WYSIWIS AND SHARED WINDOWS

Prasun Dewan
Department of Computer Science
University of North Carolina at Chapel Hill
dewan@cs.unc.edu
## COUPLING

<table>
<thead>
<tr>
<th>Issue</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Management</td>
<td>How do distributed users create, destroy, join, and leave collaborative sessions?</td>
</tr>
<tr>
<td>Single-user Interface</td>
<td>What are the application semantics if there is a single user in the session?</td>
</tr>
<tr>
<td>Coupling</td>
<td>What is the remote feedback of a user command and when is it given?</td>
</tr>
<tr>
<td>Access Control</td>
<td>How do we ensure that users do not execute unauthorized commands?</td>
</tr>
<tr>
<td>Concurrency Control</td>
<td>How do we ensure that concurrent users do not enter inconsistent commands?</td>
</tr>
</tbody>
</table>
APPLICATION-SPECIFIC COUPLING

Coupling depends on model-interactor division, when changes are announced, when they are sent, and when they are applied.

Application-Independent Coupling?

Why?

What?
Model is a blackbox and we had to make assumptions about it to automate sharing

Not all programmers are aware of or care about model-interactor division
END-USER

- Easier to understand for the user
  - Is synchronization real-time or not real time?
  - What changes are sent?
  - .....
**CHALLENGES**

- How to define it in application independent fashion.
  - Mapping between input and local feedback is application dependent.
  - Implies mapping between input and remote feedback is also app-dependent.
  - Relationship between local and remote feedback may be app-independent.
WYSIWIS: What You See Is What I See

Remote Feedback = Local Feedback

What?
Remote User Sees Everything Local User Sees

When?
If Interaction Stopped, Remote User Will Eventually Get Feedback
Remote User Feels Collaboration is Real Time
“Best Effort” to Give Immediate Feedback
Remote User does not Notice Delay (<50ms)

Application

Coupling

STRICT WYSIWIS COUPLING

If user 1 moves green window
Then green window moves on user 2’s screen

If user 2 scrolls up in orange window
Then orange window scrolls up on user 2’s screen
**Pros and Cons**

**Pros**
- Easy to understand
- Application-independent
- Automatable

**Cons**
- Window and scroll wars
- Size and view wars
- Communication overhead

Are some wars worse than others?
NEAR/RELAXED-WYSIWIS

Different window sizes create ambiguities (cropping, scaling) and scroll synchronization creates referential transparency.

Screen pointer has not much meaning if some windows are shared or if windows are moved independently.

How to point?

Telepointer: A shared shape (possibly per user) that can be dragged over any part of a top-level window.

May have private windows which are obscured.

How to automate WYSIWIS/near WYSIWIS?

**Automation & Pre-Requisites**

- **Automation**: Some application-independent infrastructure provides some functionality.
  - UI toolkit automates widgets.
  - Sync automates model sharing
- **Use assumptions to provide the automation.**
  - Swing and AWT assume applications do not want round widgets.
  - Sync assumes users do not want interactor sharing
- **Need to make similar assumptions.**
  - As not model sharing, assume I/O in interactor instead of write methods are tapped
Automating Constraints

Need access to input and/or output
Assume I/O can be tapped.
Assume certain properties of I/O
**Shared I/O Systems**

- **Collaboration-Unaware Application**
  - Input/Output
  - **Collaboration-Aware Proxy**
  - **Collaboration-Unaware I/O System**
  - **User**

**Assume application uses some I/O system**

**Will put a proxy module in between application and I/O system, if possible**

**To the application it behaves like the I/O system**

**To the I/O system it behaves like the application**

**Application and I/O system are collaboration-unaware**

**Proxy (infrastructure) is collaboration-aware**

**Proxy will distribute I/O**

**Much line proxies in shared model systems**
I/O Abstractions

- Console-based Interactor
  - Console Text Component
  - Widgets (Text Component, Button, Slider)
- Toolkit-based (GUI) Interactor
- Window-based (Graphics) Interactor

FrameBuffer

Flexibility vs. Automation Tradeoff in Abstraction Design
Collaboration-Unaware Application

read(input)

Input/Output

Collaboration-Aware Proxy

print(output)

Collaboration-Unaware Console Library

User

Programming languages and OS’s provide teletype (console library)

Cannot handle GUIs
**Toolkit Library I/O**

- **Collaboration-Unaware Application**
  - **User**
  - **Text Changed**
  - **Input/Output**
  - **Collaboration-Aware Proxy**
    - **SetText**
  - **Collaboration-Unaware User-Interface Toolkit**
  - **Press a**

- **Toolkit provides widgets such as text box, slider, and buttons.**
  - **Input is notifications about state changes**
  - **“Output” sets and gets widget state**
  - **Widget automation bound to specific toolkit (Swing, AWT)**
  - **Toolkits are built on top of window system**
  - **Cannot be used to share window state (e.g. window size)**
  - **Cannot be used to share collaboration-unaware window apps**
Neither console nor toolkit sharing can share this user-interface
**FRAMEBUFFER I/O**

Collaboration-Unaware Application

Input: Mouse and character events with screen coordinates

Output: Draw pixel images

Forced into WYSWIS

No private windows in pure FrameBuffer

Commercial systems combine shared window and framebuffer (later)

Collaboration-Aware Proxy

Collaboration-Unaware Console Library

press a

User

\$a^1, x, y\$

drawImage (Pixmap)
**Window System (Simple Model)**

- **Collaboration-Unaware Application**
  - User presses a
  - Input/Output: $a^w_1, x, y$

- **Collaboration-Aware Proxy**
  - Input/Output: draw $a, w_1, x, y$

- **Collaboration-Unaware Window System**

**Windows are untyped rectangular screen areas in which point is a pixel**

- Input indicates keyboard and mouse operations with window relative coordinates

**Output draws text, shapes**

- Application = programmer-defined code + toolkit

**Window-level automation accommodates all toolkits and allows private and public windows**

- Architecture with N users?
**Window System I/O?**

- **Window Positions**
- **Window Size**
- **Screen Pointer**
- **Mouse Move**
- **Window Border**
- **Scroll Positions**
- **Key Click**
- **Mouse Click**
- **Mouse Drag**

**Input indicates keyboard and mouse operations with window relative coordinates**

**Window Manager**

**Toolkit**

**(Top-Level) window moves, resizes and mouse clicks processed by window manager**

**Scrolling is just mouse clicks processed by a scroll widget**
**Window Manager**

- **Input**: Window move, resize user operations
- **Output**: Window move, resize
- Window manager separate from window system
- Can change window manager for a window system
- Window manager is a separate module/process
- Puts decorations on window for moving resizing

- Window Client
  - Win1, x, y
  - Move window x, y
- Window moves
- Window moved
Window Manager vs. System

Client receives events from both and can invoke actions on both
**Proxy-based Injection and Interception**

- **Collaboration-Unaware Application**
  - Input/Output: $a \wedge, w_1, x, y$
  - User: press $a$

- **Collaboration-Aware Proxy**
  - Input/Output: draw $a, w_1, x, y$

- **Collaboration-Unaware Window System**
Replicated Window System

Collaboration-Unaware Application

Input Distributor

Collaboration-Unaware Window System

User 1

press a

Collaboration-Unaware Application

Input Distributor

Collaboration-Unaware Window System

User 2

draw a, w^1, x, y

a^, w^1, x, y

Collaboration-Unaware Application

Input Distributor

Collaboration-Unaware Window System

User 2

draw a, w^2, x, y

a^, w^2, x, y

Collaboration-Unaware Application

Input Distributor

Collaboration-Unaware Window System

User 2

draw a, w^3, x, y

a^, w^3, x, y
How are corresponding windows found?

Replicated shared window systems assumes same sequence of windows.
Can use names of Java windows
### Replicated Model Algorithm

<table>
<thead>
<tr>
<th>For each input I</th>
</tr>
</thead>
<tbody>
<tr>
<td>I should be followed by matching EditInput, EditMade, EditNotified, EditObserved, EditDisplayed</td>
</tr>
</tbody>
</table>

| For each replica, I should be followed by matching EditSent to Others |

### How to change to replicated window system

<table>
<thead>
<tr>
<th>For each EditReceived R</th>
</tr>
</thead>
<tbody>
<tr>
<td>R should be followed by matching EditMade, EditNotified, EditObserved, EditDisplayed</td>
</tr>
</tbody>
</table>

| For each replica, R should be followed by matching EditSent |


Replicated Window System (Review)
# Replicated Model Algorithm

For each input I

- I should be followed by matching EditInput, EditMade, EditNotified, EditObserved, EditDisplayed

For each replica, I should be followed by matching EditSent to Others

---

How to change to replicated window

For each EditReceived R

- R should be followed by matching EditMade, EditNotified, EditObserved, EditDisplayed

For each replica, R should be followed by matching EditSent

---
Replicated Window Algorithm

For each Window w, create Telepointer \( w^t \)

Dispatching means giving it to appropriate (sub) window listeners to process

Window system can only guarantee delivery not processing

For each Window (including Telepointer) Input I

I should be followed by matching WindowEventDispatched

For each replica, if Transmit(I) then I should be followed by matching WindowEventSent

Not all events may be sent (relaxed WYSIWIS)

For each WindowEventReceived R

R should be followed by matching WindowEventDispatched

For each replica, R should be followed by matching WindowEventSent
**TRANSMIT FUNCTION**

- **Filter by Event Type**
  - Window Manager State
  - Window Size
  - Screen Pointer
  - Mouse Move
  - Scroll Positions
  - Key Click
  - Mouse Click
  - Mouse Drag

- **Telepointer Drag**
  - Send all mouse drags (of Telepointer)?
  - Desired (actual) Time between drags < 30 ms (10ms)

- **Filter by Event Time**
  - Send actions of all windows?
  - Not locking window, mail window

- **Filter by Window Name**
**SYSTEM-SPECIFIC ISSUES**

For each Window $w$, create Telepointer $w^t$

For each Window (including Telepointer) Input $I$

$I$ should be followed by matching $WindowEventDispatched$

For each replica, if $Transmit(I)$ then $I$ should be followed by matching $WindowEventSent$

For each $WindowEventReceived$ $R$

$R$ should be followed by matching $WindowEventDispatched$

How to intercept $I$?

How to inject $I$?

How to filter events?

How to translate window IDs?
Discussion so far fairly abstract

Need real window system to make it concrete

Will use Java as language for exercises and class examples
CASE STUDY: JAVA AWT

Hides the underlying window system from programmer

Portability
Window System Classes

Top Level

Window¹

Sub Windows

Window¹²

Sub Windows

Window¹²¹

Window²

Window¹²²
AWT/Swing Window System Classes

Top Level
- Window
  - (J)Frame
  - Container
    - (J)Component
      - (J)Panel
        - Canvas

Sub Windows
- (J)Component
  - Canvas
All OS events forwarded to AWT can be intercepted and injected

All Swing events may not be interceptable/injectable

Could not intercept/inject caret position in text components

Some OS events may not be correctly interceptable/injectable

Seem to inject two events for AWT checkbox

Which layer is more sharable: AWT/Swing?

Telepointer possible over Swing but not less controllable AWT windows

AWT/Windows

Swing Windows

OS Window
CLASS AWT/Swing Library

Library provided to make hide messy details of AWT/Swing

(Easily) Sharable AWT/Swing

Swing Windows

AWT Windows
Window I/O: Output

Window Client

Window System

a, win1, x, y

draw a, win1, x, y

press a

Target of draw in Java, on what kind of object is it invoked?
Java Helper Class: Graphics(2D) Context

- `drawString()`
- `drawOval()`
- `drawLine()`
- `drawRect()`
- `setColor()`
- `fillOval()`
- `fillLine()`
- `fillRect()`
EXAMPLE GRAPHICS CALLS

```
g.drawOval(charX - X_OFFSET, charY - Y_OFFSET, DIAMETER, DIAMETER);
g.drawLine(charX, charY, charX, charY - CARAT_LENGTH);
g.drawString("" + lastChar, charX, charY);
```

Called when?

- **When input is given**
- **When output is required**

What if window is hidden?

- **0 to N times for each input**

Who repaints the window when it is exposed: window system or application

- **While the component is unexposed, the drawing might have changed, so system copy might not be current**

So, in general the application must redraw it.
Expose Events

Window Client

Window Manager
- Window moved
- Window resized

Window System
- Application can requests expose event
  - When a window is resized, or (un) obscured expose event is sent to it with exposed region(s)
  - Application can draw only exposed region(s)
  - Or may draw complete window
    - Window system clips in case app draws outside window bounds

- draw complete win
- draw exposed win
OVERLAPPING WINDOWS

Exposed rectangle
OVERLAPPING WINDOWS (VERTICAL WINDOW ON TOP)

Multiple exposed rectangles
**Single Exposed**

- Single exposed rectangle
- Drawing outside exposed region will be clipped
- Window system could also keep last drawn pixels of each window as backing store
- Repainting trades off time efficiency for space
- Important if windows are lightweight and nested
How to accommodate both?

Called when?

When what is to be drawn changes

lastChar, charX, charY

When exposed area changes?

What is the Java API for receiving expose events?
**AWT based Expose Event Processing**

- **Window Client Class**
- **Window Class (JPanel)**
- **repaint()**

**paint (Graphics)**

**IS-A**

**Window manager queues paint/expose event for window**

**Processing queue results in (paint event) in paint method of window to be called**

**Repaint method can be called by application to queue an expose event for window**

**Application can override paint to draw**
public class ACircledCharacterDrawer extends JFrame implements MouseListener, KeyListener {
...
// called when an enqueued paint event for this component is dequeued
public void paint(Graphics g) {
    super.paint(g); // clears the window
    // better to use FontMetrics to center circle
    g.drawOval(charX - X_OFFSET, charY - Y_OFFSET, DIAMETER, DIAMETER);
    g.drawLine(charX, charY, charX, charY - CARAT_LENGTH);
    g.drawString("" + lastChar, charX, charY);
}
public void keyTyped(KeyEvent event) {
    setChar(event.getKeyChar());
}
public void setChar(char newValue) {
    lastChar = newValue;
    repaint(); // enqueues a paint event
}
public void mousePressed(MouseEvent event) {
    charX = event.getX();
    charY = event.getY();
    repaint(); // enqueues a paint event
}
Window I/O: Input

Window Client

- press a
- a

Window System

- a ^, win1, x, y
- draw a, win1, x, y

Correct
JAVA INHERITANCE BASED INPUT EVENT PROCESSING

processEvent (AWTEvent)  Window Client Class  public final dispatchEvent() in window called in response to input event in that window

IS-A

dispatchEvent() calls protected processEvent

processEvent (AWTEvent)  Window Class (JPanel)  dispatchEvent (AWTEvent)

processEvent can be overridden by application subclasses

This was the approach used in Java 1.0
**Single-Inheritance Problem**

Window Client Class \( \rightarrow \) Client Class

Window Class (JPanel) \( \rightarrow \) IS-A

IS-A
**Conceptual Problem with Inheritance**

- **Window Client Class**
  - IS-A **Client Class**
  - IS-A **Window Class (JPanel)**

- **Window input processor (or drawer) is not a window**
DELEGATING TO WINDOW SYSTEM

Window Client Class

Client Class

Window Class (JPanel)

IS-A

HAS-A
Java Coarse-Grained Delegation-based Input Event Processing

Window Client Class

eventDispatched (AWTEvent)

Toolkit

HAS-A

addAWTListener()

processEvent (AWTEvent)

Window Class (JPanel)

dispatchEvent (AWTEvent)

Must distinguish between mouse and event and key pressed, key typed, mousepressed, mousedragged, and other actions

Single way to get all events and then possibly dispatch them – useful for sharing events and telepointer
Higher-Level, Per-Window Listeners

- KeyPressed (MouseEvent)
- mousePressed (MouseEvent)
- processEvent (AWTEvent)
- addMouseListener()
- addKeyListener()
- addMouseMotionListener()
public class ACircledCharacterDrawer extends JFrame implements MouseListener, KeyListener {
    public ACircledCharacterDrawer() {
        addMouseListener(this);
        addKeyListener(this);
    }
    public void keyTyped(KeyEvent event) {
        setChar(event.getKeyChar());
    }
    public void mousePressed(MouseEvent event) {
        charX = event.getX();
        charY = event.getY();
        repaint(); // enqueues a paint event
    }
}
**Fine-Grained Implementation**

Window Class (JPanel)

- keyPressed (MouseEvent)
- processEvent (AWTEvent)

Window Client Class

- mousePressed (MouseEvent)
- addMouseListener()
- addKeyListener()
- addMouseMotionListener()
**Output Processing: Inheritance**

- **Window Client Class**
  - `paint (Graphics)`
  - `IS-A` relation
  - `Call` to `Window Class (JPanel)`
  - `repaint()`

- **Window Class (JPanel)**
  - `paint (Graphics)`
Output Processing: Delegation

Window Client Class

Window Class (DelegateJPanel)

addPaintListener()

Delegate classes not related by subtype relationship !(DelegateContainer IS-A DelegateComponent)

The paint method of library classes will call paint methods in delegates

As does Shareable AWT/Swing Library

SWT supports delegation model in toolkit

O U T P U T  P R O C E S S I N G :  D E L E G A T I O N

paint (Graphics)

HAS-A

paint (Graphics)
Understanding Java Window System

```java
public class ACircledCharacterDrawer extends JFrame implements MouseListener, KeyListener {

    // called when an enqueued paint event for this component is dequeued
    public void paint(Graphics g) {
        super.paint(g); // clears the window
        // better to use FontMetrics to center circle
        g.drawOval(charX - X_OFFSET, charY - Y_OFFSET, DIAMETER, DIAMETER);
        g.drawLine(charX, charY, charX, charY - CARAT_LENGTH);
        g.drawString("" + lastChar, charX, charY);
    }

    public void keyTyped(KeyEvent event) {
        setChar(event.getKeyChar());
    }

    public void setChar(char newValue) {
        lastChar = newValue;
        repaint(); // enqueues a paint event
    }

    public void mousePressed(MouseEvent event) {
        charX = event.getX();
        charY = event.getY();
        repaint(); // enqueues a paint event
    }
```
**System-Specific Issues**

- How to create a telepointer?
- How to intercept input for broadcast?
- How to inject received input?
- How to translate window IDs?
- How to filter events?
How to Create a Window TelePointer

A component can be painted by it and all of its ancestors

A key or mouse event in a component is also an event in all of its ancestors

Nesting: smaller component overrides drawing and input processing of enclosing static components

TelePointerFrame

IS-A

(J)Frame

HAS-A

(J)Panel

HAS-A

(J)TextArea

Cannot share existing user interfaces

Cannot use nesting to draw telepointer

A component can be painted by it and all of its ancestors

A key or mouse event in a component is also an event in all of its ancestors

Nesting: smaller component overrides drawing and input processing of enclosing static components

Cannot share existing user interfaces

Cannot use nesting to draw telepointer
How to Create a Window Telepointer?

Replace the top-level frame’s window with one that draws movable telepointer shape?

Cannot use nesting to draw telepointer

Even if we could, cannot share existing user interfaces
**Layering vs. Nesting**

**Frame**
- setGlassPane()

**Frame and components**

**Glass Pane**

**Nesting:** smaller component overrides drawing and input processing of enclosing static components.

**Layering:** Higher dynamic layer overrides drawing and input processing of lower, possibly smaller components. Can simulate dynamic multiple parents of a child.

**Type based vs. structure based overriding**
**How to Create a Window TelePointer**

Layering: Higher dynamic layer overrides drawing and input processing of lower, possibly smaller components. Can simulate dynamic multiple parents of a child.

How do children components get events?

Glass pane will consume its events (drag of telepointer) and re-dispatch others to deepest children.

Tricky and my solution does not always work (menus) – based on code found on the web.
**HOW TO CREATE A WINDOW TELEPOINTER ABSTRACTION?**

- **TelepointerUtility**: 
  - attach(Frame)

- **Cannot draw our own shape**

- **GraphicsPainter**: 
  - paint(Graphics)

- **AnExtendibleTelePointerGlassPane (Frame)**

- **AnExtendibleTelePointerGlassPane**
  - addPainter(GraphicsPainter)
  - getPointerX/Y()

- **GlassPaneController**
  - getGlassPaneController()
ATTACHING A TELEPOINTER AND Painter

```java
public interface GraphicsPainter {
    void paint(Graphics g);
}

glassPane = new AnExtendibleTelePointerGlassPane(telePointedFrame);
glassPane.addPainter(createTelePointerPainter());

public interface GlassPaneController {
    int getPointerSize();
    void setPointerSize(int aSize);
    int getPointerWidth();
    void setPointerWidth(int aWidth);
    int getPointerHeight();
    void setPointerHeight(int aHeight);
    boolean isShowTelePointer();
    void setShowTelePointer(boolean showTelePointer);
}

Painter should use the dimensions in controller to draw shape
HOW TO CREATE A TELEPOINTER

- Instantiate a telepointer glasspane, passing it a JFrame
- Implement a telepointer painter
- Painter should reference the telepointer glass pane to get paint position
- Painter should reference the telepointer controller to get paint dimensions
SYSTEM-SPECIFIC ISSUES

- How to create a telepointer?
  ✔️
- How to intercept input for broadcast?
- How to inject received input?
- How to translate window IDs?
- How to filter events?
GENERAL MODEL OF SINGLE-USER WINDOW SYSTEM

Collaboration-Unaware Application

Collaboration-Unaware Window System

User 2

press a

a^, w^2, x, y
**General Model of Replicated Window System**

![Diagram showing the general model of replicated window system]

- **Collaboration-Unaware Application**
- **Input Distributor**
- **Collaboration-Unaware Window System**
- **User 2**

The diagram illustrates the flow of input from User 2 to the application through the input distributor and collaboration-unaware window system.
**General Model of Single-User Window System Again**

Collaboration-Unaware Application

\[ a^w, w^2, x, y \]

Collaboration-Unaware Window System

\[ a^w, w^2, x, y \]

press a

User 2

Hot to tap and inject input in Java
Can intercept events at the same time they are dispatched to local components

No direct way to inject or stop events – need to inspect event and send it to appropriate component
JAVA INPUT QUEUE

Collaboration-Unaware Application

EventQueue

Collaboration-Unaware Window System

User 2

press a

$\text{^a}, w^2, x, y$

$\text{^a}, w^2, x, y$
Replacing Queue

Collaboration-Unaware Application

\[ a^w, w^2, x, y \]

InputDistributorQueue

\[ a^w, w^2, x, y \]

Collaboration-Unaware Window System

press a

Can intercept events before they are dispatched to application

Must remember to forward to application

Must prevent cycles

Must filter uncoupled events

Toolkit.getDefaultToolkit().getSystemEventQueue().push(
    new InputDistributingQueue());
Library Listenable Event Queue

Collaboration-Unaware Application

AnExtendibleAWTEventQueue

Collaboration-Unaware Window System

ListeningInputDistributer

Toolkit.getDefaultToolkit().getSystemEventQueue().push(this)

AnExtendibleAWTEventQueue.getEventQueue().addEventQueueHandler(new ListeningInputDistributer());
**How to Intercept and Inject Window Events**

- Window Manager State
- Window Size
- Screen Pointer
- Mouse Move
- Key Click
- Mouse Click
- Mouse Drag

Will define a listener to get filtered events

AWTEventQueueHandler

newEvent (AWTEvent)

Singleton class, invoke static method in it to get global queue

Inject event?

AnExtendibleAWTEventQueue

static getEventQueue()

getCommunicationEventSupport()

Will convert serializable event to local event

addEventQueueHandler (AWTEventQueueHandler)

dispatchReceivedEvent (AWTEvent)

Does not fire new event
**LISTENABLE, INJECTABLE EVENT QUEUE**

```java
package util.awt;
public interface ExtendibleAWTEventQueue extends PropertyVetoerRegistrar {
    public void addEventQueueHandler(AWTEventQueueHandler listener);
    public void removeEventQueueHandler(AWTEventQueueHandler listener);
    public void clearEventQueueHandlers();
    public void dispatchEvent(AWTEvent event);
    void dispatchReceivedEvent(AWTEvent anEvent);
}
```

dispatchEvent vs. dispatchReceivedEvent ~ replicatedAdd vs. observableAdd
**System-Specific Issues**

- How to create a telepointer? ✔
- How to intercept input for broadcast? ✔
- How to inject received input? ✔
- How to translate window IDs?
- How to filter events?
How to find corresponding windows in different replicas?
**Translating Window IDs**

Collaboration-Unaware Application

\[ a^\wedge, w^1, x, y \]

Input Distributor

\[ a^\wedge, w', x, y \]

Collaboration-Unaware Window System

\[ draw\ a, w^1, x, y \]

User 1

Collaboration-Unaware Application

\[ a^\wedge, w^2, x, y \]

Input Distributor

\[ a^\wedge, w^2, x, y \]

Collaboration-Unaware Window System

\[ draw\ a, w^2, x, y \]

User 2

Collaboration-Unaware Application

\[ a^\wedge, w^3, x, y \]

Input Distributor

\[ a^\wedge, w^3, x, y \]

Collaboration-Unaware Window System

\[ draw\ a, w^3, x, y \]

User 2

User 2
Connecting Replica Windows

Replicated shared window systems can assume same sequence of windows.

Can use names of Java windows

How to replace local window ids (Component instances) in events with global ids (integer, string) and global ids in events with local ids?
Translation

Given AWTEvent e, component c

Send ASerializableAWTEvent (e, toID(c))

ASerializable AWTEvent

ASerializable AWTEvent (AWTEvent, String)

Given SerializableAWTEvent (event e, id i)

Dispatch toDispatchedEvent(e, toComponent(i)) to AnExtendibleAWTEventQueue

CommunicatedAWTEventSupport

AWTEvent toDispatchedEvent(SerializableAWTEvent, Component component);

AnExtendibleAWTEventQueue.getEventQueue().getCommunicatedEventSupport()
package util.awt;
public class ASerializableAWTEvent implements SerializableAWTEvent {
    public ASerializableAWTEvent(AWTEvent theEvent, String theComponentId) {
        ...
    }
}

SerializableAWTEvent serializableEvent (SerializableAWTEvent) aMessage;
AWTEvent aDispatchedEvent =
    AnExtendibleAWTEventQueue.getEventQueue().
        getCommunicatedEventSupport().toDispatchedEvent(
            serializableEvent, toComponent(serializableEvent));
AnExtendibleAWTEventQueue.getEventQueue().dispatchReceivedEvent
    (aDispatchedEvent)

toComponent() written by programmer to translate between global id and local component
How to find corresponding windows in different replicas?

How to find the windows and creation sequence in each replica?
Once we find a window, we can recursively find all of its descendants

\[
((\text{Container} ) \text{ component}).\text{getComponents}();
\]

Resize event sent to EventQueue when it is created

\[
\text{AWTMisc.} \text{.isResizeEvent(event);} 
\]

If E is window creation (resize) event then register the global ids of its subtree if the subtree has not already been registered
A TALE OF TWO RESIZE EVENTS

Resize event sent both when window resized and it is created.

Want to dispatch normal received resize events but not creation events.

Connect queue listener before window tree created to get resize events.

To prevent window creation events remotely broadcast.

| broadcaster can be attached after window tree is created, which means two different listeners |
| broadcaster can have a special mode to separate the two phases |
| receive listener can be attached after local window tree created |
SYSTEM-SPECIFIC ISSUES

- How to inject received input?
  ✔

- How to create a telepointer?

- How to intercept input for broadcast?
  ✔

- How to inject received input?
  ✔

- How to translate window IDs?
  ✔

- How to filter events?
Transmit Function

**Window Manager Events**
- Filter by Event Type
- Done by library

**Mouse Drag**
- Filter by Event Time

**Filter by (Top) Window Name**
- Object event.getSource();
- SwingUtilities.getRoot (Component)

**Need to cast source as Component**

**System.currentTimeMillis();**

**AWT Misc. isMouseDragged Event(event);**

**Need to ensure that last mouse drag event is sent**
Replicated vs. Centralized Window System

Collaboration-Unaware Application

Input Distributor

Collaboration-Unaware Window System

User 1

Press a

Centralized?

Collaboration-Unaware Application

Input Distributor

Collaboration-Unaware Window System

User 2

Draw a, w^1, x, y

Draw a, w^2, x, y

Draw a, w^3, x, y

a^, w^1, x, y

a^, w^2, x, y

a^, w^3, x, y
Centralized Shared Window System

Collaboration-Unaware Application

Collaboration-Aware Proxy

Collaboration-Unaware Window System

Input/Output

Collaboration-Aware Proxy

Collaboration-Unaware Window System

Master Computer

User 1

The shared application runs on the (master) computer of only one of the collaborators

The shared application runs on the (master) computer of only one of the collaborators

Each user’s input relayed to application through proxies

Each output of application broadcast to all users through proxies

Slave Computer

User 2
**Proxy Functions**

- **Window Application**
  - User 1
    - $a^w, w^1, x, y$
    - draw $a, w^1, x, y$
  - I/O Relayer & Output Broadcaster
    - $a^w, w^i, x, y$
    - draw $a, w^i, x, y$
  - Window System
    - press $a$

- **I/O Relayer**
  - User 2
    - $a^w, w^2, x, y$
    - draw $a, w^2, x, y$
  - Window System

- **Slave proxy relays input to app through master proxy**
  - User 3
    - $a^w, w^3, x, y$
    - draw $a, w^3, x, y$
  - Window System

- **Master proxy broadcasts output to all slave proxies, which relay to local window system**

- **Proxies translate window ids**
Replicated Window Algorithm

For each Window w, create Telepointer w^t

For each Window (including Telepointer) Input I

I should be followed by matching WindowEventDispatched

For each replica, if Transmit(I) then I should be followed by matching WindowEventSent

For each WindowEventReceived R

R should be followed by matching WindowEventDispatched

For each replica, R should be followed by matching WindowEventSent

How to change it to centralized?
**Centralized Window Algorithm**

- **Master and Slave**
  - For each Window $w$, create Telepointer $w^t$

- **For each Window (including Telepointer) Event $I$**
  - $I$ should be followed by matching WindowEventDispatched (including Telepointer)
  - If isSlave() and Transmit(I) then $I$ should be followed by matching WindowEventSent to Master

- **Master Receiver**
  - For each WindowEventReceived $R$ at Master, $R$ should be followed by matching WindowEventDispatched
  - For each output call $O$, $O$ should be followed by WindowRequestMade and WindowRequestSent to all Slaves

- **Slave Receiver**
  - For each WindowRequestReceived $R$ at Slave, $R$ should be followed by WindowRequestMade
Centralized vs. Replicated Window Systems

Centralized vs. Replicated Shared Window Systems

~ Centralized vs. Replicated Shared Model Systems
Replicated Model: Issues

Consistency issues of causality and concurrent operations (to be addressed later)

Correctness and performance issues when model is non deterministic, accesses central resources, and has side effects

All of these problems still occur in replicated window systems
**Example of Non Determinism and Other Replication Problems (Review)**

Different users will see different output.

Behavior of centralized and replicated different.

Assumption: Output should be only a function of input.

Non determinism!
A central model executes operations that are expensive, non-idempotent, or access central resources.

Distinquished “Replica” Model Solution

Cannot use application-specific solution if supporting collaboration-unaware applications.
Centralized vs. Replicated Model

- None of the replication issues
- Feedback times involve round trip delays
- Feed through incurs extra hop (beyond relaying)
- Refresh and query operations also involve round trip delays (e.g. searching history)
- Can we fix the last problem?

Cannot use caching of high-level state, no local non window state

Refresh, scrolling involves round trips
**Shared Window Systems**

Problems of centralization and replication get aggravated

Collaboration-awareness required for distinguished process in replicated systems

In central systems, round trip for readable model state

Plus other problems
CONCURRENT/INTERLEAVED INTERACTION

User inputs can get (un) desirably mixed

Can multiple users generate a stream not creatable by one user?

Window System

Window Application

I/O Relayer & Output Broadcaster

User 1

Type a

User 2

Type b

User 3

I/O Relayer

Window System

I/O Relayer

Window System
Explicit Floor Control

Multiple users can indeed generate input sequence that cannot be generated by single user.

Can break (explicit/implicit) assertions of collaboration-unaware code.

Solution: Explicit Floor Control

One user interacts at a time and explicitly gives floor to another.

Floor control is application-independent and hence automatable.

Problem occurs in both centralized and replicated shared window systems.
_SHARED WINDOW SYSTEMS_

Problems of centralization and replication get aggravated

Collaboration-awareness required for distinguished process in replicated systems

In central systems, round trip for readable state

Plus other problems

Invalid window sequences possible in shared (centralized and replicated) window systems because intra-sequence constraints in window system events
Problem with Relaxed WYSIWIS in Centralized Systems

<table>
<thead>
<tr>
<th>Window Manager State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window Size</td>
</tr>
<tr>
<td>Screen Pointer</td>
</tr>
<tr>
<td>Mouse Move</td>
</tr>
<tr>
<td>Key Click</td>
</tr>
<tr>
<td>Mouse Click</td>
</tr>
<tr>
<td>Mouse Drag</td>
</tr>
</tbody>
</table>
Sharing Window Manager State

Not broadcasting can prevent window wars.

Not all systems couple window manipulation events (XTV).
RELAXED WYSIWIS COUPLING

Implemented in XTV

Independent Window Minimization

Independent Window Positions
Pop-up menus are top-level windows drawn at absolute positions by application.

Inner windows drawn relative to containing window.
Correct Pop-Up Menus

Proxy keeps track of root window and translates.

More than a proxy – understands underlying window system.

Not uncommon in proxies.
SEMANTIC ISSUE

- Should window state be coupled?
  - Coupled $\rightarrow$ window wars (Stefik et al ’85)
  - Uncoupled $\rightarrow$ no referential transparency
    - Cannot refer to the “upper left” shared window
    - Problems in centralized systems
- Compromise for centralized system
  - Create a virtual desktop on a slave computer for physical desktop of the master user
VIRTUAL DESKTOP

Privately scrollable, movable window representing master screen

Nested, shared, WYSIWIS master window

Pop up menus not a problem

But slave system has to create a virtual desktop, much more than a proxy, a window manager
ANOTHER ISSUE: COUPLING OF UNEXPOSED REGIONS?
**Replicated Systems (Review)**

Collaboration-Unaware Application

Input Distributor

Collaboration-Unaware Application

Input Distributor

Collaboration-Unaware Application

Input Distributor

User 1

User 2

Centralized?
CENTRALIZED SYSTEMS (REVIEW)

- **Window Application**
  - User 1: $a^i, w^1, x, y$
  - User 2: $a^i, w^1, x, y$
  - User 3: $a^i, w^1, x, y$

- **I/O Relayer & Output Broadcaster**
  - User 1: draw a, w^1, x, y
  - User 2: draw a, w^i, x, y
  - User 3: draw a, w^i, x, y

- **I/O Relayer**
  - User 1: $a^i, w^2, x, y$
  - User 2: $a^i, w^2, x, y$
  - User 3: $a^i, w^2, x, y$

- **Window System**
  - User 1: press a
  - User 2: draw a, w^2, x, y
  - User 3: draw a, w^3, x, y

**Slave proxy relays input to app through master proxy**
**Master proxy broadcasts output to all slave proxies, which relay to local window system**
**Proxies translate window ids**
**SHARED WINDOW SYSTEMS**

Problems of centralization and replication get aggravated

<table>
<thead>
<tr>
<th>Collaboration-awareness required for distinguished process in replicated systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>In central systems, round trip for readable state</td>
</tr>
</tbody>
</table>

Plus other problems

<table>
<thead>
<tr>
<th>Invalid window sequences possible in shared (centralized and replicated) window systems because intra-sequence constraints in window system events</th>
</tr>
</thead>
<tbody>
<tr>
<td>In centralized systems with relaxed WYSIWIS, pop up menus and obscured master windows can create problems</td>
</tr>
</tbody>
</table>
Relaxed WYSIWIS Coupling (First-Class Remote Windows))

Implemented in XTV

Independent Window Minimization

Independent Window Positions
Corresponding windows in first-class remote windows and virtual desktops may be exposed differently.
EXPOSE EVENTS

Collaboration-Unaware Application

- expose w, rects

Window System

- draw exposed w
- draw complete w

User

- resize, (un)obscure w
- clip w

When a window is resized, or (un)obscured expose event is sent to it with exposed regions

Application is expected to draw only exposed regions

May draw complete window

Window system clips in case it app. drawn outside window bounds
APPLICATIONyll Draws Exposed Windows

Window Application
- expose w1, rects
  - draw exposed w1

I/O Relayer & Output Broadcaster
- expose w1, rects
  - draw exposed w1
  - draw exposed w1

Window System
- draw exposed w1
  - clip w1
  - resize, (un)obscure w1

User 1
- clip w1

Window System
- draw exposed w2
  - clip w2

User 2
- clip w2

Window System
- draw exposed w3
  - clip w3

User 3
- clip w3

Each computer shows only regions exposed in master
- Slave shows old contents or black holes
-Expose events not sent from slaves
-Window may be exposed differently on different computers
Application Draws Complete Window

Window Application
- expose w^1, rects
  - draw w^1

I/O Relayer & Output Broadcaster
- expose w^1, rects
  - draw w^i

I/O Relayer
- draw w^1
  - Window System
    - User 1
      - resize, (un)obscure w^1
      - clip w^1
  - draw w^2
    - Window System
      - User 2
      - clip w^2
  - draw w^3
    - Window System
      - User 3
      - clip w^3

Slave shows obscured contents in master
Slave shows contents not seen in master

How to overcome problem?
PREVENTING PRIVACY ISSUES: EXPOSE COUPLING

Send expose event to all proxies
They can then filter draw requests to show only exposed regions

Window Application

I/O Relayer & Output Broadcaster

Window System

User 1

User 2

User 3

expose \(w^1\), rects
draw \(w^1\)
expose \(w^i\), rects
draw \(w^i\)

Draw \(w^1\)

Clipdraw rects, \(w^2\)

Clipdraw rects, \(w^2\)

Clip \(w^1\)

Clip \(w^2\)

Clip \(w^3\)
Window-based Coupling

- Mandatory
  - Window sizes
  - Window contents

- Optional
  - Window positions
  - Window stacking order
  - Window exposed regions

- Optional can be done with or without virtual desktop
  - Without virtual desktop, problems of pop up menus

- In both cases, expose events are an issue that proxies must address
"Output" Calls with Return Values

Collaboration-Unaware Application

Input/Event: Transfer info to App

Window System

User

Output/Request/Call: Transfer Info from App

"Output"/Request/Call: Transfer Info to App

Fonts listFonts

press a

move w

move w

move w

move w

a \wedge, w^1, x, y

move w

expose w, rects

draw a, w^1, x, y

draw in exposed w

resize, (un)obscure w

press a

move w

move w

move w
UNICASTING TO MASTER

A call with return value may be unicast to master window system
Broadcasting Calls with Return Values

A call with return value may be broadcast to all window systems.

Master proxy combines replies.

Window Application

I/O Relayer & Output Broadcaster

I/O Relayer

Window System

User 1

User 2

User 3

listFonts() 9, 11

listFonts() 5, 9, 11

listFonts() 1, 9, 11

listFonts() 3, 9, 11

listFonts() 1, 9, 11

listFonts() 3, 9, 11

listFonts() 9, 11

listFonts() 1, 9, 11

listFonts() 5, 9, 11

listFonts() 3, 9, 11

A call with return value may be broadcast to all window systems.

Master proxy combines replies.

Window Application

I/O Relayer & Output Broadcaster

I/O Relayer

Window System

User 1

User 2

User 3

listFonts() 9, 11

listFonts() 5, 9, 11

listFonts() 1, 9, 11

listFonts() 3, 9, 11

listFonts() 9, 11

listFonts() 1, 9, 11

listFonts() 5, 9, 11

listFonts() 3, 9, 11

A call with return value may be broadcast to all window systems.

Master proxy combines replies.
**Unicast to Some Slave**

A call with return value may be unicast to any window system.

Unicast to slave involves network delay and translation.

- **Window Application**
  - listFonts()
  - 3, 9, 11

- **I/O Relayer & Output Broadcaster**
  - 3, 9, 11

- **Window System**
  - User 1

- **Window System**
  - User 2

- **Window System**
  - User 3

- **I/O Relayer**
  - listFonts()
  - 3, 9, 11
**Unicast to Active Slave**

Application may draw at pointer location

A call with return value may need to be unicast to window system with floor

Should really listen to event than poll

Some window systems are more conducive to sharing

**Window Application**

\( w^1, x, y, \) leftButton ^

**I/O Relayer & Output Broadcaster**

queryPointer()

**Window System**

**User 1**

\( w^1, x, y, \) leftButton ^

**I/O Relayer**

**User 2**

\( w^2, x, y, \) leftButton ^

queryPointer()

**Window System**

Has Floor

**User 3**

**Window Application**

\( w^1, x, y, \) leftButton ^
**Shared Window Systems**

Problems of centralization and replication get aggravated

Collaboration-awareness required for distinguished process in replicated systems

In central systems, round trip for readable state

Plus other problems

Invalid window sequences possible in shared (centralized and replicated) window systems because intra-sequence constraints in window system events

In centralized systems with relaxed WYSIWIS, pop up menus and obscured master windows can create problems

In centralized systems with query calls, the target of these calls is an issue
Research systems tried both architectures.

VConf (Lantz ’86, Stanford)
Rapport (Ahuja ’89, Bell Labs)
XTV (Abdel-Wahab ’91, UNC/ODU)
MeetingPlace (Cisco)
CollaborateNow (IBM)
Webex
LiveMeeting (Microsoft)
Meeting Space (Vista)

Commercial ones implement centralized
Why is centralized harder based on what you know about Java?
**Interception Differences**

In replicated architecture, must intercept and inject input.

In centralized architecture, must intercept input and output, and inject both.
INPUT VS. OUTPUT DIFFERENCES

Collaboration-Unaware Application

EventQueue

Collaboration-Unaware Window System

User 2

All input defined by a single event type

Funneled through a single replaceable object

$\text{press a}$

$a^$, $w^2$, $x$, $y$
**How to Intercept Output Calls?**

Calls to both frame and graphics must be intercepted.

Java not designed to allow output interception.

Calls are made by making different invocations, not passing a unifying data structure.

How to change AWT/Swing?
Library Global Queue

Window App

- paint(ListenableGraphics)
- draw()

ADelegateFrame

- repaint()
- setSize()
- notifyListeners (Serializable GraphicsRequest)

AnOutputQueue

- notifyListeners (Serializable FrameRequest)
- addOutputListener()

AnOutputQueue.addOutputListener(this);
GLOBAL QUEUE

All frame calls and listenable graphics calls sent to output queue

AnOutputQueue, like AnExtendibleAWTQueue, allows listeners
**MASTER PAINTER**

```java
public class AListenableCharacterDrawer
    extends ACursorTrackerOfDelegateFrame
    implements ListenablePainter {
    final static int CARAT_LENGTH = 10;
    public AListenableCharacterDrawer(ADelegateFrame theDelegateFrame) {
        super(theDelegateFrame);
        delegateFrame.addPainter(this);
    }
    public void paint(ADelegateFrame theDelegateFrame, ListenableGraphics g) {
        g.drawLine(charX, charY, charX, charY - CARAT_LENGTH);
        g.drawString("" + lastChar, charX, charY);
    }
    public void mousePressed(MouseEvent event) {
        super.mousePressed(event);
        delegateFrame.repaint();
    }
    public void keyTyped(KeyEvent event) {
        super.keyTyped(event);
        delegateFrame.repaint();
    }
}
```

For each frame, need to track graphics requests issued in last paint call so we can make transactions at the other end.
public interface SerializableRequest extends Serializable {
    public int getFrameId();
    public String getName();
    public Object[] getArgs();
}
public interface SerializableGraphicsRequest extends SerializableRequest{
    public static final String DRAW_OVAL = "drawOval";
    public static final String DRAW_RECT = "drawRect";
    public static final String DRAW_STRING = "drawString";
    public static final String DRAW_LINE = "drawLine";
    public static final String PAINT_START = "paintStart";
    public static final String PAINT_END = "paintEnd";
    public Rectangle getClipBounds();
    public void setClipBounds(Rectangle theRectangle);
}
public interface SerializableFrameRequest extends SerializableRequest{
    public static final String CREATE_FRAME = "createFrame";
    public static final String SET_SIZE = "setSize";
}
**INTERCEPTING OUTPUT IN REAL WORLD**

- **Call in the window system/widget usually does not result in notifications**
- **Local infrastructure module can send screen diffs to other computers**
- **Must poll and send periodically**
- **Intercepting output needed only in centralized architectures**

Diagram:
- **Collaboration-Unaware Application**
  - `a^, w^1, x, y`
- **Input Distributor**
  - `draw a, w^1, x, y`
- **Collaboration-Unaware Window System**
  - `draw pixrect_1, pixrect_n`
**Window System as a Library**

**Collaboration-Unaware Application**

**Collaboration-Aware Proxy**

**Collaboration-Unaware Window System**

**In Java window system is a library and not a separate process**

**Microsoft Windows**

**Window App and System run in same process**

**Library proxy possible but not common, have to run proxy code for some compiled process**
**Window System Proxy**

![Diagram](image)

- **Collaboration-Unaware Application**
- **Collaboration-Unaware Window System**
- **Window Sharing System (e.g., Live Meeting, Webex)**
- **Collaboration-Aware Proxy**

Possible for remote process to get input, polling for output
Proxy Addition in Client-Server Architecture

Collaboration-Unaware Application

Collaboration-Unaware Application

Collaboration-Unaware Window System

Collaboration-Unaware Window System

X Window System

Collaboration-Aware Proxy

Collaboration-Unaware Application

Collaboration-Unaware Window System

setenv hostid:port

setenv hostid:port

setenv hostid:port

Window system is Window Server

Window App and System run in different processes
**Shared Window Systems**

- **Coupling**
  - WYSIWIS, Relaxed WYSIWIS

- **Shared Window Systems**
  - Constraints on input sequences can be violated
  - Layering for telepointers

- **Replicated**
  - Input broadcast
    - Input interception, injection
  - All problems of replicated models and no awareness

- **Centralized**
  - Input relay and output broadcast
    - Input interception, injection
    - Output interception, injection
  - All problems of centralized plus pop up menus, different exposed areas, output query calls
  - Virtual desktop solves problem of referential transparency vs. per user window configuration

- **Java Implementation**
  - Input interception and injection possible
    - Implementation problems as demo shows
  - Output interception and injection not directly supported

- **Address space of proxies**
  - Different from shared application in reality
  - Easier to attach proxies if distributed window systems such as X