Input: Interaction Techniques

Administration

- **Questions about homework?**

Interaction techniques

- A method for carrying out a specific interactive task
  - Example: enter a number in a range
    - could use… (simulated) slider
    - (simulated) knob
    - type in a number (text edit box)
    - say it out loud (speech recognition)
  - Each is a different interaction technique
  - Instances of interaction styles

Interaction techniques in libraries

- Generally interaction techniques now come in the form of "widgets", "controls", "components", "interactors"
- Typically in reusable libraries
  - e.g. widget sets / class libraries
  - Big win in producing software
- Also need custom ones

Interaction Techniques

- Addresses complete cycle of execution and evaluation
- Typically includes
  - (simulated) input device
  - Mapping of input signal to semantics
  - Feedback to user
  - (simulated) output device

Design of interaction techniques

- Three things to pay the most attention to:
  - Affordance
    - Most important for novices
  - Feedback
    - Important for all
  - Performance (mechanics)
    - Feel and difficulty
    - Most important for experts

Mechanics: difficulty and "feel"

- Good models predicting physical movement difficulty (e.g., Fitts' law)
- "Feel" is trickier
  - Can depend on physical input device
    - physical movements, forces, etc.
  - Really gets back to the difficulty of the movement, but harder to characterize
- Important for all, but especially experts or people who are going to use a technique over and over again

Fitts' law (if you haven't seen it before)

$$\text{Time} = A + B * \log_2(\text{Dist}/\text{Size} + 0.5)$$

- Predicts time to move
- Time is linearly proportional to log of "difficulty"
  - proportionality constants depend on muscle group, and device
  - Difficulty controlled by distance & required accuracy (size of target)

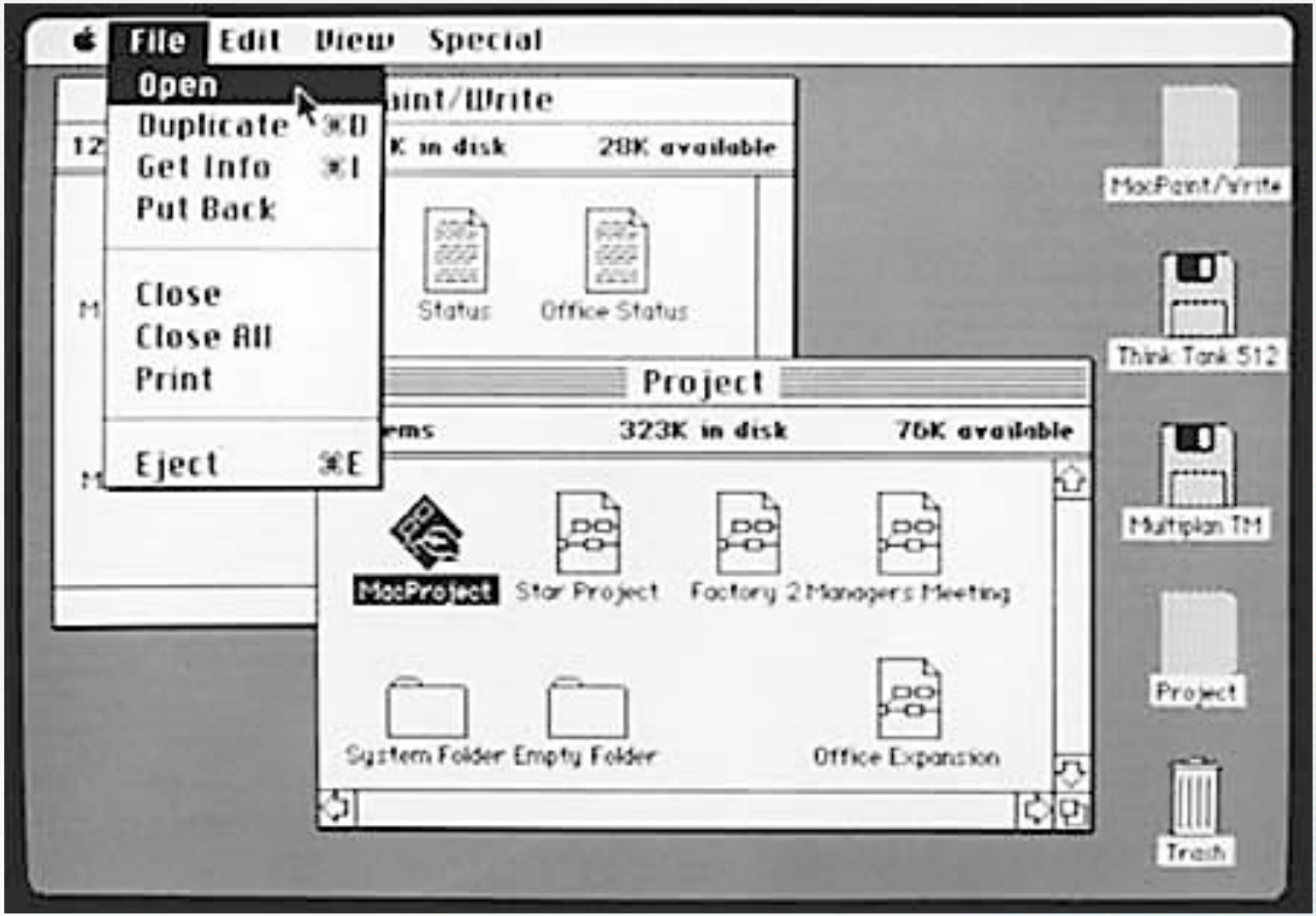
Fitts' law

- (True) expert performance tends to be closely related to time required for movements
  - not that closely related to learning (or overall performance) of novices
  - still need to consider "cognitive load" of performing some motion/ selection

Fitts' law

- Actual numbers from Fitts' law generally not all that helpful
  - that level of detailed analysis is hard
- General guideline
  - Keep required movements (accuracy & distance) firmly in mind
    - Avoid device swapping
    - Avoid disturbing focus of attention

Mini case study #1
The original "Macintosh 7"



Mini case study #1
The original "Macintosh 7"

1984—Mac 128K

- Macintosh (1984) was first big success of GUIs
  - originally came with 7 interactors built into toolbox (hence used for majority)
- Most not actually original w/ Mac
  - Xerox Star (+ Smalltalk & earlier)

The Macintosh 7

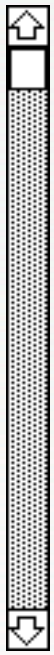
- Generally very well designed (iterated with real users!)
  - very snappy performance
    - dedicated whole processor to updating them (little or no "OS")
- Huge influence
  - These 7 still cover a lot of today's GUIs (good and bad to that)

Button

- **Shaped as rounded rectangles**
  (compare to "modern" boxish look…)
- **Inverted for feedback**
  - **Recall Mac was pure B/W machine**
  - **Pseudo 3D appearance harder (and hadn't been invented yet)**
- **Affordance, feedback, performance?**

Slider

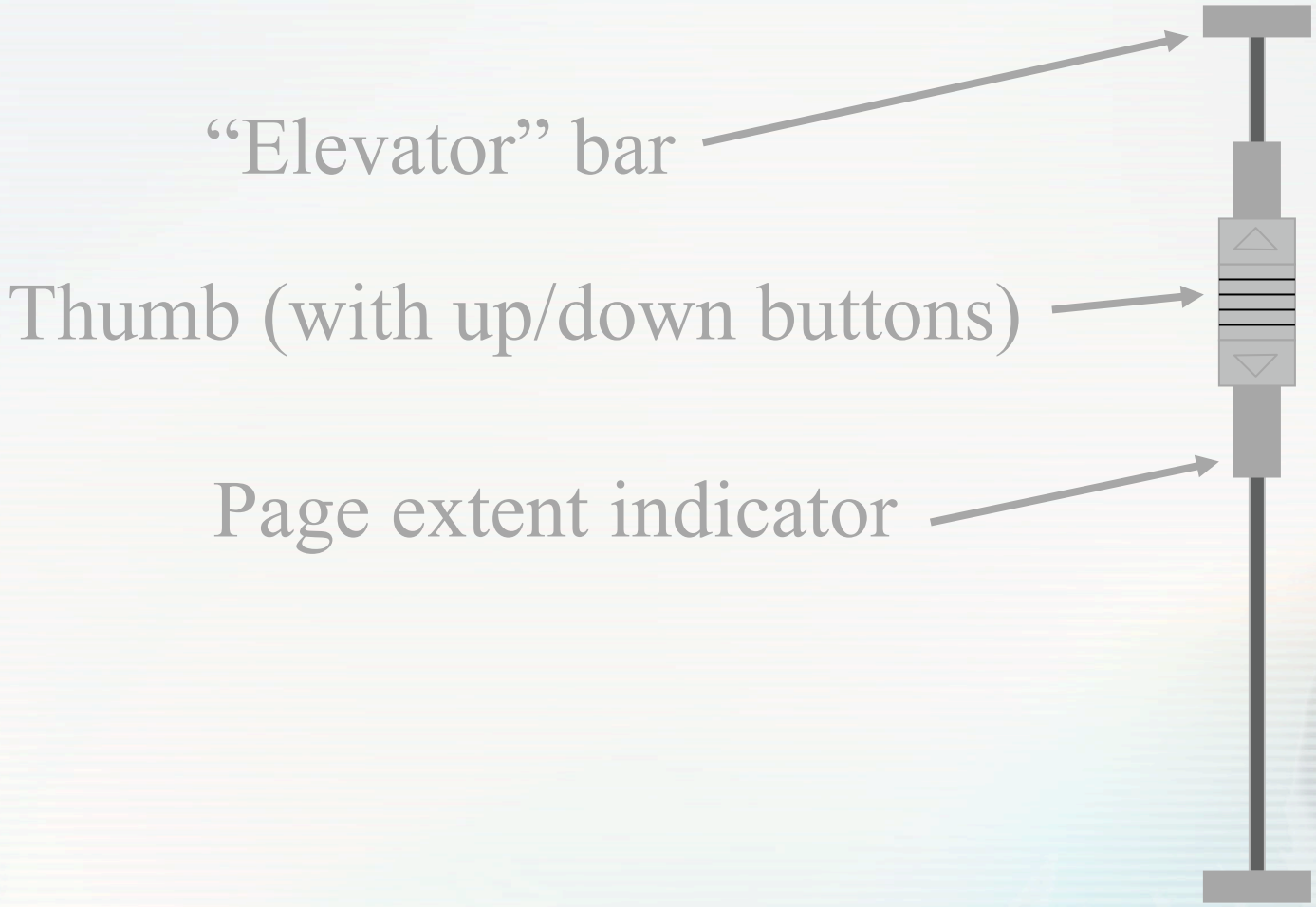
- Used for scroll bars (but fixed size "thumb")
  - Ridges on the thumb added later
  - "Pogo stick" problem



- Affordance, feedback, performance?

Aside: a different scrollbar design

- **Openlook scroll bar**

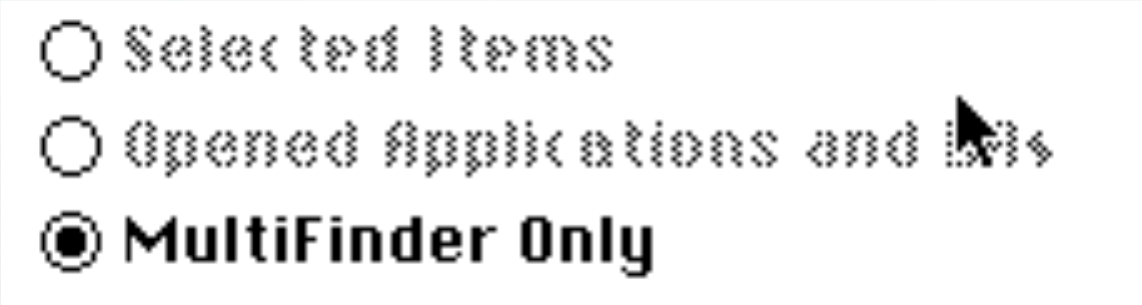


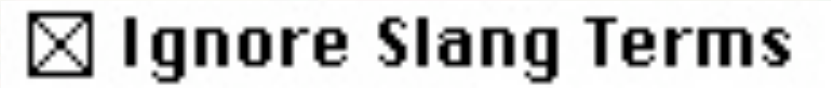
Pulldown menu

- This was original with Mac
- Differs slightly from Windows version you may be familiar with
  - had to hold down button to keep menu down (one press-drag-release)
- Items highlight as you go over
- Selected item flashes
- Affordance, feedback, performance?

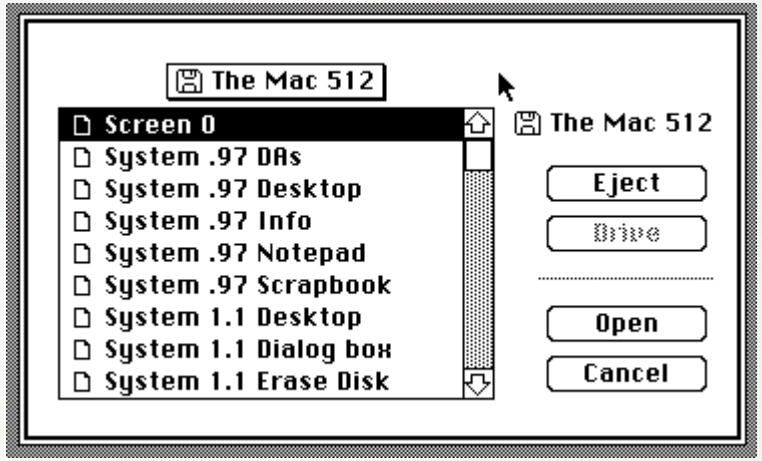
Check boxes, radio buttons, text entry / edit fields

- Pretty much as we know them
- Single or multi-line text supported from the beginning





File pick / save



- Much more complex beast than the others
  - built from the others + some
    - e.g. no affordance, but you could type and file list would scroll to typed name

Original Mac also had others

- Window close and resize boxes
- Drag & open file icons and folders
- Not made generally available
  - not in toolbox, so not (re)usable by other programmers

Second major release of Mac added a few

- Lists
  - single & multiple selection
  - from textual lists (possibly with icons)
- Hierarchical ("pull-right") menus
- Compact ("in-place") menus
  - select one-of-N pulldown
- Window zoom box

Have seen a few more added since then

- Tabbed dialogs now widely used
- Hierarchical lists (trees)
- "Combo boxes"
  - Combination(s) of menu, list, text entry
- Typically don't see much more than that

Almost all GUIs supported with the above 10-12 interactor types

- Good ones that work well
  - uniformity is good for usability
- But, significant stagnation
  - "dialog box mindset"
  - opportunities lost by not customizing interaction techniques to tasks

Mini case study 2: Menus

- Menu
  - supports selection of an item from a fixed set
  - usually set determined in advance
  - typically used for “commands”
  - occasionally for setting value (e.g., picking a font)

Design alternatives for menus

- Simple, fixed location menus (see these on the web a lot)
  - easy to implement
  - good affordances
    - easy for novices (always same place, fully visible)
  - Focus of attention problems
  - Screen space hog

Popup menus

- Menu pops up under the cursor (sometimes via “other button”)
  - close to cursor
    - not under it, why?

Popup menus

- Menu pops up under the cursor (sometimes via "other button")
  - close to cursor
    - Performance: What does Fitts' law say about this?
    - Affordance and Feedback?

Popup menus

- Menu pops up under the cursor (sometimes via “other button”)
  - close to cursor
    - Fitts law says: very fast
    - also focus not disturbed
  - takes no screen space (until used)
  - can be context dependent (!)
  - poor (non-existent) affordance

Getting best of both: Mac pulldown menus

- Menu bar fixed at top of screen, with pull-down submenus
  - benefits of fixed location
  - provides good affordance
  - good use of space via partial popup
  - but splits attention & requires long moves

Fitts' law effects

- Windows menus at top of windows, vs. Mac menus at top of screen
  - Interesting Fitts' law effect
    - what is it?

Fitts' law effects

- Windows menus at top of windows, vs. Mac menus at top of screen
  - Interesting Fitts' law effect
    - thin target vertically (dir of move) ➔ high required accuracy
    - hard to pick
    - but… (anybody see it?)

- **Break 15 minutes (?)**

Fitts' law effects

- With menu at top of screen can overshoot by an arbitrary amount

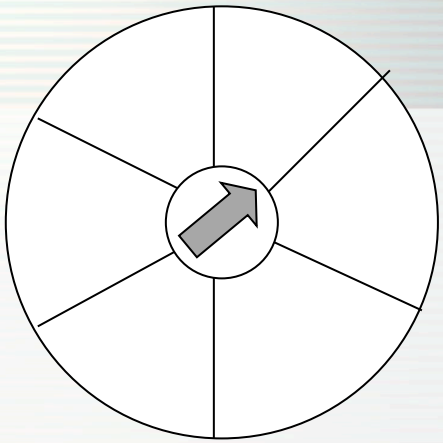
  (Example of a "barrier" technique)

  – What does Fitts' law say about that?

Fitts' law effects

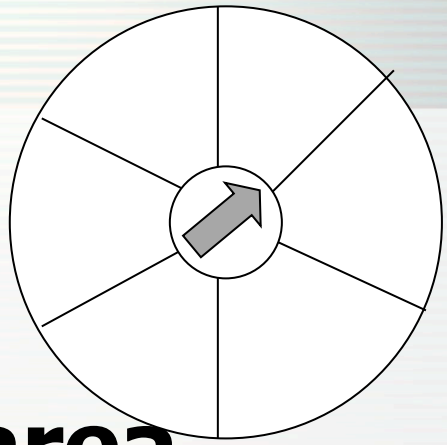
- With menu at top of screen can overshoot by an arbitrary amount
  - very large size (dominated by horizontal which is wide)
  - Original Mac had 9" screen so distance not really an issue
  - very fast selection

Pie menus



- A circular pop-up menu
  - no bounds on selection area
    - basically only angle counts
    - do want a "dead area" at center
  - Performance: What are Fitts' law properties?
  - Affordance and feedback?

Pie menus



- A circular pop-up menu
  - no bounds on selection area
    - basically only angle counts
    - do want a "dead area" at center
  - Fitts' law properties:
    - minimum distance to travel
    - minimum required accuracy
    - very fast

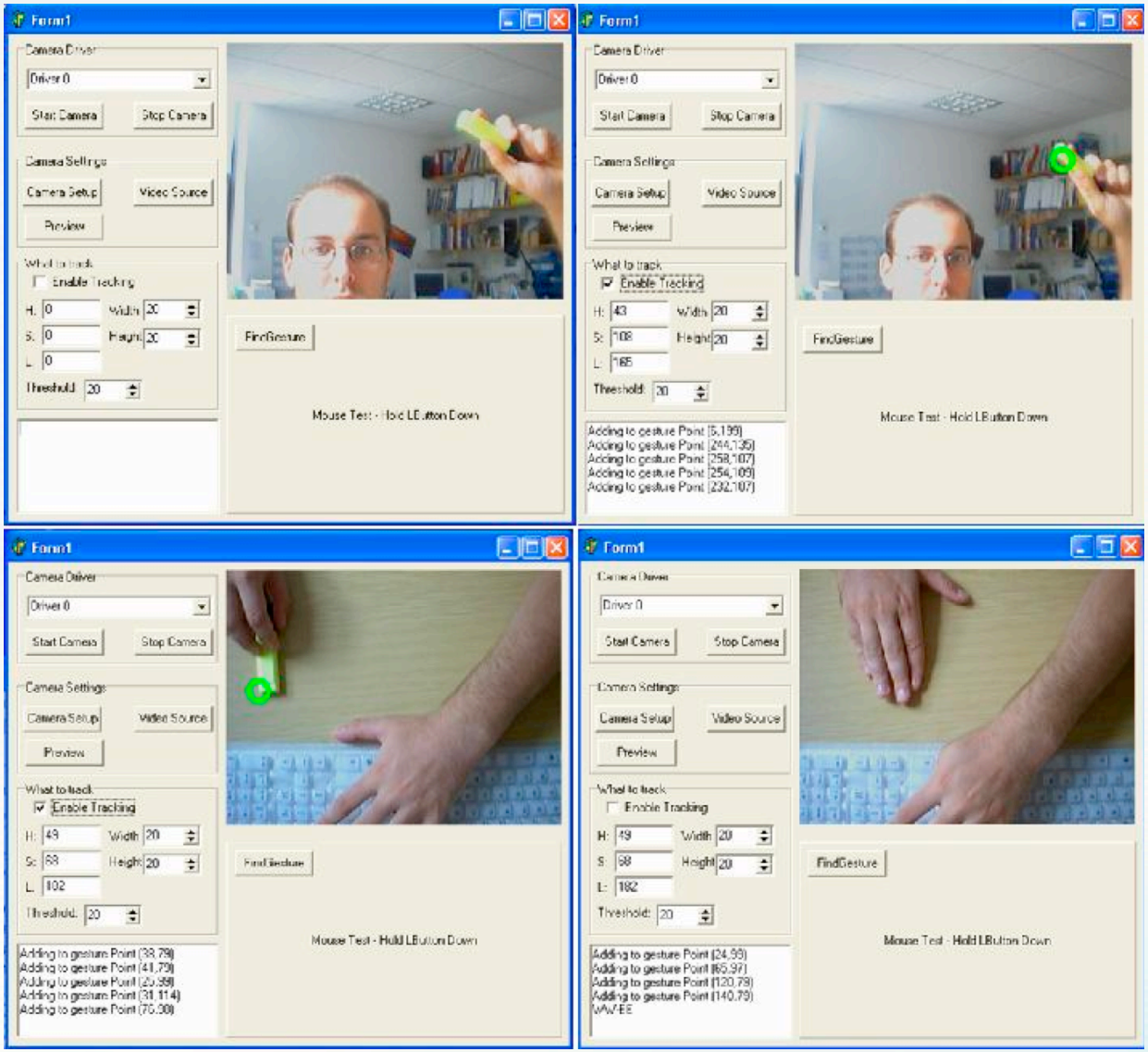
Pie menus

- **Why don’t we see these much?**

Pie menus

- Why don't we see these much?
  - Just not known
  - Harder to implement
    - particularly drawing labels
    - but there are variations that are easier
  - Don't scale as cleanly
    - Hard to do hierarchy

Pie Gestures



Beating Fitts' law

- Can't really beat it
  - property of being human
  - but you can "cheat"!
- One approach: avoid the problem
  - use a non-pointing device
    - shortcuts & fixed buttons
    - mouse wheel for scrolling

Beating Fitts' law

- Not everything can be a shortcut
- Other major approach: manipulate interface to reduce difficulty
  - distance (put things close)
    - but not everything can be close
    - have to make them smaller!

Beating Fitts' law

- Most ways to "cheat" involve manipulating size
  - typically can't make things bigger w/o running out of screen space (but look at that as an option)
  - but… can sometimes make things act bigger than they are

"Cheating" on target size

- Consider targets that are not just passive
  - not all movements end in "legal" or useful positions
  - map (nearby) "illegal" or non-useful onto "legal ones"
    - hit of "illegal" position treated as legal
      - e.g. positions above Mac menubar
    - effective size bigger

Snapping (or "gravity fields")

- Treat movement to an "illegal" point as if it were movement to closest "legal" (useful / likely)
  - Cursor or other feedback snaps to "legal" position
  - Drawn to it as if it has gravity

Snapping

- Simplest: grids
- Constrained orientations & sizes
  - 90° & 45°, square
- More sophisticated: semantic
  - only attach circuit diagram items at certain spots

Snapping

- Even more sophisticated: dynamic semantics
  - Check legality and consequences of each result at every move
    - don’t catch errors, prevent them!

Interaction Techniques

- Input device, mapping, feedback, output device
- Key issues of
  - Feedback, performance, affordance
- When choosing an interaction technique, tradeoff between task-specific and ease of implementation

Questions?