Pen-Based User Interfaces
Overview

- Challenges in pen-based UIs
- Platforms for development
  - Satin
  - Microsoft Tablet SDK
- Tablet operating systems
Tablet Hardware

- Electromagnetic digitizers
  - Behind LCD screen
  - Receive RF signals
  - Susceptible to electromagnetic interference
    - Require shielding from motherboard
    - Calibration
    - Surrounding objects (like wristwatches) can alter distortion
Programming challenges

- Parallax
  - Eye and pen are offset
  - Glass in front of LCD display
- Still motion
  - You can’t hold your hand steady
  - Also you can’t repeatedly target, etc.
- Handedness
  - Right-handed users occlude typical pull-down menus
Programming challenges

- Text input is slow and error prone
  - Limit text, explore other modalities
  - text

- Limited input
  - Like a 1-button mouse
Toolkits for Tablet Programming

- Mouse mapping
  - Roll your own libraries

- Satin
  - Java-based research system
  - Circa 2000, by Jason Hong and James Landay

- Microsoft Tablet SDK
  - ‘Real’ system
  - Integrated with Visual Studio/.NET framework
Satin contains

- Sheets:
  - Canvas that supports drawing
  - Contain scenegraphs of graphical objects

- Patches
  - Areas of sheets to which an interpreter is applied

- Graphical objects
  - Objects that can be displayed and manipulated
  - A stroke is the simplest graphical object in satin
SATIN

- Graphical objects contain one or more view object
  - If more than one, must also have a MultiView object
    - A policy for which view to use
- Manipulation also occurs via user gestures (strokes)
  - Strokes can be aggregated and sent to graphical objects as events via a **Stroke Assembler**
• Stroke are recognized via Interpreters
Demo

- Denim demo
Satin – Positives and Negatives

- **Positives**
  - Pen-based classes in Java
  - Useful recognizers
  - Extensible

- **Negatives**
  - Relatively limited classes, recognition
  - Uses only mouse mapping (performance)
Microsoft Tablet SDK

Tablet PC SDK

- Managed APIs
  - Tablet Input
  - Ink Data Management
  - Ink Recognition

- Standard Ink Controls
  - InkEdit
  - InkPicture

- COM Automation API

- .NET Framework
- WIN32 APIs
Digital Ink Input

- Digitizer
- Windows HID (USB)
- Tablet PC WISPTIS
- InkObj.dll (Tablet-aware Application)
- Mouse
- Windows User32.dll
- Tablet Unaware Application

- Pen input
- USB
- HID data
- Data hook
- Tablet events
- Mouse behavior
- Mouse messages
- Serial/PS2
- HID data
- Mouse messages
For performance, communicates directly with application
- For simplicity, pre-processing rather than integrated with OS
Managed API Classes

- Tablet Input
  - Cursor, CursorButton, Cursors, Gesture, InkCollector, InkOverlay, Tablets, Tablet

- Ink Data Management
  - DrawingAttributes, Ink, PacketProperty, Stroke, Strokes

- Ink Recognition
  - RecognitionAlternate, RecognitionAlternates, RecognitionProperty, RecognitionResult, Recognizer, RecognizerContext, RecognizerGuide, Recognizers, WordList
Demo

InkCollector

InkOverlay

Options:
- Draw
- Edit
- Erase

Actions:
- Recognize
- Clear
Operating Systems

- Windows XP Tablet Edition
  - Good hardware support
  - Good programming model
  - Bad operating system for tablet pcs

Diagram:

- WISPTIS
  - Tablet input events
  - Queue of tablet data (packets)
  - RPC Notifications
  - User32.dll (Mouse)
  - InkObj.dll
  - Tabet-aware application
Keepin’ It Real:
Pushing the Desktop Metaphor with Physics,
Piles and the Pen in BumpTop

Anand Agarawala, Ravin Balakrishnan
Dynamic Graphics Project
www.dgp.toronto.edu