Menus, Forms, and Dialogs

- Another viable interaction style, alternative or complementary to direct manipulation
- These rely on *recognition* instead of *recall* — cognitively, an easier task for users to perform
- Particularly effective for:
  1. Users with little training
  2. Intermittent use of the system
  3. Users who are unfamiliar with task terminology
  4. Tasks that require structure in decision-making

As Always, Must Fit the Mental Model

- Menus, forms, and dialogs must still fit the user's real-world tasks:
  - Decomposition/categorization
  - Labeling and terminology
  - Distinctive and clear choices
- OAI model helps here too, on the same principle that interface objects must fit the domain objects…maybe not as literally as in direct manipulation, but the clear correspondence should still exist

Single Menus

- Yes, you may have thought of these as dialogs, but technically a row of buttons is a menu:
  - OK/Cancel (once known as "Do It/Cancel"); Yes/No
  - Save/Cancel/Don't Save; Yes/No/Cancel
  - Multiple-choice radio buttons (one of many choices)
  - List of check boxes (orthogonal choices)
- Single menus are "context-free" — they represent a standalone set of choices…but of course they are the building blocks for more complex menu structures

Pull-Down Menus

- Initially introduced by the Xerox Star, sent to the mainstream by the Macintosh as part of an overall desktop metaphor (pull down menus = drawers of related tasks); now used pretty much everywhere
- *Positional constancy* — while not all menu items apply to all situations, they should be *dimmed* instead of *removed* when not applicable, to maintain their location onscreen and help a user's spatial memory
- Keyboard shortcuts allow for efficiency when a user becomes familiar with the menu content

Toolbars, Iconic Menus, Palettes

- First there were palettes, representing tools for direct manipulation (such as on a drawing program) — note how they represent *state* as opposed to atomic *actions*
- Icons were then used for actual actions and/or commands (previously text-only menu items) — menus started having a picture associated with them, and icon-only menus took the form of *toolbars*
- Functionally speaking, toolbars and menus are actually the same — as noted in Windows interfaces
- Apple has strict rules for including icons in menus

Popup Menus and Their Variants

- Context-sensitive menus — in many respects, a better term is "contextual menu"
- They appear in place, and are generally specific to the object at the location from which they are invoked
- Pie menus — popup menus with items arranged radially around the context object (The Sims)
- Marking menus — popup menus that associate items with the "paths" that lead to them (Maya)
- FlowMenu — integrates data entry with menu

Menus for Long Lists

- Various approaches for handling menus that exceed the available real-estate
- Familiar examples: scrolling menus, keyboard navigation (e.g. first-letter jumping), combo boxes (i.e. combine text field with menu), sliders (an interesting combination of direct manipulation and menu styles), incremental disclosure (i.e. "5 more items…")
- Less familiar: fisheye menus, alphasliders
- Or, add a dimension — spread menu items over both *x* and *y* axes

Embedded Menus/Hotlinks

- *Embed* menu choices in an overall display or context, not as an *explicit* list of choices
- Straddle the line between menu and direct manipulation interaction styles
- When embedding is well-done (e.g. clickable maps, onscreen calendars, interactive seating charts, hyperlinks in text or images), this helps to clarify what menu items mean or what they will do
- One can think of Web pages as embedded menus

Combining Multiple Menus

- Linear menus: sequential presentation, though sometimes sequence is only organizational and does not imply strict precedence
  - Main idea is to present one decision at a time, particularly for novice or intermittent users
- When precedence is not really necessary, multiple simultaneous menus may be presented instead — allows decision-making in any order
  - But they take up a lot more screen space

Tree-Structured Menus

- Used for particularly large lists of items — we're talking thousands of items here
- Hierarchical decomposition and categorization allow for [potentially] easier navigation and decision-making — like with algorithms, they convert linear searches into logarithmic searches, if decomposed well
- Menu level traversal may *expand* or enforce a *sequence*
- Studies have shown that breadth is preferable over depth — makes sense intuitively, as this leverages the logarithmic nature of tree structures

Organizing Menus

- This is essentially applying theory to form principles and guidelines — we know that a "good" menu corresponds well with a user's mental model…but how, specifically? Let us count the ways…
- *Task-related grouping*
  1. Cluster menu items as a user might cluster them
  2. Provide choices that cover all possibilities
  3. Don't overlap (or possibly overlap) choices
  4. Use familiar but easily distinguishable terminology

Menu Item Sequencing

- When there is a natural sequence, use it (e.g. time, number, physical correspondence such as size)
- When there is no natural sequence, choose from a number of approaches:
  1. Alphabetic sequencing (but how about i18n?)
  2. Linear grouping (e.g. separators, white space)
  3. Frequency of use
  4. Importance of item — though this can be relative
- Studies have shown that alphabetical works best when users have a "target word," but groupings are better when task search is more conceptual

“Frequency” and “Importance”

- Except in the most obvious cases, menu item “frequency” and “importance” are very relative and user-individualized concepts
- In many designs, “frequent” or “important” items are already highlighted by their repeated presence — such as in both the menu bar and in a toolbar
- “Adaptive” interfaces were all the rage once — that is, dynamically modify menu sequencing as the user works with an application

Caveat Adaptor

- Menus (and other user interface elements) that “morph” as the user works are a textbook example of a tradeoff that isn’t worth it — adaptive menus removed a menu’s *stability* — the user didn’t know where something will be the next time they need it
- In the end, this wasn’t worth the benefit of having a “true” clustering of frequently-used items
- Current user interfaces just provide both, perhaps redundantly — specific sections that are explicitly adaptive (e.g. “Open Recent,” partitioned font menus)

Menu Layout, Appearance, and Behavior

- Not much experimental evidence here, mostly heuristics and "common sense" guidelines
- Menu titles — consistent grammatical style; in multiple-menu environments, use the same phrase for menu items and the sections/states that they lead to
- Menu item phrasing — familiar and consistent terms; distinctive item text; consistent and concise phrasing; identify the *key word* and put that in front
- Graphic/visuals — generally a consistency issue, typically dependent on platform-specific guidelines

Kicking It Up a Notch

- Pointer-based menu navigation may be fine for novices, but as a user becomes more proficient they will want faster ways to invoke menu items
- *Keyboard shortcuts*: two kinds overall; using Swing's terms, they are "mnemonics" and "accelerators"
  - Mnemonics are like keyboard paths to an item — pick the menu mnemonic first, then drill down
  - Accelerators are direct routes — they invoke the menu item immediately
- Other techniques include *mouse ahead* in marking menus, *bookmarks*, *macros*, and *tear-off menus*

More Generalized Data Entry — Forms, Dialogs

- The form fill-in interaction style present a complete "package" of information required from the user
  - Maps well to real-world paper forms
  - Clean "transaction" where the user can review all entered information before proceeding
- Form fill-in interaction style is usually in the context of a *dialog* — that is, a multi-step "conversation" with the user where the system requests information before proceeding, then acts upon the provided information before going to the next step, if any

Form Fill-In Design Guidelines

meaningful title	comprehensible instructions	logical grouping and sequencing of fields
visually appealing layout of the form	familiar field labels	consistent terminology and abbreviations
visible space and boundaries for data-entry fields	convenient cursor movement	error correction for individual characters and entire fields
error prevention where possible	error messages for unacceptable values	immediate feedback
marking of optional fields	explanatory messages for fields	completion signal to support user control

Format-Specific Fields

- While we always enter strings at the raw field level, these strings sometimes represent non-string concepts — that is, strings are *artifacts* of a different abstraction or information unit: phone numbers, social security numbers, times, dates, currency
- This requires parsing, which implies formatting
- Guidelines for display and behavior exist, but some data types are intrinsically difficult (e.g. dates)
- Sometimes, a direct manipulation widget helps eliminate possible formatting errors

Dialog Boxes: Forms-in-a-Box

- Generally synonymous with forms except for context — dialog boxes frequently encapsulate common system tasks, or functions shared by many applications: Open/Save, Print/Page Setup, Find, Font, Color
- Thus, many environments provide standardized dialog boxes for consistency purposes, with some customization (in a controlled fashion)
- Another dialog box concept: *modality* — does a dialog box put a hold on the current task? And if so, on what level — system? Application? Document?

Thoughts on Modality

- General rule — try to avoid modes; they limit the user's autonomy
  - Modes used to be an implementation issue — they added constraints that made things easier to code
  - Not a huge deal today; for example, are these functions truly modal: about boxes, preferences, font selection, color selection?
- Mode "scope" is also evolving — used to be either system- or application-level; in current platforms, system-modal dialogs are almost unheard of, and a new "document" scope has emerged (e.g. "sheets")

Dialog Layout	External Relationships
Meaningful title, consistent style	Smooth appearance and disappearance
Top-left to bottom-right sequencing (for most languages)	Distinguishable but small boundary
Clustering and emphasis	Size small enough to reduce overlap problems
Consistent layouts (margins, grid, whitespace, lines, boxes)	Display close to appropriate items
Consistent terminology, fonts, capitalization, justification	No overlap of required items
Standard buttons (OK, Cancel)	Easy to make disappear
Error prevention by direct manipulation	Clear how to complete/cancel

Emergent Ideas

- Hybrid direct manipulation/menu components: control menus, marking menus, FlowMenu, Toolglass
  - Some less researchy components that are more familiar: buddy lists, Exposé, task bars/docks
- Audio menus — for accessibility or interfaces without a visible display
- Small displays — cell phones, portable devices, PDAs have unique constraints due to their size — "Less is More" becomes particularly important

Information Appliance Design Considerations

Mohageg, Wagner from Bergman, 2000

1. Account for target domain
2. Dedicated devices mean dedicated user interfaces
3. Allocate functions appropriately
4. Simplify
5. Design for responsiveness