higher-Level Abstraction**

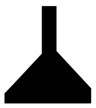
- **Example: Text Cursor in NeWS**

  - **Old way:**



    - **Determine pixel coordinates of cursor position**
    - **Turn cursor on by xoring cursor mask with screen contents**
    - **Turn cursor off by xoring again (don’t ever forget cursor state!)**

  - **NeWS way:**



    - **Create canvas shaped like the cursor (can be any shape)**
    - **Move canvas to location**
    - **Map canvas to turn cursor on, unmap canvas to turn off**

- **There Are Direct Replacements For Some RasterOps**

- **Setrasteropcode is Available If Necessary**

Debugging & Performance Improvement

- **Use psh For Incremental Testing of the Server Side**
  - **'PostScript Shell' that allows interactive PostScript execution**
- **debug.ps for the Server Side**
  - **Multiprocess debugger**
  - **Set breakpoints**
  - **Manipulate the execution stack**
  - **Add preambles and postambles to routines**
  - **Call individual routines**
  - **All data manipulation and display is done with standard PostScript**
  - **debug.ps can be run from any server connection**
- **Monitor NeWS Events**
  - **seteventlogger NeWS command**
- **Debug Client Side Just Like Any Other Program**
  - **dbx, adb, etc.**
- **Monitor Client/Server Communication**
  - **Use ASCII for Client/Server interaction while debugging**
  - **Be able to switch back to ASCII after going to bytestream encoding**
- **Performance Tips**
  - **Avoid class mechanism in performance-critical graphics sections**
  - **Relinquish inactive server storage with undef operator**
  - **Bytestream encoding offers tremendous improvement for network-intensive applications**

Summary

- **Several Important Factors in NeWS Application Design**
  - **Client/Server Code and Data Split**
  - **Data Modification Responsibility**
  - **Client/Server 'Protocol'**
- **Variety of Design Choices Available**
- **Interesting Applications Out There**
- **Many More Innovative Applications to Come**
- **Various Debugging and Performance Improvement Techniques Available**
- **Writing Applications on NeWS is a Blast!**

Acknowledgments

Go Board by Owen Densmore and Steve Evans; it will be the basis for a tutorial included in NeWS 1.1.

GNUPlot is a public domain package written by Colin Kelly and Thomas Williams of Villanova University. NeWS driver by Dick Phillips of Los Alamos National Labs.

SURFMODL is a public domain package created by Kenneth Van Kamp. NeWS driver by Dick Phillips of Los Alamos.

Alliant Performance Monitor by Ben Passerelli of Alliant; it will be included in the next release of the Alliant operating system.

Backgammon program by Phillipe LaCroute; it will be in NeWS 1.1.

Gem Emulation by Leif Samuelson of Sun. It is currently under development.

Principle implementers of Common Windows on NeWS are Ron Mak of Sun and Jim Veitch of Franz, Inc. The SPE will be released by Sun in Q1'88.

Calculator by David Lavallee of Sun. It is in NeWS 1.0.

Journalling package by Steve Isaac and Jerry Farrell of Sun; it will be in NeWS 1.1.

Principle implementer of Stock Trader demonstration is Saul Wold of Sun. It is currently under development.

Tell me about the NeWS applications you develop!

sisaac@sun.COM