Lecture 13: User experience II

Lecture goals
- Construct appropriate UI prototypes to iterate on design
- Bridge mental and computer models
- Evaluate UI designs against established principles
- Design interfaces within the constraints of web browsers and mobile devices
- Evaluate UI designs with user testing

User experience (continued from Lecture 12)

Dark patterns
- Many of our experiences with UI are in a marketing context
  - Goal is to maximize engagement and manipulate user decisions
  - Being commonplace and effective in marketing goals does not make a design pattern good
    - Avoid simply aping features of slick websites (even if libraries make it easy to do so)
- User-centric design
  - Interface should facilitate, not redirect, users’ objectives

Analyze/design/build/evaluate loop

Development processes
- Written requirements poor fit
  - Requirements benefit from sketches, comparison with existing systems
  - Designs should include graphical elements, benefit from prototypes
  - Don’t try to capture everything, but consider capturing justifications when client gives feedback or chooses between options
- UI must be tested with users; expect requirements and design changes
  - Schedules must include time for testing and time to make changes

UI prototypes
- Preliminary version used to iterate rapidly between requirements and design
  - Minimize polishing effort to maximize iteration speed
- Paper sketches
  - Lowest effort, so amenable to major changes
- Wireframe
  - Outline layout
- Mock-up
Graphic designs with detailed layout, color
- Operational prototype
  - Enables interaction and navigation

Interactive prototypes
- "Clickable" - responds in limited ways to user interactions
  - Illustrate time-dependent design
    - Animations
    - Drag-and-drop
  - Navigate between pages, dialogs
- Not production code
  - Does not update model data, trigger external events
  - Make sure client understands limitations
- Collaborative tools:
  - Figma
  - Adobe XD
  - (many others)

Models
Relating user and system models
Mental model
- User's view of system and the UX it provides
- May include physical metaphors for digital interactions
- Examples:
  - Pieces on a game board
  - File folders and desks

Program model
- Data, relationships, and functions making up the system
- Examples:
  - Object identity & coordinates, rules constraining movement
  - Tree of data units with metadata

Model mismatches
Model-view-controller (as a "model")
Layer 0: Computer systems and networks
- Performance, reliability, predictability of systems have a large impact on user experience
- Interfaces may be designed for specific hardware capabilities and constraints
  - Screen sizes, input devices, sensors, graphics/multimedia processing
  - Later: Adapting to constraints of web browsers and smartphones
Layer 1: Model
- Provides all functionality of program except for user interaction
  - Program logic, services
  - Data structures, file systems
  - Content (text, graphics, audio, metadata, etc.)
- Beware: easy for clients, designers to specify new behavior that is not supported by existing model

Separation of content from view

Layer 2: Control (navigation)
- Controller manages flow of application
  - Controls navigation between various "displays"
    - Web pages, window forms, pop-up dialogs, app screens
  - Updates model, view in response to user interaction
- Controller role varies between implementations

Layer 3: View (user interface)
- Appearance of displays and facilities for interaction
  - Graphical elements (fonts, colors, icons, images, animations)
  - Control widgets (text boxes, menus, buttons, sliders)
  - User input (touchscreen, gamepad, keyboard & mouse, buttons & knobs)
- For a quality user interface, teams need someone skilled in graphic design

Design principles
UI design principles
- UI design is partly an art, but some general principles apply:
  - Consistency (in appearance, interaction, function)
  - Feedback (what is the system doing? why does the user see what they do? what is about to happen?)
  - Ability to interrupt or reverse actions
  - Comprehensible and non-destructive error handling
- The user should feel in control (not like they're being controlled)

Example considerations: navigation menus
Advantages
- Easy for users to learn and use
- Avoids certain categories of error

Challenges
- How to handle large number of choices?
- Scrolling menu (e.g. lists of countries or states)
Users typically prefer menu systems that are broad and shallow (rather than deep)

Design choices: text vs. graphics

Text
- Precise, unambiguous (hopefully)
- Fast to compute, transmit

Graphics
- Quick to comprehend, learn
- But icons may be difficult to recognize
- Variations can show different cases

Command line interfaces
- Limitations of GUIs
  - Only suitable for human users (difficult to automate)
  - Awkward to control complex interactions (difficult to compose)
- Command line interfaces (CLI)
  - User interacts with system by typing commands
  - Composable
  - Scriptable
  - Can be adapted for users with disabilities
  - Amenable to formal specification
  - Usually requires learning or training

Web and mobile interfaces

Device-aware interfaces
- How does a laptop computer differ from a desktop?
- What is special about a smartphone?

Web and mobile apps
- Must consider network
  - Transfers may need to be asynchronous to hide latency
    - Need visual feedback that operation is in progress
    - Should support cancellation
  - Connections may be unreliable
    - Should be robust to duplication
Leverage simulation
- App development environments (e.g. Xcode, Android Studio) allow you to simulate screen sizes, touch events
- Web browser developer tools allow you to simulate screen sizes, network speed

Test for accessibility

Responsive design
- Automatically adjust user interface based on size of screen (or other device properties)
  - Beyond simple layout scaling – can completely change layout to accommodate device
  - Use CSS media queries to select different style rules in different situations