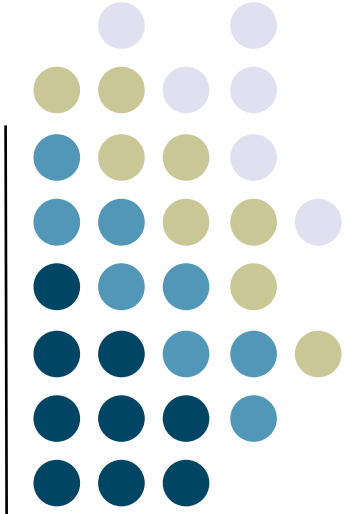
Pen- (and Simple Touch-) Based Interaction



Georgia Tech

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
- Then suppressed again by rise of multitouch (iPhone, iPad, Android)
- Now coming back with MS Surface, etc.

Intro

- Deep dive on pens and "basic" touch interaction

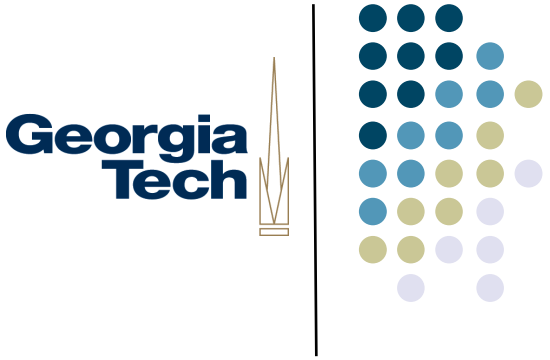
- Why discuss these together?
  - Interaction is actually somewhat different, hardware is somewhat different, but software model is similar for both
  - I'll generally call this "pen interaction" since you don't see so much basic touch these days, but pens are still prevalent

- Our first example of a "natural data type"
  - Form of input that humans normally do "in the wild," not specially created to make it easy for computers to interpret (as the case with keyboards and mice)

Natural Data Types

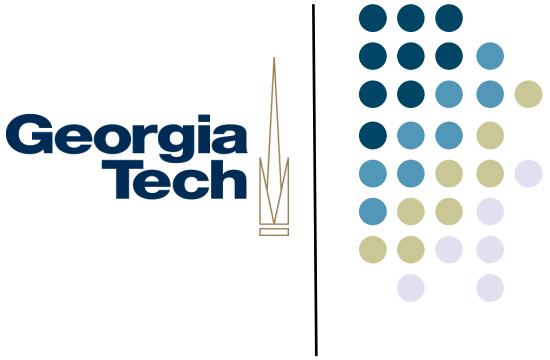
- As we move off the desktop, means of communication mimic "natural" human forms of communication
  - Writing.....Ink
  - Speaking.....Audio
  - Seeing/Acting……Video
- Each of these data types leads to new application types, new interaction styles, etc.

Interaction Model for Pens and Simple Touch



- What's the same for both pens and simple touch?
  - 2D Absolute Locator in both cases: system detects contact and reports X,Y coordinates
  - Generally (but not always) used on a *display surface*. In other words, the site of input is the same as the site for output
    - One exception to this is trackpad, which more closely emulates a mouse
    - Another exception is pens used on paper surfaces, but which can digitize input and transmit to a computer
  - Motion of pen or finger on surface can be interpreted to generate a *stroke*
    - Succession of X,Y coordinates that—when connected—can act as "digital ink"

Interaction Model for Pens and Simple Touch

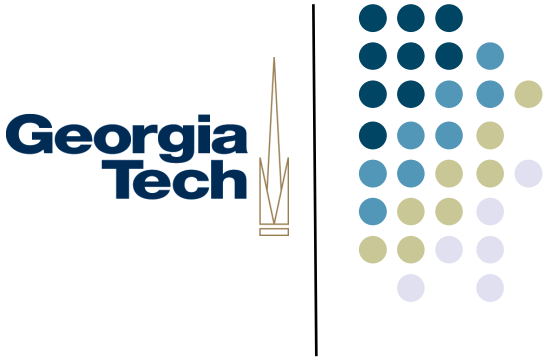


- What about differences?
  - Obvious one: precision of input. Hard to do fine-grained input with fingers, so difficult to do writing for instance
  - Not so obvious? Pens usually build in many more dimensions of input than just the basic 2D locator functionality (see next slide)

- What's the difference between pens/simple touch versus the mouse?

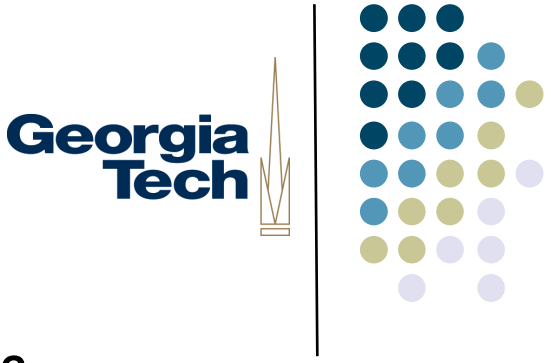
Dimensionality of Input

- What operations detectable
  - Contact – up/down
  - Drawing/Writing
  - Hover?
  - Modifiers? (like mouse buttons)
  - Which pen used?
  - Eraser?
- Fingers do not have the same dimensionality of input (when used in the simple touch case), so we have to do things like use gestures or switches for different modes of input



Quick Overview of Pen Hardware (we'll talk about touch hardware later)

Example Pen (and touch) Technology



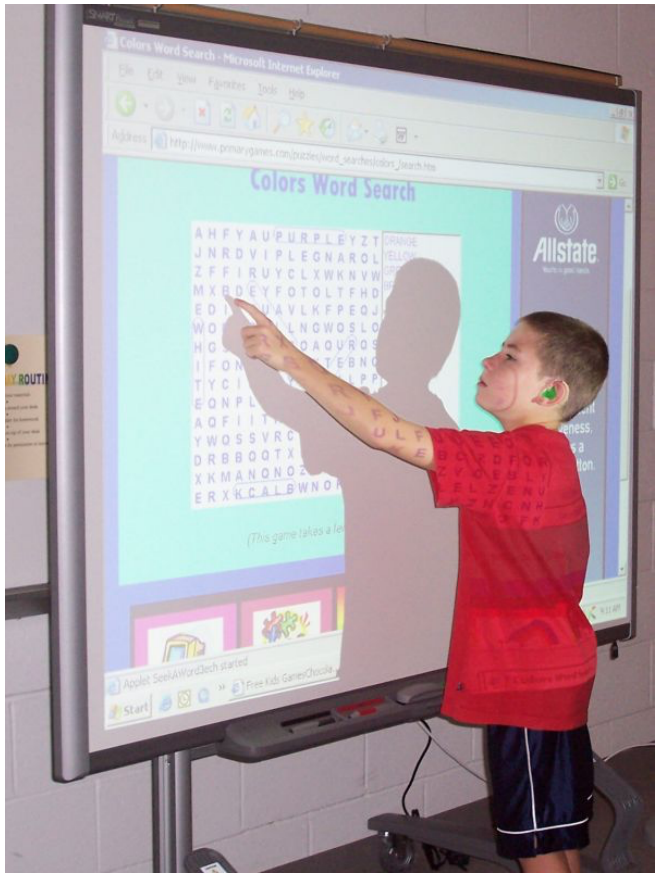
- Passive: surface senses location of "dumb" pen, or finger
  - Resistive touchscreen (e.g., PDA, some tablets)
  - Contact closure
  - Vision techniques (like MS Surface Tabletop)
  - Integrated with capacitive touch sensing (like iPhone)
  - Passive approaches also work for fingers!
- Active: pen or surface provides some signal, so that together they can determine position
  - Where is sensing? Surface or pen?
  - Pen emits signal that are detected by surface
    - e.g. IR, ultrasonic, etc.
    - Wacom electomagnetic resonance
  - Pen detects signals that are emitted by surface
    - e.g., camera-based approaches that detect "signal" printed onto surface

Passive Example #1: Palm Pilot

- Circa 1996
- 512kB of memory
- 160x160 monochrome resistive touchscreen
  - Worked with fingers or pens
- **Resistive** technology:
  - Two electrically sensitive membranes
  - When finger or stylus presses down, two layers come into contact; system detects change in resistance
- Palm interaction innovation:
  - Stylus (or finger) interpreted as **command** inputs to widgets in top of screen area, but at bottom are interpreted as **content** via simple "unistroke" recognizer

Passive Example #2: SmartBoard

- Circa 1991
- **Optical** technology:
  - Requires specialized whiteboard
  - Cameras mounted in each corner of the whiteboard
  - Signals analyzed to determine position of stylus (or finger)
  - Output projected over whiteboard, or rear projected
- SmartBoard interaction innovation:
  - Can disambiguate multiple pens



Passive Example #3: Surface Table

- Circa 2007
- **Optical** technology (in original version):
  - Cameras underneath table surface pointed upward
  - Detect contact between objects and the surface
  - (Uses *frustrated total internal reflection* technique, described later)
- Surface interaction innovation:
  - Detects fingers (multiple ones), pens, and other objects
  - Intended to support *multi-user* input

Active Example #1: mimio

- Circa 1997
- Pen emits signal, surface detects
- Active pens
  - IR + ultrasonic
- Portable (!) sensor
  - Converts any surface to input surface
  - Ultrasonic pulses emitted by pens are triangulated by sensors to derive position
- Can chain these to create big surface

- http://www.mimio.com

Active Example #2: Wacom

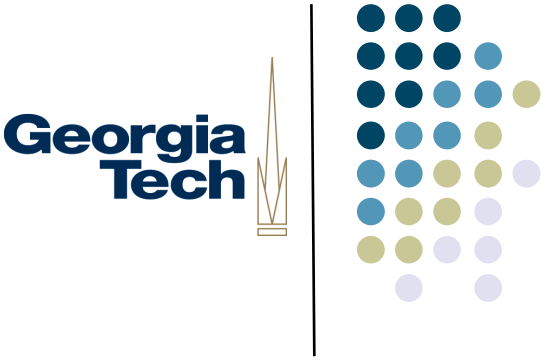
- Considered current state-of-the-art in high-quality pen input
- Electromagnetic resonance technology
  - Surface provides power to the pen via resonant inductive coupling (like passive RFID tags)—so no batteries needed in pens
  - Grid of send/receive coils in surface, energize pen, detects returned signal
  - Signal can be modulated to convey additional info (pressure, orientation, side-switch status, hardware ID, …)
  - Read up to 200 times/second
- Wacom interaction innovations
  - Extremely high dimensionality: pressure, orientation, tilt, etc etc.

Active Example #3: LiveScribe Pen

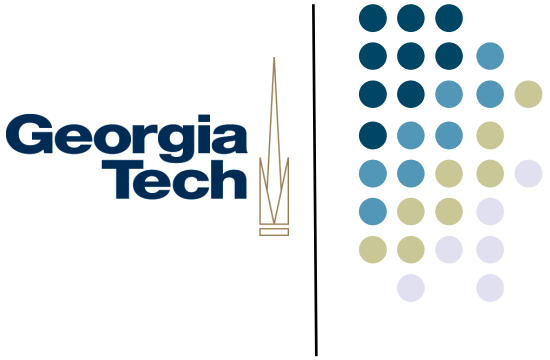
- "Smart pen" functionality while writing on real paper
  - Tiny dot pattern printed onto paper (*Anoto* ™ paper)
  - IR camera in pen detects position encoded in dots
  - Each page has a unique ID so that pages can be distinguished from each other
  - Stroke data transferred back to computer in realtime via Bluetooth
- Also includes timestamped voice recording capabilities

- Interesting app ideas: check out Paper PDA system from Hudson, et al.

What can you do with a 2D Locator? Interactions with Pens and Simple Touch



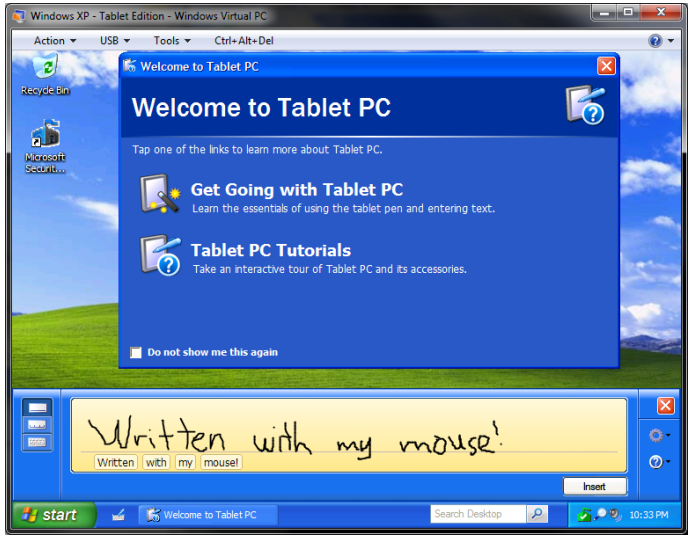
- What kinds of interactions do these afford? Several basic types
- In increasing order of complexity:
  - 1. Pen or touch as mouse replacement. BORING!
  - 2. Specialized input techniques for pens (swipe, tap, tap+hold, pull-to-refresh, …)
    - Sometimes coupled with haptic output, a la Force Touch?
  - 3. Soft keyboards: on-screen interactors to facilitate text entry
  - 4. Stroke input: free-form, uninterpreted digital ink
  - 5. Stroke input: **recognition and interpretation** of digital ink
    - As control input
    - As content

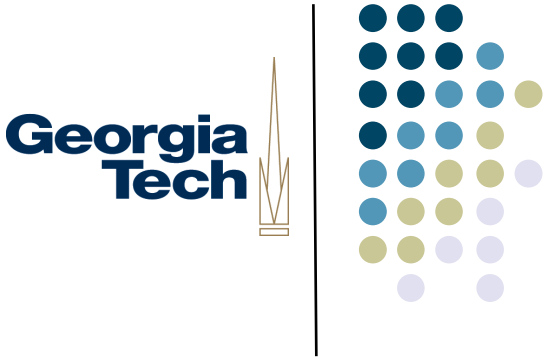


1. Pen/Touch as Mouse Replacement

Pen/Touch as Mouse Replacement

- Pretty boring.
- Canonical case: Circa 2005 Windows XP Tablet Edition
  - Standard Windows interface—build for mouse —but with a pen
  - Extra software additions for text entry
- Lots of small targets, lots of taps required (e.g., menus) — a common failure mode with pen-based UIs!
- More recent example: Windows 8 (and later) mix touch-based with mouse-based interaction



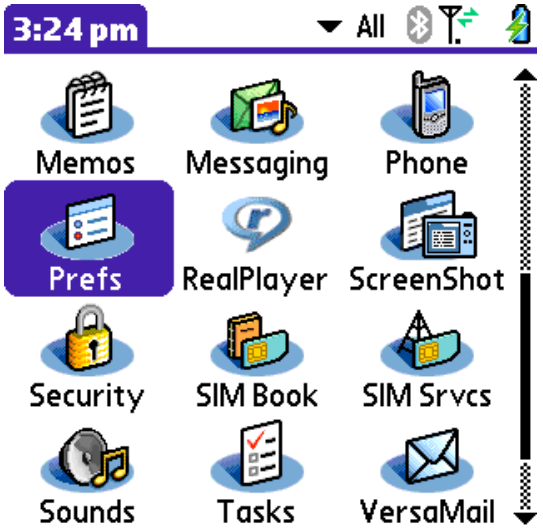


2. Specialized Input Techniques for Pens/Touch

- If you don't assume a mouse, what would you do differently?
- Fewer menus: input at the site of interaction
- Don't assume hover (no tool tips)
- Take advantage of more precise swipe movements, which are easier with pen/touch

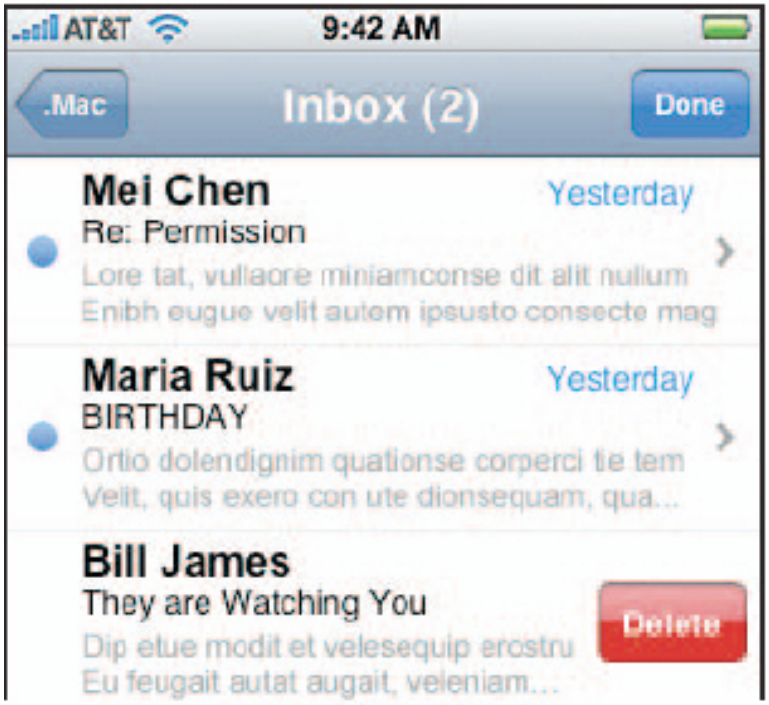
Pen & single finger touch gestures

- Typically inputs used for *command* input, not *content* input
- Most common: press/tap for selection
  - Not really much of a "gesture" at all
- Slightly more complex:
  - Double-tap to select
  - Double tap, hold, and drag to move windows on OS X
  - Tap, hold and drag to select text on the iPad
- Note: some of these don't require a screen, just a touchable surface

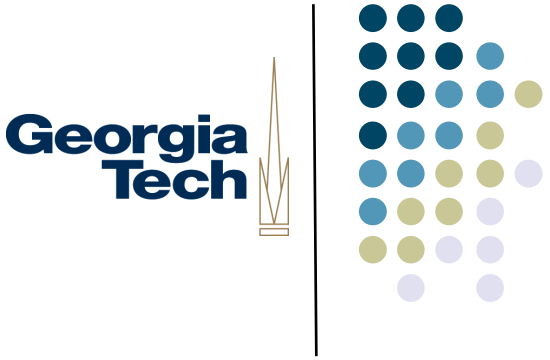


Other examples

- One-finger:
  - Special interactions on lists, etc.
  - Example: swipe over mail message to delete
  - Example: pull to refresh
  - Specialized feedback for confirmation
  - Still no good affordances though.



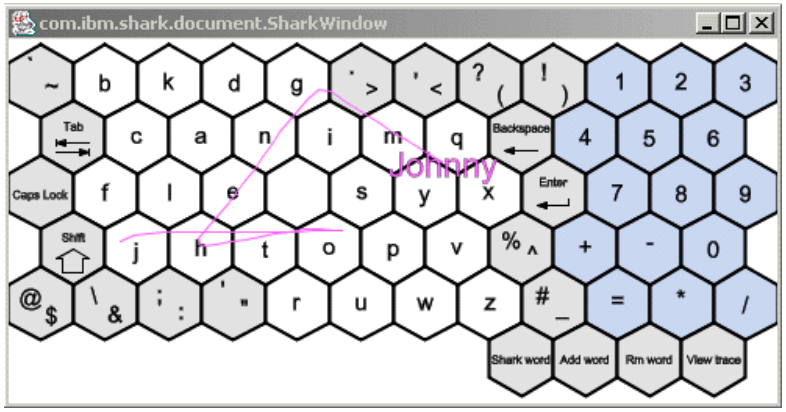
- Non-finger gestures?
  - Surface--use edge of hand for special controls
  - Technically "single touch," although most hardware that can support this is probably multitouch capable



3. Soft Keyboards

3. 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

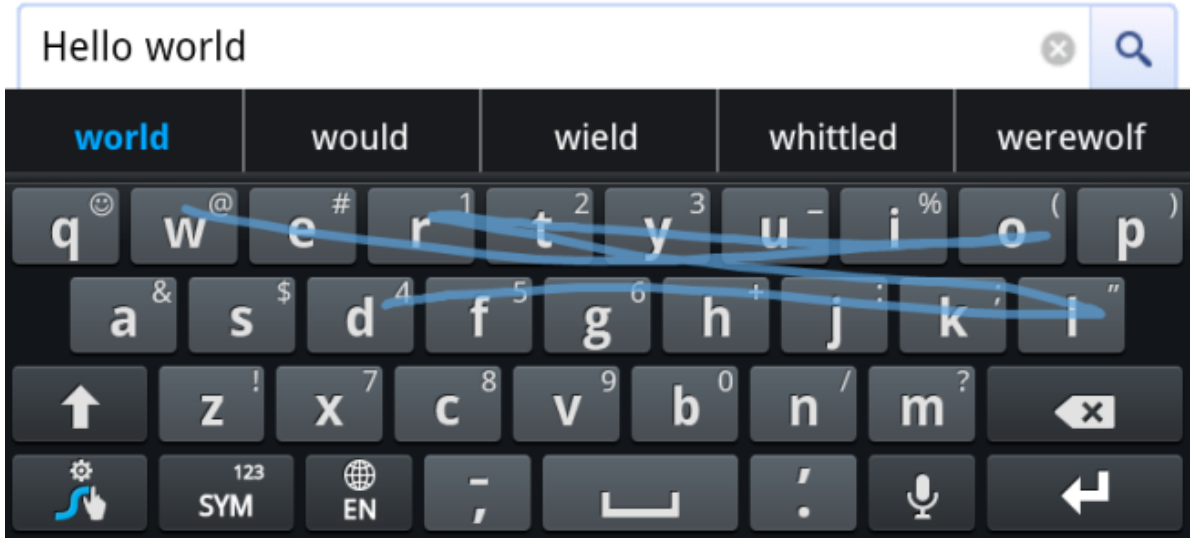
- Key layout (QWERTY, alphabetical, …)
- learnability vs. efficiency
- language model/predictive input

- Earliest ones were focused on pen usage: small, high-precision targets. Newer approaches targeted at touch usage.

Swype Keyboard

- User enters word by sliding finger or stylus from first word of a letter to its last, lifting only between words.
- Uses language model for error correction, predictive text.

- Many similar systems: SwiftKey, Swipelt,, etc.



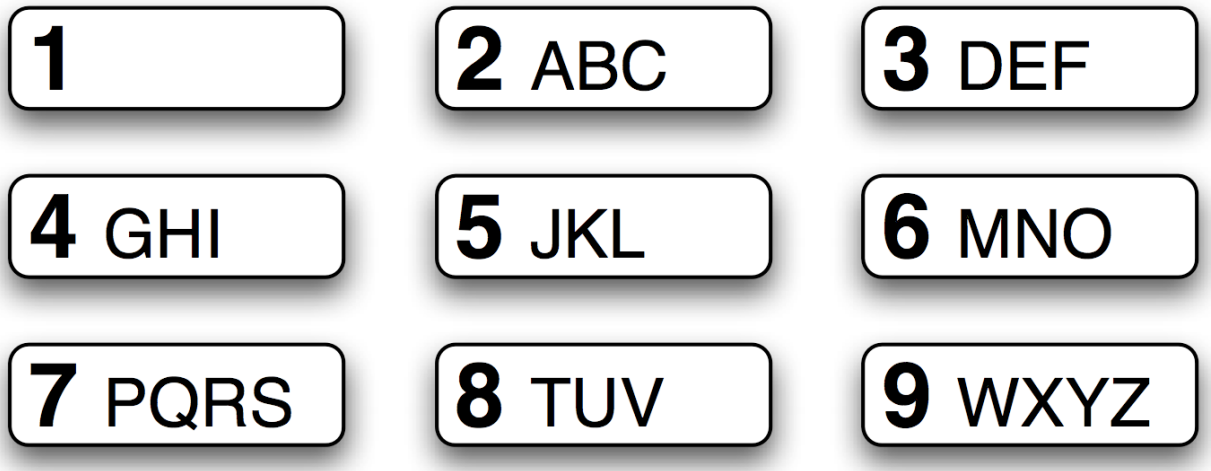
- Original version started as a research prototype by Shumin Zhai (IBM, now Google): Shorthand-Aided Rapid Keyboarding (SHARK)

T9 (Tegic Communications)

- Alternative tapping interface
- Phone layout plus dictionary

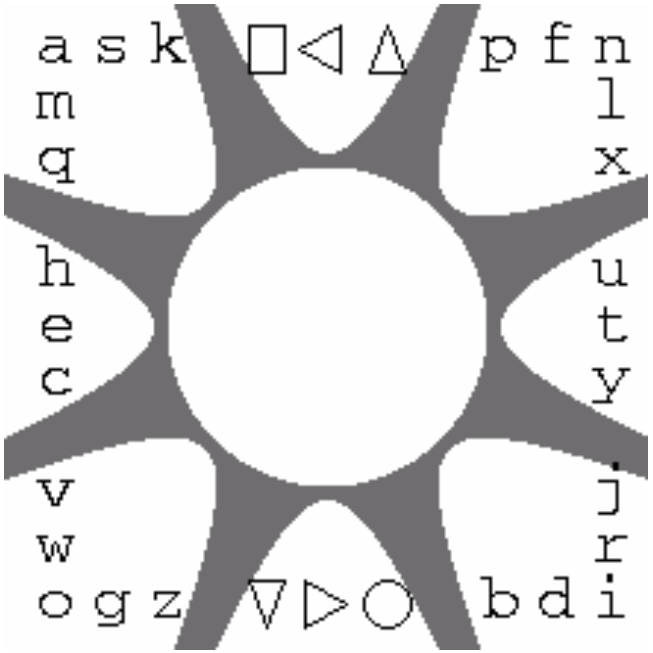
- Soft keyboard or mobile phone

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



Quickwrite (Perlin)

"Unistroke" recognizer

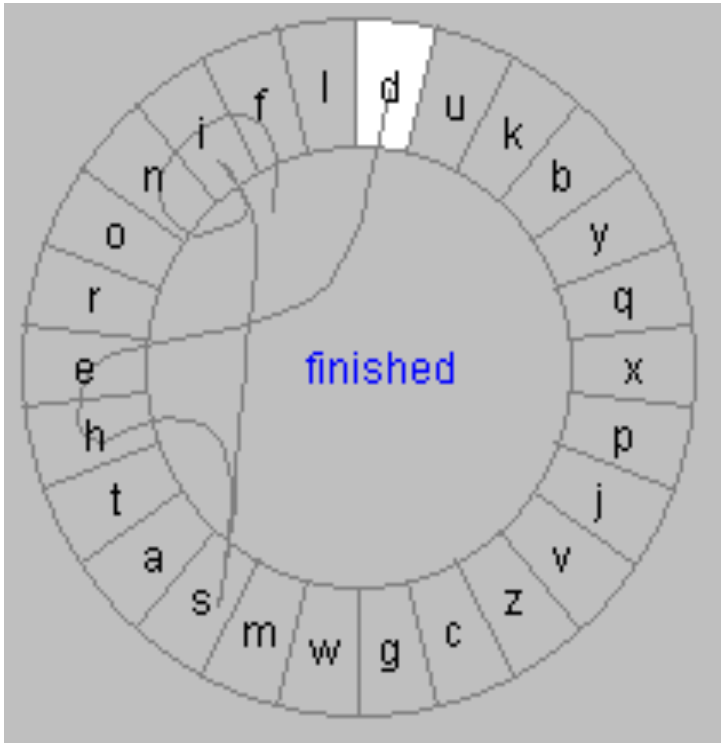


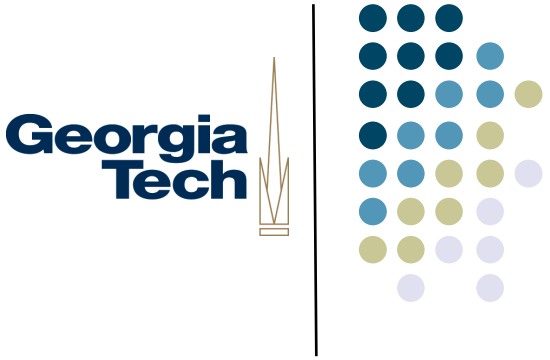
- Start in "rest" zone (center)
- Each character has a *major zone*: large white areas
- ... and a *minor zone:* its position within that area
- To enter characters in the center of a major zone, move from the rest zone to the character's major zone, then back
  - Example: for *A*, move from rest to 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)





4. Stroke Input: Free-form, Unrecognized Digital Ink

4. 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

Notetaking and Ink-Audio integration

- Classroom 2000/eClass (GT)
- Dynomite (FX-PAL)
- The Audio Notebook (MIT)
- NotePals (Berkeley)

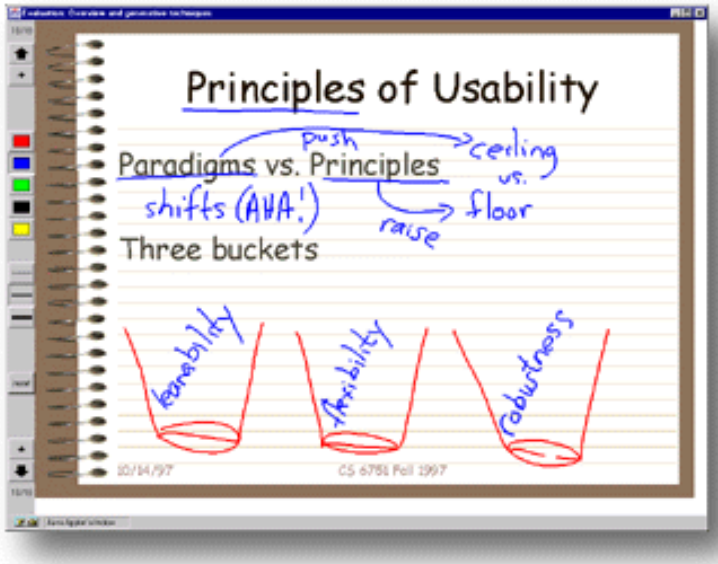
Systems with minimal/optional recognition

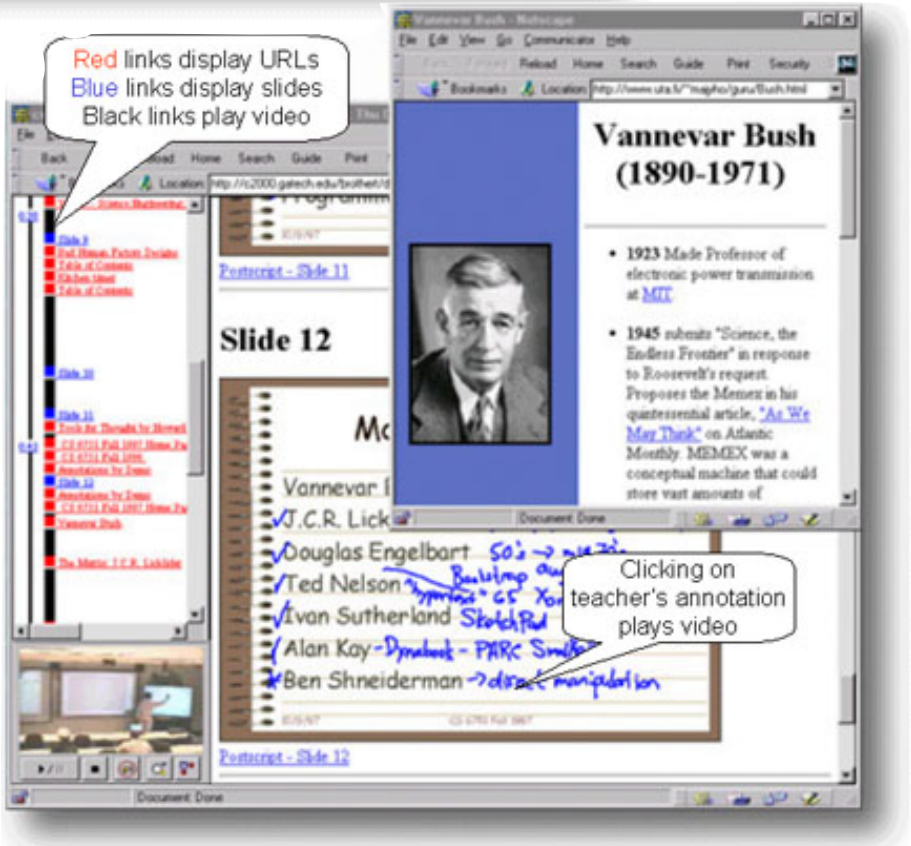
- Tivoli (Xerox PARC)

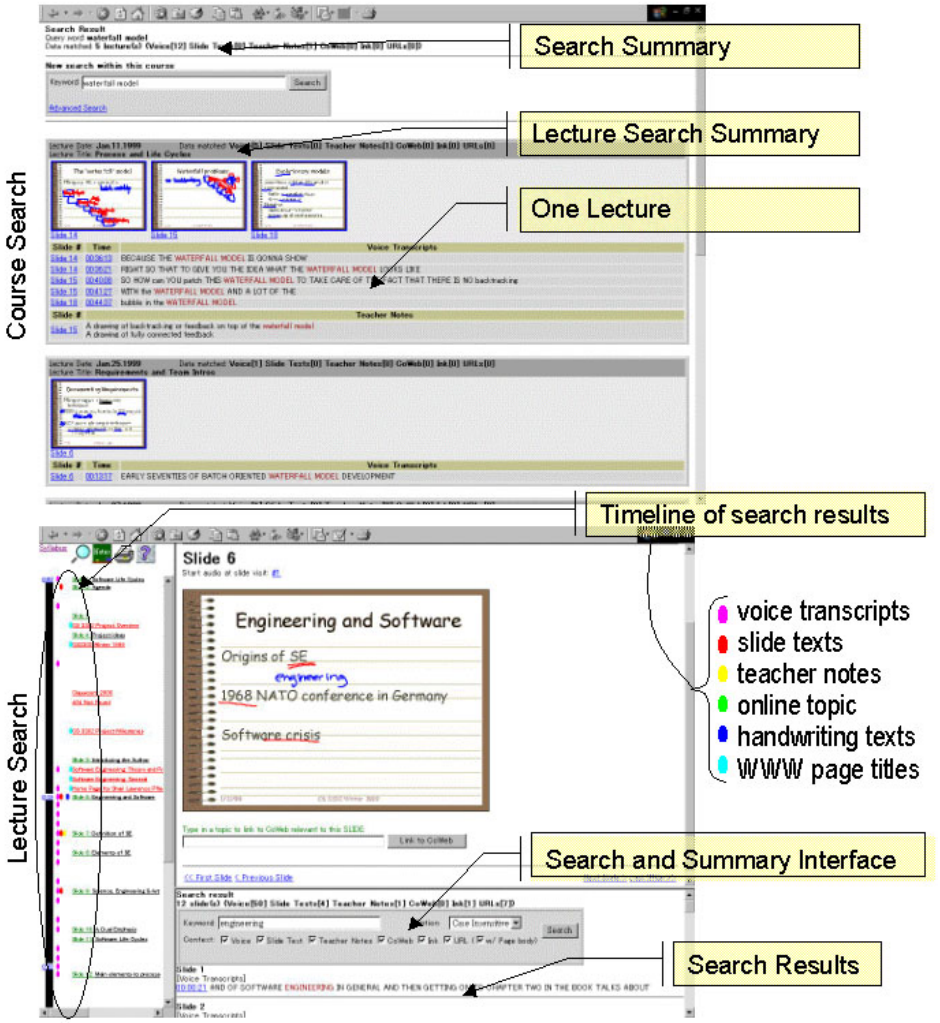
Classroom 2000 (later eClass)

- Marks made by professor at whiteboard are captured
- Simultaneously, student notes can be captured and shared
- Everything is timestamped; browsing interface includes a timeline that links to video, slides, and notes

Classroom 2000 (later eClass)

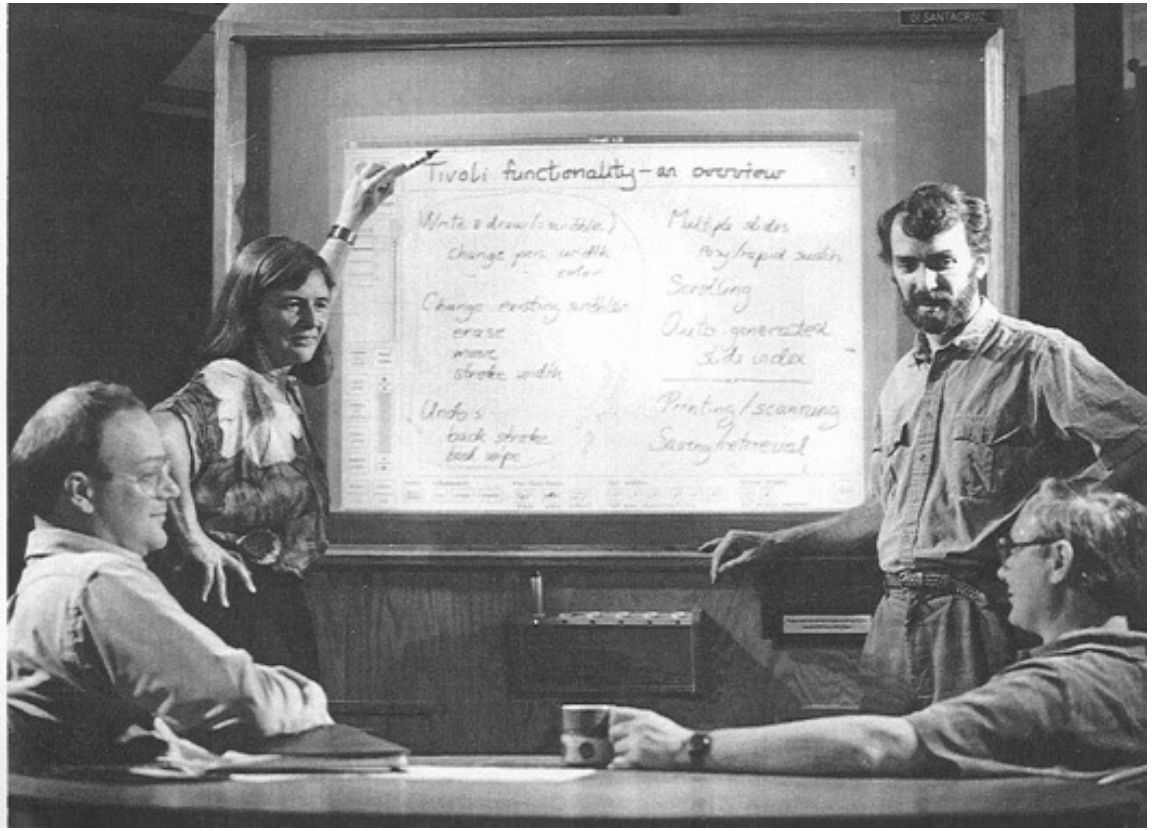






Tivoli

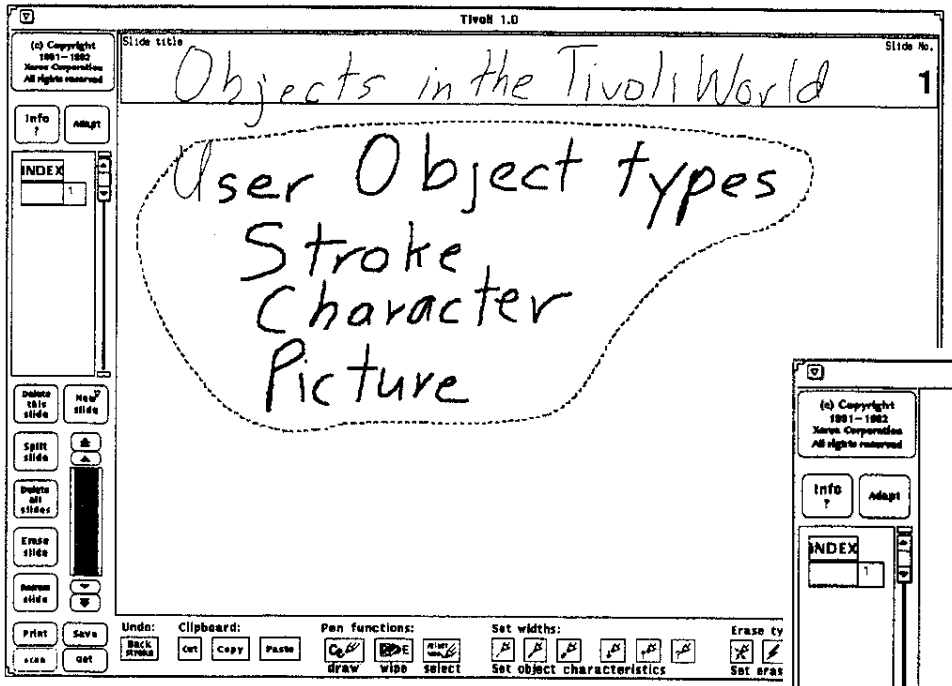
- Mark Weiser predicted three form factors for post-PC computing:
  - Tab (phone sized)
  - Pad (tablet sized)
  - Board (wall sized)
- PARC Liveboard hardware
  - Large drawing surface
  - Multipen input
  - Detection of nearby tabs and pads via IR

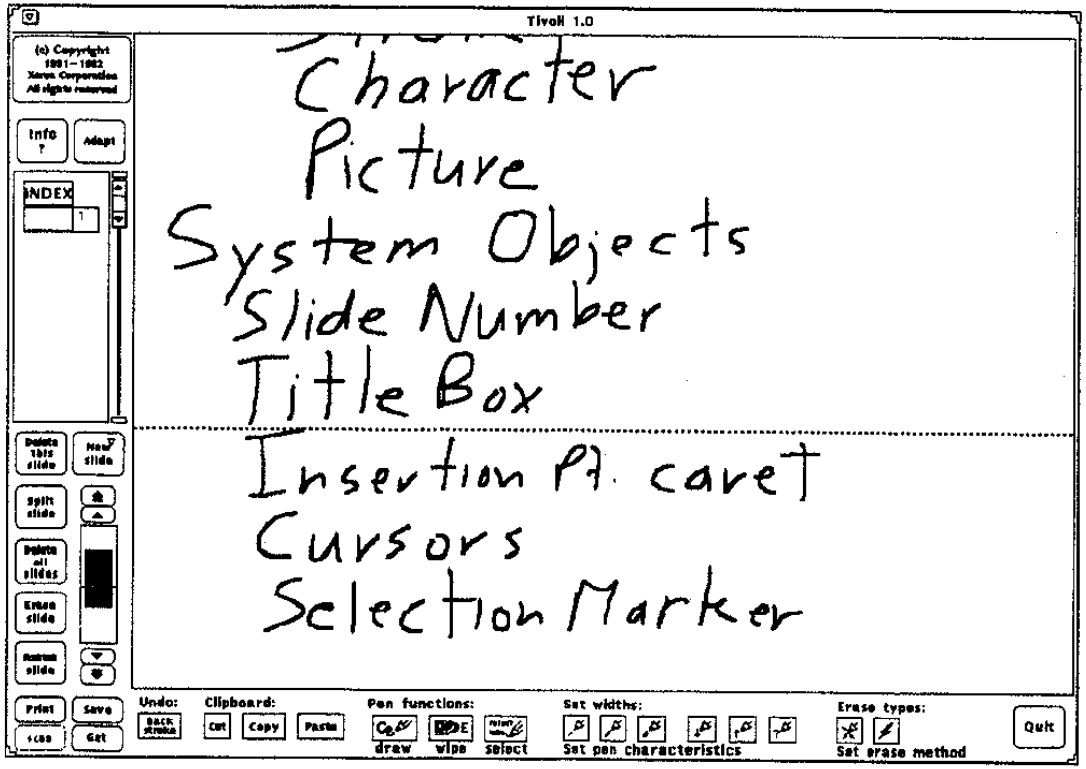


Key Tivoli Ideas

- Recognize **implicit structure** of freeform material on the board
  - In other words, don't recognize text, but look for natural groupings in digital ink
  - Then, recognize only a **small handful** of input strokes that allow user to make operations on these
- **Examples:**
  - Strokes are automatically geometrically grouped based on position with each other and whitespace between groups
  - Selection gesture (circling) easily allows selection of entire groups of strokes
  - Drawing a line between lines of text splits them apart to make room for new text.

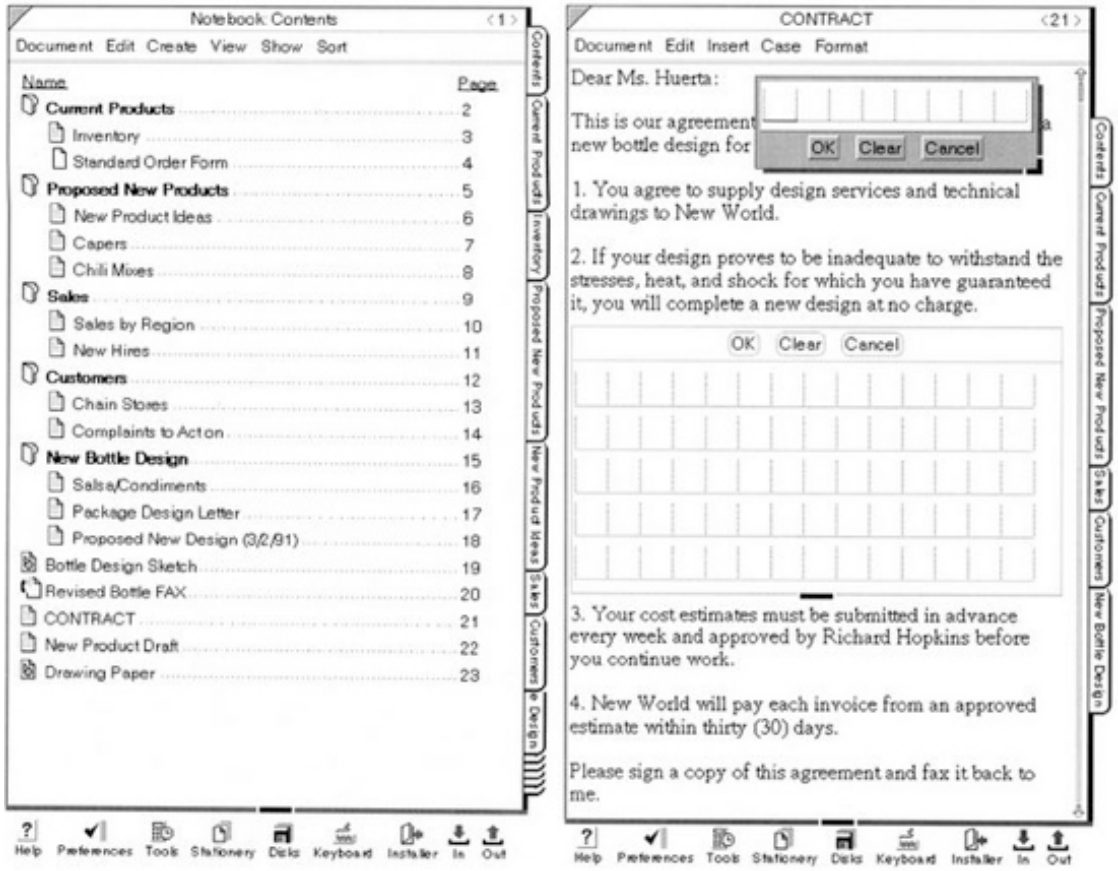
Tivoli Examples

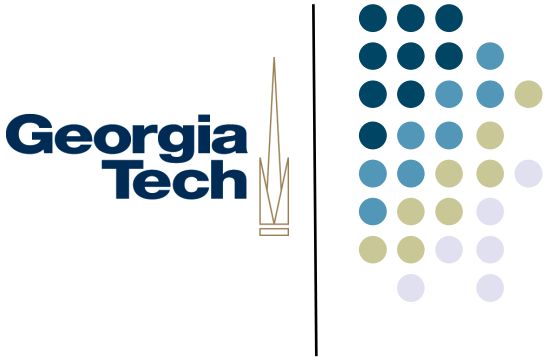




Penpoint OS

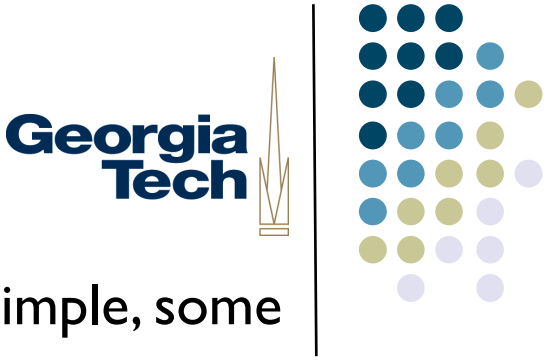
- Pen-specific OS, created from the ground up by Go Corporation
- Ink organized into "notebooks" and, for the most part, unrecognized
- However, certain gestures integrated into the OS for manipulating digital ink
  - Circle to edit, X to delete, caret to insert
- Special entry fields for text that should be recognized





5. Stroke Input: Recognizing and Interpreting Digital Ink

Recognizing Digital Ink



- A variety of recognition algorithms are available: some simple, some complex (we'll discuss a few in class…)
- Some work for full-blown handwriting, others are limited to recognizing certain fixed shapes or symbols
- Generally: the more complex and featureful the recognizer, the more you can use it for **content** recognition. Simpler recognizers are useful mostly for recognizing a handful of **commands.**
  - Content doesn't mean just text. Also drawing cleanup, sketch beautification, etc.
    - early storyboard support (SILK, Cocktail Napkin)
    - sketch recognition (Eric Saund, PARC; others)

- **Thus, the choice of recognizer (and power of the recognizer) impacts the user interface**

- TIP: you can do a whole lot with out needing a "real" recognizer

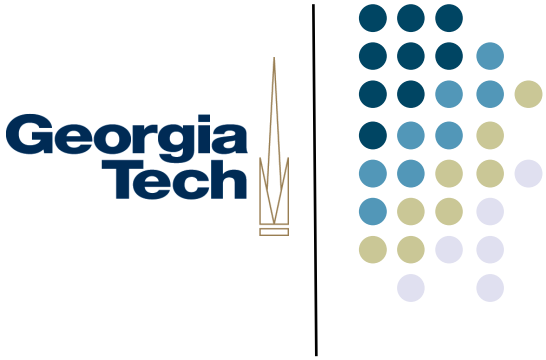
Example: Graffiti/Unistroke

- From Palm and/or Xerox
- Innovation: make the recognition problem easier by making the user adapt her behavior
  - So, not quite "natural" media
  - Simple alphabet of "unistrokes"
    - Each time the pen goes down, assume a new stroke; don't need to worry about multistroke recognition
    - Close enough to alphabet letters to be memorizable quickly
    - Yet easy for the computer to distinguish reliably.
- See "Touch Typing with a Stylus," D. Goldberg, C. Richardson. CHI 1993

Example: Flatland

- Main ideas:
  - Whiteboards should be "walk-up-and-use" (in other words, don't need to a launch a special app or tell the system how some content you're about to draw should be processed)
  - BUT then allow interpretation to be applied to digital ink strokes after you've made them
- Domain-specific recognizers, for a variety of tasks
  - Drawing beautification, map drawing, list management, simple arithmetic

Handwriting (content) recognition



Lots of resources

- see Web
- good commercial systems

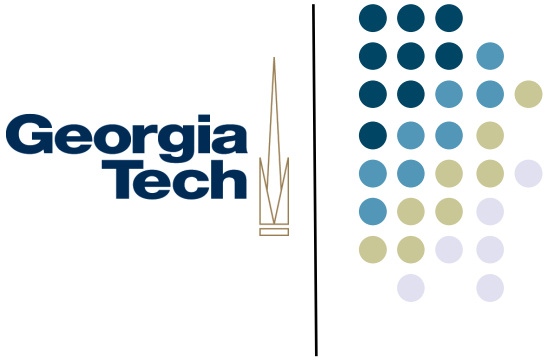
Two major techniques:

- on-line (as you write)

- off-line (batch mode)

Which is harder?

Handwriting (content) recognition



Lots of resources

- see Web
- good commercial systems

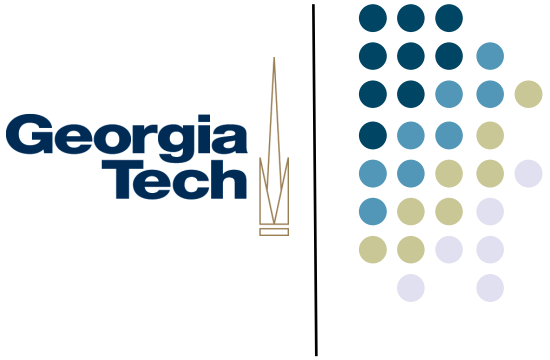
Two major techniques:

- 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.



Other Issues in Pen Input

Mixing modes of pen use

Users want free-form content and commands

How to switch between them?

Explicit:

- have an explicit mode switch, a la Graffiti (make a special command gesture preceding a stroke that should be interpreted as a command)
- special pen action switches that produce a temporary or transient model, e.g., the barrel switch on the Wacom pen

Implicit:

- Recognize which "mode" applies based on context of the stroke, e.g., Tivoli, Teddy, etc.

"Gorilla Arm"

- Challenge when using vertically-oriented touch screen (or pen input).
- Prolonged use results in fatigue/discomfort.
- Credited with a decline in touch/pen input in the 1980's (think desktop CRTs) that wasn't completely resolved until very light portable devices appeared.
- Some desktop touch interfaces have made a comeback with Windows 8 however!

Error correction

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

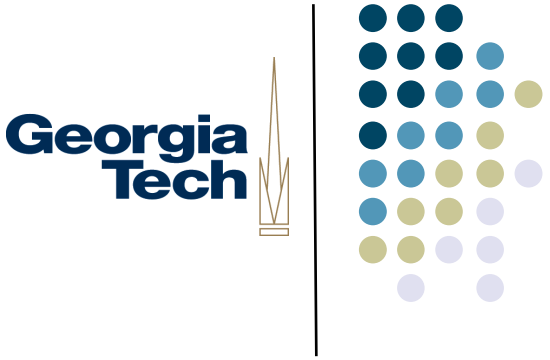
UI implications: even small error rates (1%) can mean lots of corrections, must provide UI techniques for dealing with errors

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



Resources

Toolkits for Pen-Based Interfaces

- SATIN (Landay and Hong) – Java toolkit
- GDT (Long, Berkeley) Java-based trainable unistroke gesture recognizer
- OOPS (Mankoff, GT) error correction