Pen- and Touch-Based Computing

Georgia Tech

Agenda

- Natural data types
  - Pen, Audio, Video
- Pen-based topics
  - Technology
  - Ink as data
  - Recognition
- Related: Gestures (on surfaces)
  - iPhone, MS Surface
  - Technology sometimes similar to pens
  - Related issues with recognition

Natural Data Types

- As we move off the desktop, means of communication mimic “natural” human forms of communication
  - Writing.............Ink
  - Speaking...........Audio
  - Seeing...............Video
- Each of these data types leads to new application types, new interaction styles, etc.

Pen Computing

- Use of pens has been around a long time
  - Light pen was used by Sutherland before Engelbart introduced the mouse
- Resurgence in 90’s
  - GoPad
  - Much maligned Newton
- Types of “pens”
  - Passive (same as using a finger)
  - Active (pen provides some signal)

Example Pen Technology

- Passive
  - Touchscreen (e.g., PDA, some tablets)
  - Contact closure
  - Vision techniques (like MS Surface)
  - Capacative sensing (like iPhone)
- Active
  - Pen emits signal(s)
  - e.g. IR + ultrasonic
- Where is sensing? Surface or pen

Questions about Pens

- What operations detectable
  - Contact – up/down
  - Drawing/Writing
  - Hover?
  - Modifiers? (like mouse buttons)
  - Which pen used?
  - Eraser?
- Differences between Pen and Finger Gestures?
  - Can't detect fine-grained points (difficult to do writing, for instance)
  - No buttons on fingers! (But can use different gestures for "modes")
- Difference between pen and mouse?

Example: Expansys Chatpen

- Reads dot pattern on paper
- Transmits via Bluetooth

- http://www.expansys.com/product.asp?code=ERIC_CHATPEN

Example: mimio

- Active pens
  - IR + ultrasonic
- Portable sensor
  - Converts any surface to input surface
- Can chain these to create big surface

- http://www.mimio.com

Pen input

1. Free-form ink (*mostly* uninterpreted)

    - Tablet PC applications, digital notebooks, etc.

2. Soft keyboards

    - Provide high-accuracy (although slow) mechanism for inputting machine-interpretable text

3. Recognition systems

    - Recognition of **content**
        - Text: handwriting recognition, simplified textual alphabets
        - Graphics, doodles, figures: sketch-based interfaces
    - Recognition of **commands**
        - Specialized vocabulary of command symbols
        - Modal input of commands
        - Contextual commands: commands distinguished from content only in how they are used

1. Free-form ink

ink as data: when uninterpreted, the easiest option to implement

- humans can interpret
- time-stamping perhaps (to support rollback, undo)
- implicit object detection (figure out groupings, crossings, etc.)
- special-purpose "domain" objects (add a little bit of interpretation to some on-screen objects)
  - E.g., Newton: draw a horizontal line across the screen to start a new page
  - See also Tivoli work (Moran, et al., Xerox PARC)

Free-form ink examples

Ink-Audio integration

- Tivoli (Xerox PARC)
- eClass (GT)
- Flatland (Xerox PARC)
- Dynomite (FX-PAL)
- The Audio Notebook (MIT)

2. Soft Keyboards

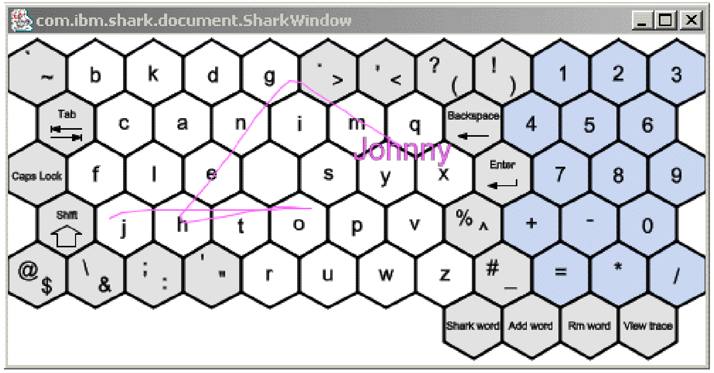
Make “recognition” problem easier by forcing users to hit specialized on-screen targets

(Sometimes a blurry line between what’s “recognition” and what’s a “soft keyboard”)

common on small mobile devices

many varieties

- tapping interfaces
- Key layout (QWERTY, alphabetical, …)
- learnability vs. efficiency



T9 (Tegic Communications)

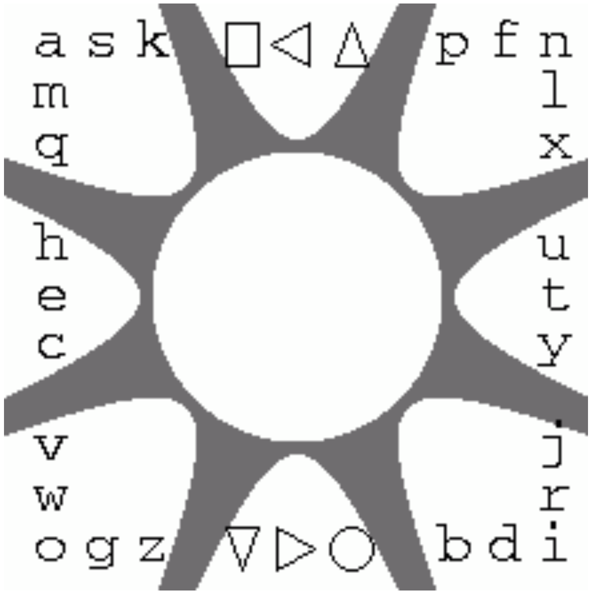
- Alternative tapping interface
- Phone layout plus dictionary

- Soft keyboard or mobile phone

  - Not usually "pen based" but ideas for rapid text entry often carry over from fingertips to pens

Quickwrite (Perlin)

“Unistroke” recognizer

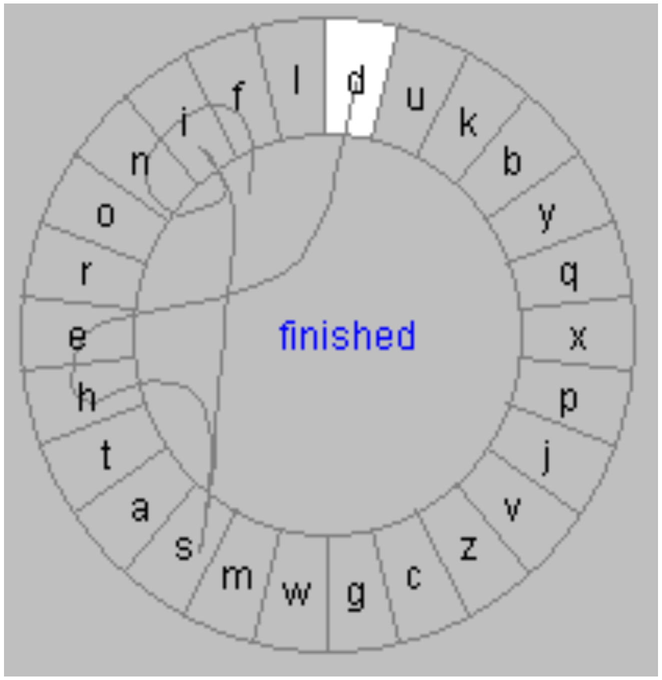


- Start in “rest” zone (center)
- Each character has a *major zone*: large white areas
- ... and a *minor zone:* its position within that area
- To enter characters in the center of a major zone, move from the rest zone to the character’s major zone, then back
    - Example: for *A*, move from rest to upper left zone then back to rest
- To enter characters at other points in a zone, move into the character’s major zone, then into *another* major zone that corresponds to the character’s minor zone
    - Example: *F* is in the top-right zone (its major zone). Move from rest to that major zone. Since *F* is in the top-center of its major zone, move next into the top-center major zone , then back to rest
- Allows quick, continual writing without ever clicking a mouse button or lifting the stylus

Cirrin (Mankoff & Abowd)

Word-level unistroke recognizer

Ordering of characters minimizes median distance the pen travels (based on common letter pairings)



3. Recognizing pen input

- Unlike soft keyboards, recognize more "natural" pen strokes
- Can be used for both content and commands

- Some are less natural than others: Graffiti
  - unistroke alphabet

- Other pen gesture recognizers
  - for commands
    - Stanford flow menus; PARC Tivoli implicit objects
  - measure features of strokes
    - Rubine, Long
  - usually no good for "complex" strokes

Handwriting (content) recognition

Lots of resources

- see Web
- good commercial systems

Two major techniques:

- on-line (as you write)

- off-line (batch mode)

Which is harder?

Handwriting (content) recognition

Lots of resources

- see Web
- good commercial systems

Two major techniques:

- on-line (as you write)

- off-line (batch mode)

Which is harder?

Offline. You don't have the realtime stroke information (direction, ordering, etc.) to take advantage of in your recognizer... only the final ink strokes.

Mixing modes of pen use

Users want free-form content and commands

- or commands vs. text

How to switch between them?

- (1 mode) recognize which applies: contextual commands, a la Tivoli, Teddy, etc.
- (2 modes) visible mode switch: Graffiti (make special command gesture)
- (1.5 modes) special pen action switches: temporary or transient mode, e.g., Wacom tablet pens

Error correction

Necessary when relying on recognizers (which may often produce incorrect results)

UI implications: even small error rates (1%) can mean lots of corrections, must provide UI techniques for dealing with errors

Really slows effective input

- word-prediction can prevent errors

Various strategies

- repetition (erase and write again)
- n-best list (depends on getting this from the recognizer as confidence scores)
- other multiple alternative displays

Other interesting applications

Signature verification

Note-taking

- group (NotePals by Landay @ Berkeley)
- student (StuPad by Truong @ GT)
- meetings (Tivoli and other commercial)

Sketching systems

- early storyboard support (SILK, Cocktail Napkin)
- sketch recognition (Eric Saund, PARC; others)

Toolkits for Pen-Based Interfaces

- SATIN (Landay and Hong) – Java toolkit

- MS Windows for Pen Computing
- MS Pocket PC, CE.net
- Apple Newton OS
- GO PenPoint
- Palm Developer environments
- GDT (Long, Berkeley) Java-based trainable unistroke gesture recognizer
- OOPS (Mankoff, GT) error correction

SATIN (UIST 2000)

- Pen input for informal input
  - Sketching (others have investigated this)
- Common toolkit story
  - Gee, “X” sure is a neat class of apps!
  - Golly, making “X” apps is tough!
  - Here’s a toolkit to build “X” things easily!

The SATIN Toolkit

- The application space
  - Informal ink apps
  - Beyond just recognition
  - Pen “look-and-feel”
- Abstractions
  - Recognizers
  - Interpreters
  - multi-interpreters