Final Exam Review

Winter 2017

What to Expect

- Saturday April 22, 4:00 to 6:30 PM (2.5 hours)
- DC 1350,1351 – assigned seats (Odyssey)
- No aids allowed (i.e. no calculators, books)

- Scope: The entire course, focusing on lecture material. The final exam is comprehensive and covers *everything*.
- Format: Question and Answer
  - Short to medium answers
  - Possibly multiple choice and/or fill-in-the-blank.
    - Concepts
    - Problem-solving
    - Short coding (reading existing code, writing small amounts of pseudocode, NOT writing syntactically correct or complex code).

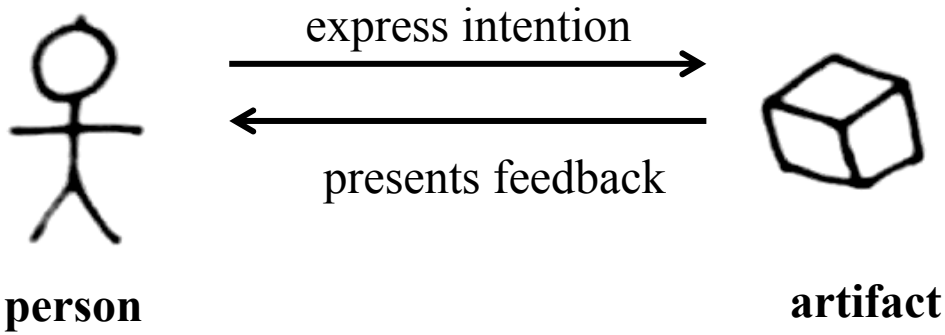
- Introduction
- History
- Windowing Systems
- Drawing
- Events
- Modern GUI Systems
- *Intro to Java (not on exam) **
- Widgets
- Event Dispatch
- Event Handling
- Layout
- 2D Graphics
- Affine Transformations
- Model-View-Control (MVC)
- Input Devices
- Input Performance
- Direct Manipulation
- Features; Undo
- Features: Cut/Paste, Drag/Drop
- Features: Responsiveness
- Touch Interfaces
- Touchless Interfaces
- *Wearables (design only) **
- *Case Study: Mobile Development (design only) **
- *Case Study: Web Development (design only) **
- Design Principles
- Visual Design
- Accessibility

* not on the exam, or limited coverage

How has the role of a UI programmer changed in the last 40 years?

What is a User Interface?

- The vehicle through which a person expresses intention to an artifact, and receives feedback from the artifact.



What is the difference between an interface and an interaction?

- External presentation vs. behavior, unfolding over time.

History

Paradigms of interaction

- Batch interfaces
- Conversational interfaces
- Graphical interfaces

What is the difference between an interface and an interaction?

- External presentation vs. behavior, unfolding over time.

Four visionaries

- Vannevar Bush: “As We May Think”, Memex
- Ivan Sutherland: Sketchpad, Lightpen, DM
- Douglas Engelbart: Mouse, Hypertext, Copy/Paste
- Alan Kay: Xerox Star, Dynabook

What are the characteristics of a GUI interface? What is a WIMP interface?

What are some of the design goals of the X11 windowing system?

- Display and device independent
- Network transparent
- Support multiple concurrent displays
- Support overlapping windows, 2D graphics, high-quality text

What is a X Client? X Server? What led to this design?

What is a Base Window System, and what is it's role?
How is it different from a Window Manager?

What are some problems that might arise when drawing on a display?

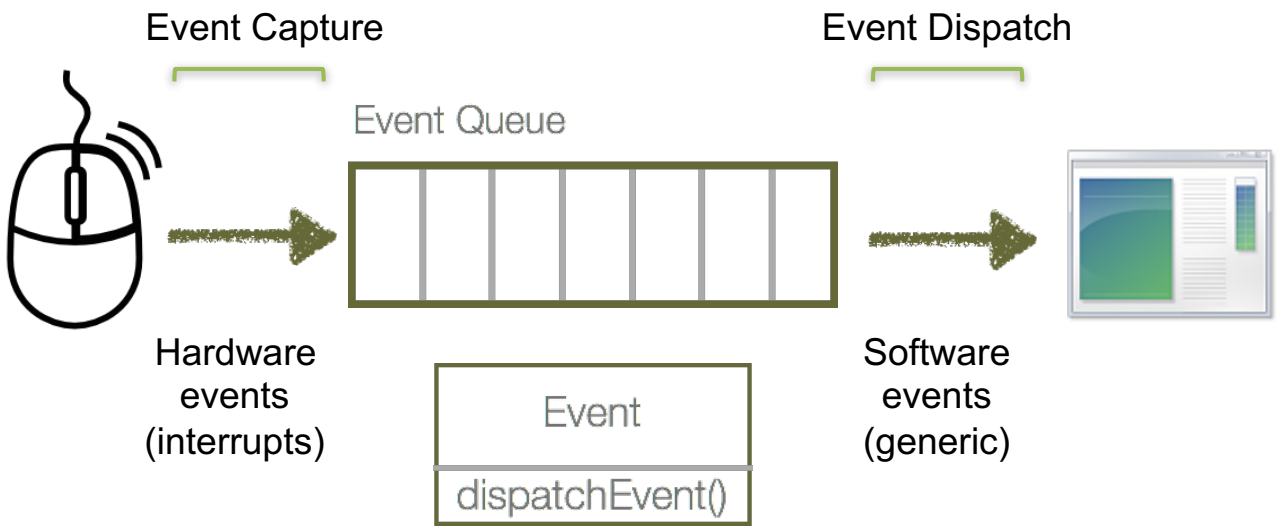
- Clipping regions
- Depth and ordering (Painter's Algorithm)
- Double-buffering

What is event-driven programming?

- "Event-driven programming is a paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs/threads."

How does the Base Window System handle events?

- Packaging (bundle into a structure) and place on **event queue**
- The **event loop** is the primary mechanism for event dispatch.
  - XNextEvent vs. XPending, XSelectInput (masks)



Event Dispatch

1. **Event dispatch**: Getting events to correct widget
2. **Event handling**: Running code for an event
3. **Notifying view and windowing system**: Model-View-Controller, and view notification.

Widgets are often the target of events

- Laid out in a hierarchy (interactor tree).
- Lightweight vs. heavyweight widgets and toolkits?

Which widget receives an event?

- Positional dispatch: top-down vs. bottom-up
- Focus dispatch: key, mouse

What are some problems with positional-dispatch (e.g. scrollbar)? How does focus dispatch help address these limitations?

Event Handling

After dispatch to a widget, how do we bind an event to code?

Design Goals

- Easy to understand (clear connection between event/code)
- Easy to implement (binding paradigm or API)
- Easy to debug (how did this event get here?)
- Good performance

Mechanisms

- Event Loop & Switch statement binding
  - All events bound by a single loop (switch statement by type)
- Inheritance binding (Java 1.0)
  - Override base class methods in each widget
- Event delegates and listeners (Java 1.1+)
  - Loose coupling
- Delegate binding (C#/.NET)
  - Type-safe function pointers (secure, multicasting)

Layout

Layout means dynamically adjusting screen composition.

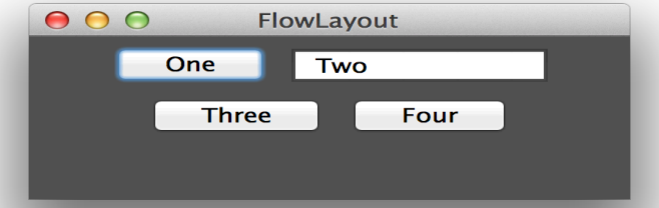
- Designing a spatial layout of widgets in a container
- Adjusting that spatial layout when container is resized

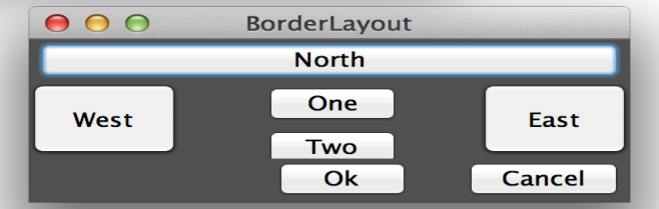
A Layout Manager provides a layout algorithm to size and position widgets (uses widgets preferred, min and max size)

Composite vs. Strategy design pattern: how are they used?

Layout strategies (Java layouts)

- Fixed layout (null)
- Intrinsic size (box, flow)
- Variable intrinsic size (border)
- Struts and springs (spring, box)





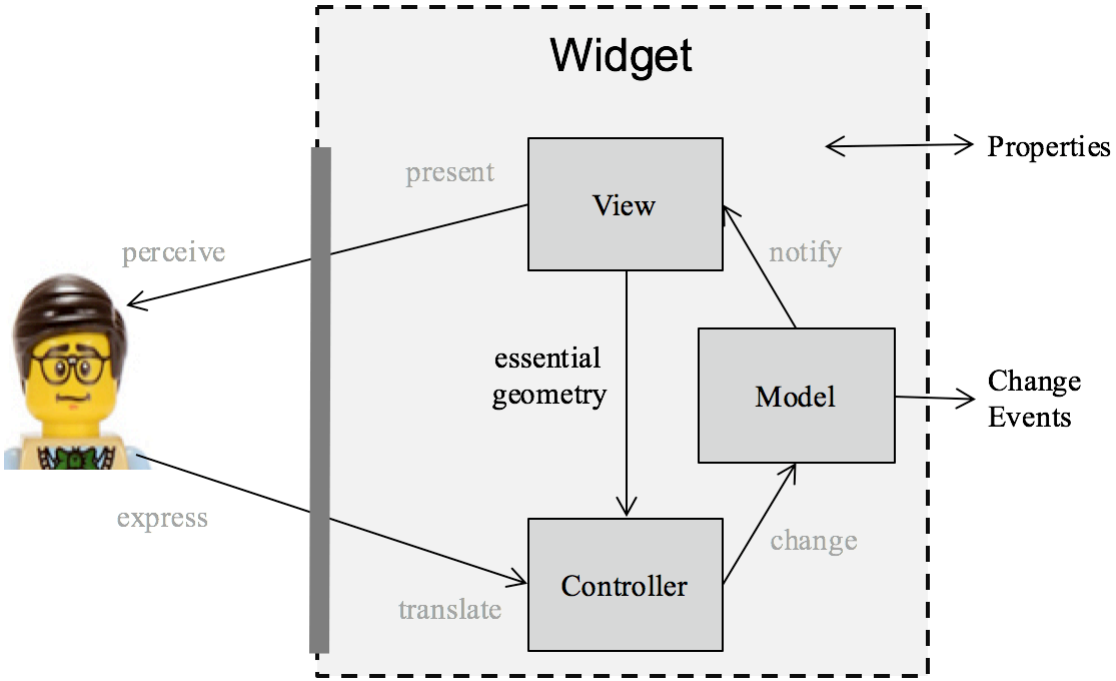
Widget is a generic name for parts of an interface that have their own behavior (e.g. buttons, progress bars)

- Draw themselves, and receive and interpret their events.
- Simple vs. container widgets, custom-controls

A **logical input device** is a graphical component, defined by its function – what kind of input primitive it manages.

- Each logical input device has a model, event and appearance (e.g. Logical Button Device: no model, "pushed" events).

MVC also applies at the widget "level", since widgets can have their own state, events, and appearance.



How do we represent and draw primitives?

- Shape models (array of points, location, bounds, …)

How do we manipulate shape models?

- Affine transformations! **Rotation**, **Scaling**, **Translation**
- Represent combined transformations using matrix notation
- Can combine successive operations but order matters!

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

How do we use affine transformations when drawing widgets in the interactor tree?

How do they help when managing mouse coordinates?

Model-View-Controller (MVC)

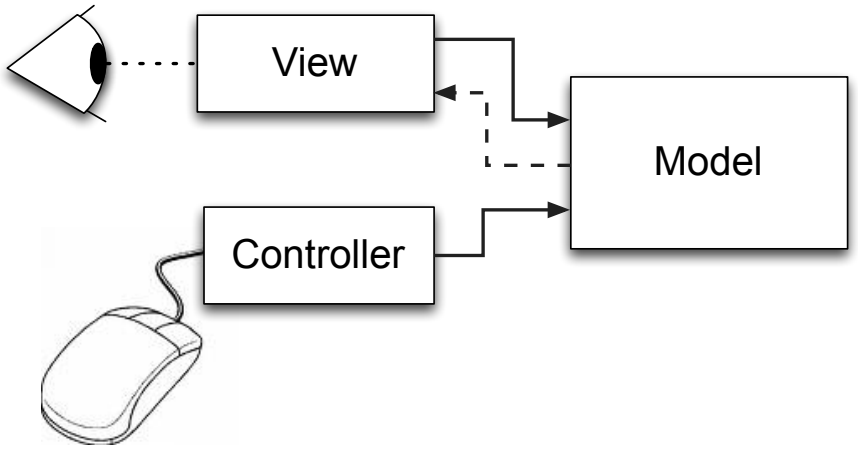
Why do we need Model-View-Controller (MVC)?

- Need multiple views of the same document
- The UI will change over time (e.g. introduce mobile app).
- Supports code separation and reuse, easy testing.

What is MVC?

- Mechanism that promotes loose-coupling between the Model (data), View (output) and controller (input).
- Instance of *Observer* design pattern

How do Model, View and Controller communicate?



The View and Controller are often tightly coupled (i.e. view class contains both). Why?

Responsiveness

User experience is affected by how responsive an application *appears* to be to the user. Two methods to achieve responsiveness:

1. Design the UI to meet human deadline requirements
- Systems can feel responsive by providing feedback.
- Poor responsiveness: ignore user input, no clues on duration.

Deadlines
- preconscious perception: 0.01 second
- cause and effect: 0.1 second
- visual-motor reaction time: 1 second

Tricks
- Use Progress vs. Busy indicators
- Render / display important information first
- Fake heavyweight computations during tasks
- Working ahead (behind-the-scenes)

2. Load data efficiently so that it's available quickly
- Run in UI thread vs. Run in separate thread

Copy/Paste

Cut and paste via the clipboard and drag and drop allows for (relatively) easy data transfer within and between applications

Data formats

- Text vs. graphics (challenges of graphics?)

Saving data on the clipboard

- Lazy writing
- How long do you persist it?
- What formats do you save?

Benefits

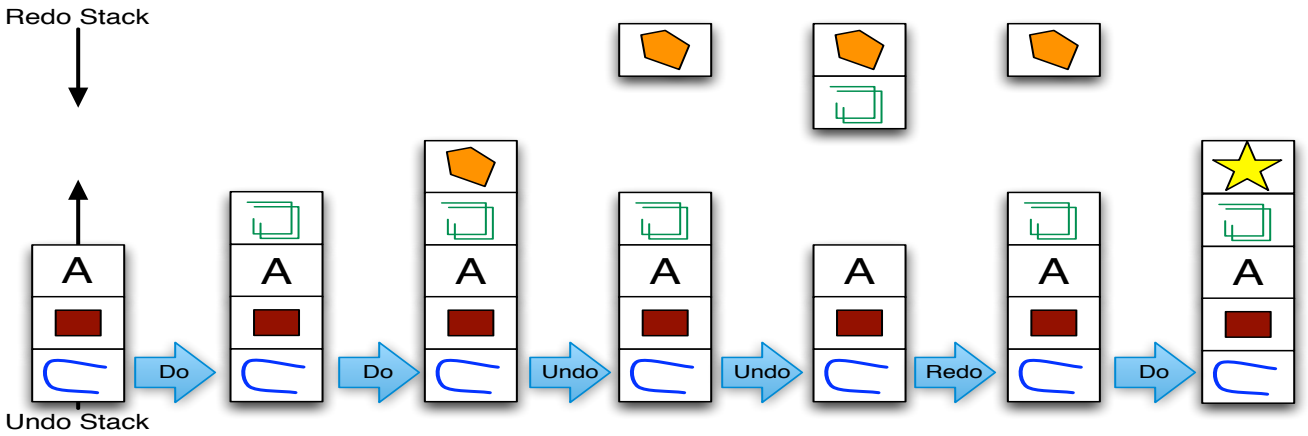
- Enables exploratory learning
- Lets you recover from errors and evaluate modifications

Design Decisions

- What is the user interface state after an undo or redo?
- What defines one undoable “chunk”?
- Is undo/redo global, local, or someplace in between?

Implementation:

- Forward vs. Reverse undo (apply CR)
- Change Record (CR) : Memento vs. Command pattern



Input

Given a computer input method, describe its sensing method and characteristics (e.g., continuous versus discrete, degrees of freedom).

What are the tradeoffs between using specific vs general input devices?

What considerations went into the design of the QWERTY keyboard? Why do we currently have so many variations of keyboards?

What is a positional input device? Given a positional input device, determine its characteristics (e.g., force versus displace sensing, position versus rate control, absolute versus relative positioning, direct versus indirect contact, sensing dimensions).

What is control-display gain (CD gain)? Why might we adjust it?

Apply the Keystroke Level Model (KLM) to a particular user interface task (You don't need to memorize the time for K, P, B, H, and M; these will be provided).

Compare KLM and Fitts' Law. What are their advantages and disadvantages?

Apply Fitts' Law to a particular target acquisition problem.

According to Fitts' Law, what is better?

- menu at the top of the screen versus application window
- context versus pie menus
- OS X dock expansion versus slowing mouse cursor near target

What is the steering law, and how is it relevant to input performance in applications?

Touch Interfaces (1)

An interaction model is a set of principles, rules, and properties that guide the design of an interface.

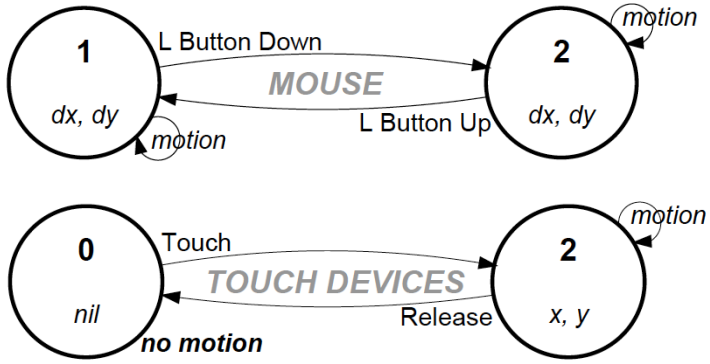
Instrumental Interaction: An interaction model based on how we use tools to manipulate objects in the physical world.

- Domain objects (the thing of interest, e.g., data/attributes) vs. Interaction instrument (a mediator between the user and domain objects).
- WIMP instruments activated spatially or temporally, with associated costs (space or time to activate).

Evaluating instruments

- Degree of indirection: Spatial/temporal offset between instrument/action
- Degree of integration: Ratio of DOF of instrument to DOF of input device
- Degree of compatibility: Similarity of action on control device to action on object

Touch Interfaces (2)



Displays

- Resistive, Capacitive

Input

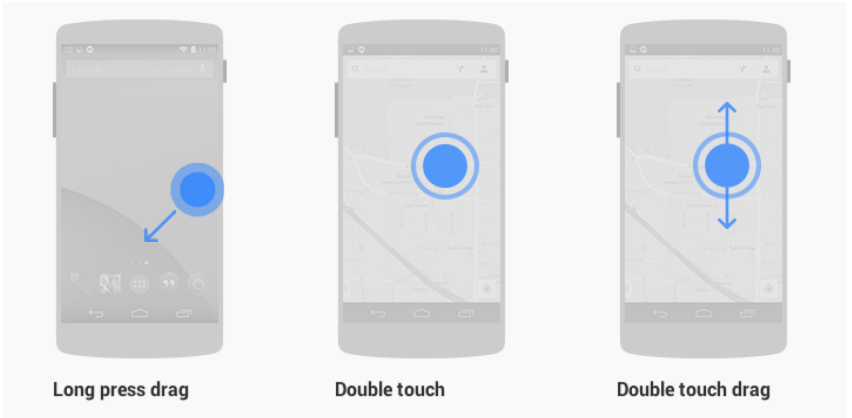
- Direct vs. indirect input devices
- Challenges of touch interfaces
  - fat finger problem: occlusion and imprecision
  - ambiguous feedback: lack of haptic feedback
  - lack of hover state: no preview before committing action
  - multi-touch capture: multiple, contradictory touches

Interaction

- WIMP: “Windows, Icons, Menus and Pointing”
- Direct manipulation: attempting to make **affordances** in the interface like affordances for **analogous** actions in the real world; discover through *exploration*; interfaces are *learnable*.
  - Direct touch breaks when movement constraints reached

Mobile Interaction

- How should gestures map to various system functions?
- Should gestures map to the most common tasks or the most complex?



DM on tabletops

- Fat “body part” problem
- Content Orientation
- Multiple, multi-touch input

Mobile Design

- Small displays: accommodate variety of screen sizes
- Navigation: single foreground app

Challenges of sensor data

- Computational cost: CV, aggregating data, noisy data
- Traditional or non-traditional interfaces: How do we integrate sensors into common, real-world applications?
- Managing errors (false positives and false negatives)

Gestural systems

- Touchless Interface is a one-state input device.
- “Live-mic problem” and “reserved actions”

Speech recognition

- Errors: rejection, substitution, insertion errors

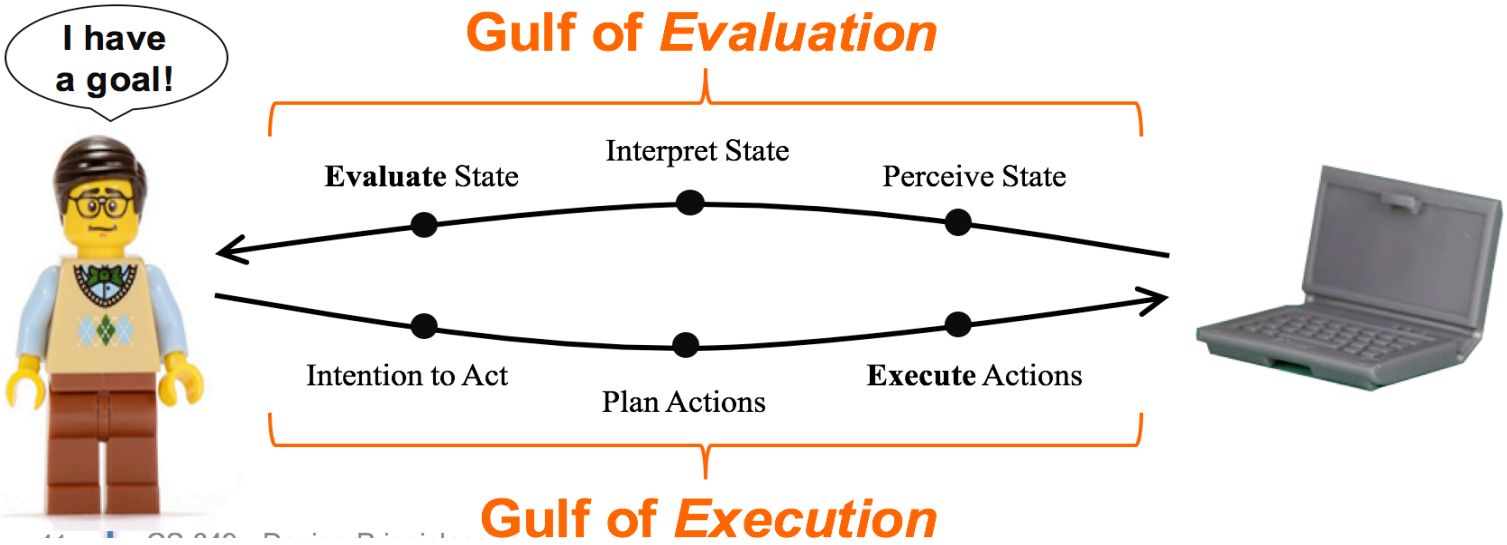
- Design
  - Small form factor
  - Device orientation handling
  - Hybrid interaction: touch, gestures, voice, keyboard, pointing
  - Constraints: limited processing, battery
  - Design tips for small devices?
- Event handling
  - OS managing activities
  - Listener model to handle touch events
- Android
  - *Not on the exam*

- Core technologies
  - Role of HTML, CSS, JS
  - *Syntax not on the exam*
- Event handling
  - JS event propagation model (capture/bubble)
  - How does it differ from listener model?
  - Need for asynchronous calls & responsiveness

Tension between Usefulness and Usability

Models of interaction

- System model, User model, (Developer model)
- Gulf of execution, Gulf of evaluation



Design Principles

- Perceived affordance: what people think you can do (e.g. pull handle)
- Mapping: relationship between input/output (e.g. turn wheel right)
- Constraints: guide certain actions, prevent others (e.g. disable control)
- Visibility/Feedback: make relevant parts visible (e.g. feedback)
- Metaphor: borrow concepts from familiar domain (e.g. desktop)

Simplicity: leads to quickly recognized/understood functionality.

Gestalt Principles: Grouping

- Proximity: Individual elements are associated more strongly with nearby elements than with those further away.
- Similarity: Elements associated more strongly when they share basic visual characteristics (shape, size, color).
- Good Continuation: Elements arranged in a straight line or a smooth curve are perceived as being more related

Gestalt Principles: Ambiguity, Missing Data

- Closure: We tend to see a complete figure even when part of the information is missing
- Figure/Ground: Our mind separates the visual field into the figure and the ground.
- Law of Pragnanz: We tend to interpret ambiguous images as simple and complete.
- Uniform Connectedness: Elements connected to one another by uniform visual properties
- Alignment: Should be included as a common Gestalt principle!

Pleasing Layouts: Contrast, Repetition, Proximity, Alignment.

Accessibility

People vary in their physical and mental capabilities

- The “average person” is just a statistical ideal

Temporary/situational disabilities

- Sickness, Underwater diving, Walking while using a smartphone

Aging

- Reduced motor coordination (fine/gross motor skills), Visual and hearing impairments, Loss of memory

Inclusive technologies to address impairments. What technologies can we design to address these issues?

- Visual
- Hearing
- Motor
- Cognitive

What is the “Curb Cut” phenomenon? Name a few examples.

Final Words

- KISS
  - Keep it simple and succinct
  - Our goal is not to confuse. Our goal is not to require long answers that we need to spend extra time deciphering.
- Neatness is your friend
  - Messy answers can cost you if TAs cannot decipher your answers. If you make a mess, use circle and callouts to clearly indicate what you want graded
- Read the questions and answer accurately
  - Give two examples means we grade *the first two* or *the two highlighted* examples. Others are ignored …
- Don’t write on the back of the exam!