CS160 Discussion Section

Matthew Kam

May 12, 2003

Administrivia

- Final project presentations due today (May 12, 2003)
- Posters due Wednesday (May 14, 2003)
- Final review session
  - May 15, 2003 (Thursday)
  - 3 to 5 PM
  - Location TBA
  - In view of limited time, email conceptual questions to newsgroup or mattkam@cs

Looking Ahead

- "Why is HCI part of the engineering curriculum?"

Looking Ahead

- "Why is HCI part of the engineering curriculum?"
- My answer: "Because the ultimate goal of any feat of engineering is to improve the human condition, and to do so requires that engineers understand and design for the intended users of technology."

Collaboratories

Telemedicine

Distance Education

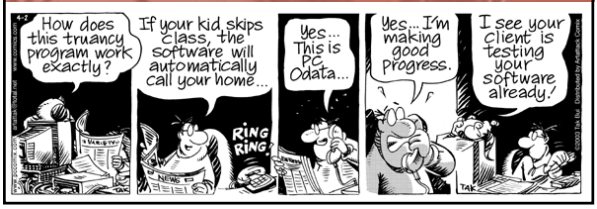
Precision Agriculture

Life After CS160

- Graduating?
  - Check out *HCI Panel scribe notes* for workplace-related information

Life After CS160

- Not yet graduating?
  - Ubiquitous Computing
    - Fall 2003 CS294-2
    - Professor Anind K. Dey



Life After CS160

- Not yet graduating?
  - Human-Centered Computing
    - Fall 2003 CS294-8 (graduate class)
    - Fall 2003 CS294-31 (seminar)
    - Professor John F. Canny

Life After CS160

- Not yet graduating?
  - Experience hands-on HCI research
  - http://guir.cs.berkeley.edu

GUIR
Group for User Interface Research
University of California at Berkeley

CS 160: Lecture 10

Professor John Canny
Spring 2003
March 3

Stroke Model

- Describe image as strokes (w/ color/thickness)
  - `Line ((10, 4), (17,4), thick 2, red)`
  - `Circle (( 19, 13), radius 3, thick 3, white)`
- Maps to early vector displays & plotters
- Most UI toolkits have stroked objects
  - arcs, ellipses, rounded rectangles, etc.

Problems with Stroke Model?

- How would you represent with strokes?
- Solution?

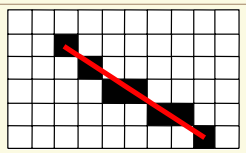
Pixel Model

- Break-up complex images into discrete "pixels" & store color for each
- Resolution
  - spatial: number of rows by columns
    - e.g., 1280 x 1024 is a good monitor display
    - quality laser printer: 6000 x 4800 (600 dpi)
  - image depth (i.e., number of bits per pixel)
    - several styles... 8-bit, 24-bit, 32-bit

Image Depth

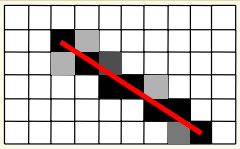
- Bit map - 1 bit/pixel (on/off)
  - B&W screens or print-outs

Aliasing



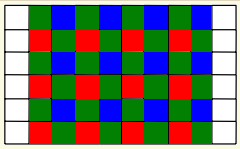
- Smooth objects (e.g., lines) appear jagged since resolution is too low
- Antialiasing - fill-in some jagged places w/ gray scale or primary colors

Anti-Aliasing



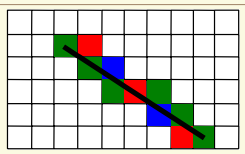
- Pixels colored in proportion to relative amount of line that crosses them.
- Equivalently, draw the line in B/W at finer resolution and then color each pixel in proportion to number of colored sub-pixels.

Cleartype



- The pixel matrix for a laptop or LCD screen.

Cleartype

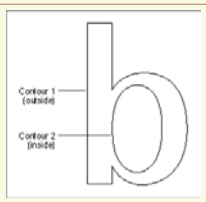


- Use sub-pixel color pixels as though they were gray pixels (can cause color anomalies).

Region Model

- Use the stroke model to outline region
- Fill the region with
  - colors & blendings (i.e., patterns)
- Advantages??
  - allows representation of filled shapes w/
    - little memory
    - independence from display resolution
- Text represented this way & converted to bitmaps inside of the printer

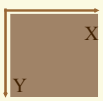
Outline Fonts



- Used by both Postscript & TrueType

Coordinate Systems

- Device coordinates
  - coordinates of the display device
  - origin usually at upper left
- Window coordinates
  - toolkit presents window as an abstraction
  - virtual display with frame around it (title, etc.)
  - program's coordinates placed in window as if frame doesn't exist
  - like device coords, always expressed in pixels
  - mouse events may be in device coords - check



Coordinate Systems (cont.)

- Physical coordinates
  - pixel-based coords don't deal well w/ devices of different resolutions (e.g., monitor vs. printer)
  - specify coordinates in physical units (e.g., inches, centimeters, or printer points)
- Model coordinates
  - coordinate system relative to drawn objects
  - need to convert from model to physical/window coordinates & back

Event-Driven Programming

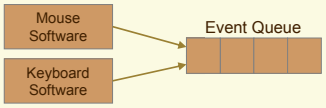
- Instead of the user waiting on program, have the program wait on the user
- All communication from user to computer is done via "events"
- An *event* is something "interesting" that happens in the system
  - mouse button goes down
  - item is being dragged
  - keyboard button was hit

CS 160: Lecture 11

Professor John Canny
Spring 2003
March 10

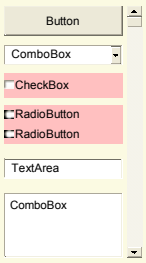
Event-Driven Programming

- All generated events go to a single *event queue*
  - provided by operating system
  - ensures that events are handled in the order they occurred
  - hides specifics of input from apps



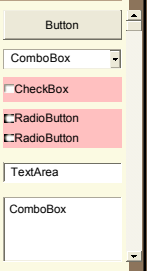
Widgets

- Reusable interactive objects
- Handle certain events
  - widgets say what events they are interested in
  - event queue/interactor tree sends events to the "right" widget
- Update appearance
  - e.g. button up / button down



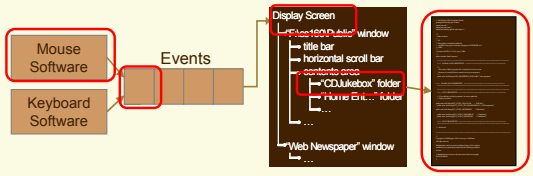
Widgets (cont.)

- Generate some new events
  - "button pressed"
  - "window closing"
  - "text changed"
- But these events are sent to interested listeners instead
  - custom code goes there



Main Event Loop

```
while (app is running) {
        get next event
        send event to right widget
}
```

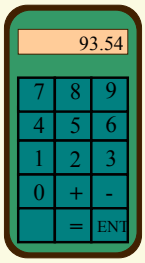


Interactor Tree

- Decompose interactive objects into a tree
  - interactive objects also known as "widgets"
  - based on screen geometry of objects
  - nested rectangles
- Used for dispatching events
  - events are dispatched (sent) to code in widget
  - the code then handles the event
- Variety of methods for dispatching events
  - return to this later

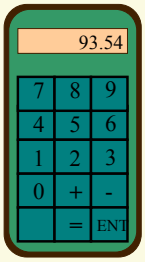
Who gets the events?

- To catch events, a widget registers a "listener" for that event
  - Mouse click
  - typed input
  - drag…
- Events go to a widget containing the pointer

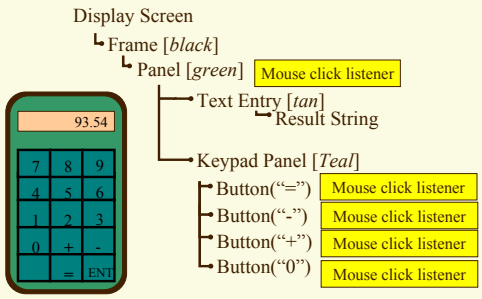


Who gets the events?

- But there are often several widgets containing the pointer
- Events go down the "stack" of visible widgets at the pointer until there is a widget that has registered a listener for that event



Interactor Tree (Java)



CS 160: Lecture 12

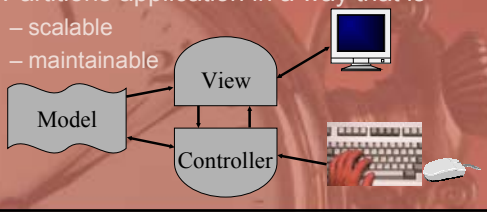
Professor John Canny

Spring 2003

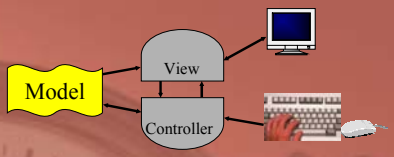
March 12

Model-View-Controller

- Architecture for interactive apps
  - introduced by Smalltalk developers at PARC
- Partitions application in a way that is
  - scalable
  - maintainable



Model



- Information the app is trying to manipulate
- Representation of real world objects
  - circuit for a CAD program
    - logic gates and wires connecting them
  - shapes in a drawing program
    - geometry and color

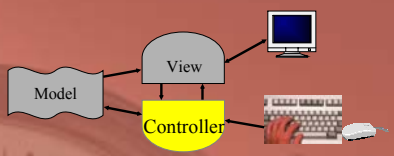
Example Code (Model)

```
class Model {
  class Shape {
    ...
    int type; // 0 - square, 1 - circle
    int x, y;
    Color color;
    Shape(int type, int x, int y, Color c) {…};
  }

  Shape shapes[MAX_SHAPES];  // array of shapes
  View views[MAX_VIEWS];  // array of Views

  void addCircle(Shape circle) {
    shapes.addElement(circle);
    for each view do
      view.refresh();
  }
}
```

Controller



- Receives all input events from the user
- Decides what they mean and what to do
  - communicates with view to determine which objects are being manipulated (e.g., selection)
  - calls model methods to make changes on objects
    - model makes change and notifies views to update

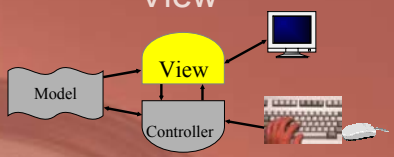
Example Code (Controller)

```
// declaration in corresponding Model
class Shape {
  ...
  int type; // 0 - square, 1 - circle
  int x, y;
  Color color;
  Shape(int type, int x, int y, Color c);
}

// code in corresponding Model
void addCircle(Shape circle) {
  ...
}

// Controller code
void onMouseClick(MouseEvent e) {
  addCircle(
    new Shape(Shape.CIRCLE, e.getX(),
              e.getY(), Color.BLUE));
}
```

View



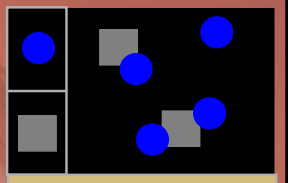
- Implements a visual display of the model
- May have multiple views
  - e.g., shape view and numerical view
- Any time the model is changed, each view must be notified so that it can change *later*
  - e.g., adding a new shape

Example Code (View)

```
// code in corresponding model
void addCircle(Shape circle) {
    shapes.addElement(circle);
    for each view do
      view.refresh();
}

// for graphical View
void refresh() {
   for each shape do
     shape.repaint();
}

// for text View
void refresh() {
  print("Blue circles:" + shapes.count(Shape.CIRCLE);
  print("Cardinal squares:" + shapes.count(Shape.SQUARE);
}
```



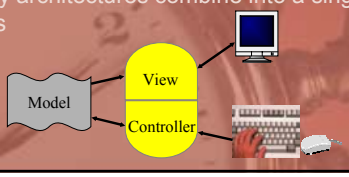
Blue circles: 4
Cardinal squares: 2

Why MVC?

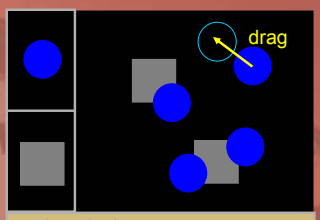
- Combining MVC into one class or using global variables will not scale
  - model may have more than one view
    - each is different and needs update when model changes
- Separation eases maintenance
  - easy to add a new view later
  - new model info may be needed, but old views still work
  - can change a view later, e.g., draw shapes in 3-d (recall, view handles selection)

Combining View & Controller

- View and controller are tightly intertwined
  - lots of communication between the two
- Almost always occur in pairs
  - i.e., for each view, need a separate controller
- Many architectures combine into a single class



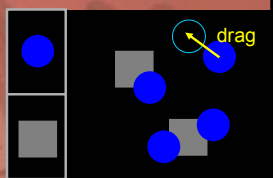
Combining V-C for Efficiency



Blue circles: 4
Cardinal squares: 2

Example Code

```
void onMouseDragged(MouseEvent e) {
   if (view1.inDraggedMode() {
     view1.getDraggedObject.setX(e.getX());
     view1.getDraggedObject.setY(e.getY());
     view1.refresh();
   }
}
```



Blue circles: 4
Cardinal squares: 2

CS 160: Lecture 13

Professor John Canny
Spring 2003
March 17

Principles

- Simplicity
- Scale, Contrast, Proportion
- Organization and Visual Structure
- Grid-based Design

Simplicity

- Simple designs are usually the most effective
- "Form ever follows function"
  - Sullivan

Simplicity - Unity

- One path to simplicity is through unifying themes:
  - Forms, colors, components with like qualities

Simplicity - Refinement

Simplicity - Fitness

- Match the design to the capabilities of the technology and the user

Why not use Roman fonts ?

Sans-serif fonts fit the medium

Be careful of slant

Module and Program

- A systematic approach to the design of many artifacts:
  - web pages on a site
  - documentation pages
  - devices in a family
- Programs describe how to build designs from modules.

Grid-based Design

Principles - Focus

- Focus: the design should highlight one or a small number of modular elements

Principles - Flexibility

- Flexibility: The program should allow variation from a theme

Univers Font

Principles - Consistency

- Consistent application: The program should maximize consistency in size, position, texture...

Techniques

- Reinforcing structure through repetition: Repeat design elements across the program
- Stylesheets can help

Techniques

- Establish modular units

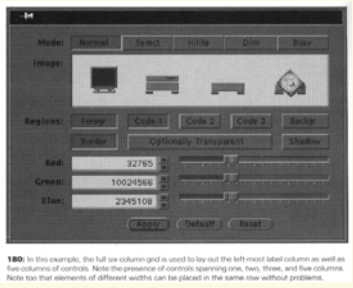
Techniques - Canonical grid

- The canonical grid (see notes)
- An six-column basic grid with column separators and label templates
- Can be implemented with HTML tables

Canonical Grid

- Determine any size restrictions
- Define horizontal and vertical modules
- Develop a rough sketch of sizes, positions, orientations
- Use the canonical grid to adjust sizes and positions of control elements
- For dynamic layouts, figure out the minimum workable size.

Canonical Grid



180: In this example, the full six column grid is used to lay out the left-most label column as well as five columns of controls. Note the presence of controls spanning one, two, three, and five columns. Note too that elements of different widths can be placed in the same row without problems.

CS 160: Lecture 14

Professor John Canny
Spring 2003

Motivation for Design Patterns

- Most examples from UI literature are critiques
  - Norman, Nielsen, etc.
- Design is about finding solutions
- Unfortunately, designers often reinvent
  - hard to know how things were done before
  - hard to reuse specific solutions
- Design patterns are a solution
  - reuse existing knowledge of what works well

Design Patterns

- First used in architecture [Alexander]
- Communicate design problems & solutions
  - how big doors should be & where…
  - how to create a beer garden where people socialize…
  - how to use handles (remember Norman)…
- Not too general & not too specific
  - use solution "a million times over, without ever doing it the same way twice"

Design Patterns

- Next used in software engineering [Gamma, et. al.]
  - communicate design problems & solutions
    - Proxy
      - surrogate for another object to control access to it
    - Observer
      - when one object changes state, its dependents are notified

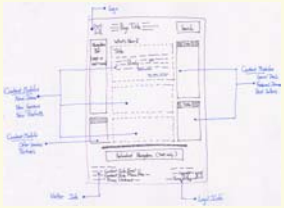
Design Patterns

- We can do the same for Web Design
  - communicate design problems & solutions
    - how can on-line shoppers keep track of purchases?
      - use the idea of shopping in physical stores with *carts*
    - how do we communicate new links to customers?
      - blue underlined links are the *standard* -> use them
- *Leverage* people's usage habits on/off-line
  - if Yahoo does things a way that works well, use it

Pattern Format

1. Pattern Title
2. Background Information
3. Problem Statement
4. Forces
   - (trade-offs, goals+constraints, motivating factors/concerns, tells why the problem is difficult)
5. Solution
6. Solution Sketch
7. Other Patterns to Consider

Home Page Design Rules



- Strong 1st impressions
  - compelling titles & logos
  - simple navigation
- Create a site that will be easy to update

Home Page Design Rules

- Problem
  - without a compelling home page (H/P), no one will ever go on to the rest of your site
  - surveys show millions of visitors leave after H/P
    - most will never come back -> lost sales, etc.

Six Ways to Make a Good Home Page

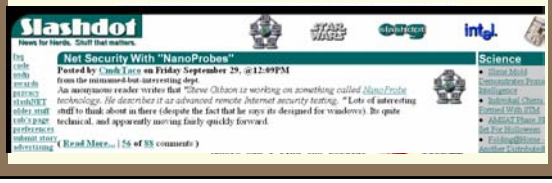
- Make a positive first impression by
  - testing
    - appropriate LINK NAMES & FAMILIAR LANGUAGE?
  - looking at GUEST PROFILES (another pattern)
    - appropriate colors & graphics?
      - neon green & screaming graphics on a skateboarding site, but not on a business-to-business or health site

Six Ways to Make a Good Home Page

- Focus on a single item of interest
  - create a good "first read"
    - draw the eye to a single graphical item
  - make it clean & larger than rest on the page
  - cut down remaining elements to chosen few

Six Ways to Make a Good Home Page

- Build your site brand
  - present the message of what your company does
  - include the value proposition (promise to visitors)
    - links to CONFIDENTIALITY INFORMATION & SITE ABUSE POLICIES to show you are trustworthy



Six Ways to Make a Good Home Page

- Make navigation easy to use
  - novices & experts must instantly "get it"
  - use MULTIPLE WAYS TO NAVIGATE
    - basic features of site as EMBEDDED LINKS
    - NAVIGATIONS BARS (there are several types)
    - HTML table COLORED BACKGROUNDS to delineate sections
    - REUSABLE ACCENT GRAPHICS to highlight new things

Six Ways to Make a Good Home Page

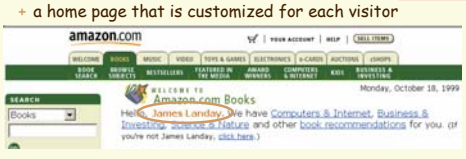
- Lure visitors to return
  - with fresh content
    - keep it updated so there is a reason to come back
  - by seducing with text
    - you have only seconds
      - ~ lively, sparkling, precise

Six Ways to Make a Good Home Page

- ⌘ Make it download quickly (2-3 seconds)
  - if not, they'll go elsewhere
- ⌘ Strategies
  - use HTML text as much as possible
    - first thing to download
    - images take 10 server-browser comms
    - get a web-savvy graphic artist (font colors, styles, & b/g color)
  - use small graphics
  - use min. number of columns & sections in a grid layout
    - easy to scan

Personalizing Your Home Page

- Problem
  - web sites would like to have visitors return often
    - to buy, see ads, use services, etc.
  - if your content isn't changing & personal, users less likely to return or stay
- Solution
  - personalization
    - a home page that is customized for each visitor



Editing Personalization Info

my.yahoo.com is a good example of editing for personalization

- Visitors click on buttons to make selections from lists
  - weather cities
  - news sources
  - stocks to follow
  - sports to follow
  - …
- Include content modules based directly on selections
- Drawbacks to this approach?
  - can get tedious if you have to do it repeatedly
  - users won't spend time entering info if little benefit

Interviewing for Personalization Info

- Visitors answer multiple choice questions
- Update GUEST PROFILE
- Include CONTENT MODULES based on one or more scoring methods
- Allow the option of continuing the personalization process over time

Deduction for Personalization Info

- Watch visitors behavior
  - Amazon tracks the books visitors order & later offers similar books
- Update GUEST PROFILE
- Select content modules based on scoring method

Collaborative Filtering for Personalization Info

- First provide popular content based on all visitors
- Provide customized content modules based on similar guest profiles
  - use correlation of profiles to determine areas of interest

Scoring Methods to Match Content to Audience

- Rank
  - show ordered list of content
- Top rank
  - content of only the top few scores shown
- Threshold score
  - show all content over a particular score
- Required attributes
  - show all content that is on "NCAA Sports"
- Combination
  - e.g., job site might use top rank & required attributes to show best jobs a person is qualified for

CS 160: Lecture 15

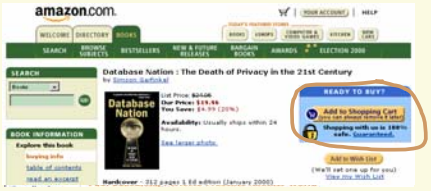
Professor John Canny
Spring 2003

Shopping Cart

- Problem?
  - how to allow customers to purchase multiple items in ***one transaction***
- Solution
  - use shopping cart metaphor to keep track of items before customer finalizes the purchase
    - track name, quantity, availability, & price

How to Apply Shopping Carts

- Make available on each page and easy to add items from product pages
  - make it easy for people to buy!
  - seems obvious, but many sites don't do it

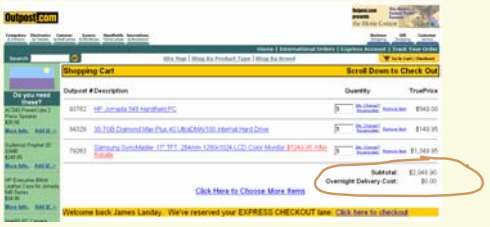


How to Apply Shopping Carts

- Provide detailed info on each item in cart

How to Apply Shopping Carts

- Provide info about all items in cart
  - sub-totals
  - shipping, taxes, other charges (if known)



How to Apply Shopping Carts

- Provide a prominent link to CHECKOUT

How to Apply Shopping Carts

- Have a link to let people continue shopping

How to Apply Shopping Carts

- Don't let unavailable things be added
  - hard to find a good example of this

Checkout

- Shopping Cart =>
  - Details, quantity, availability, subtotal
- Sign-in =>
  - New customers
  - Returning customers
- Shipping =>
  - Address, shipping method, gift wrap?, special instructions
- Payment =>
  - Method, billing address, gift certificate, coupons
- Confirmation
  - Confirm button, confirmation page, email, order tracking into, Thank you

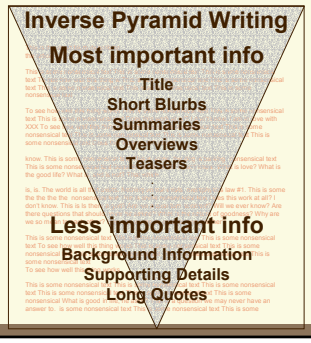
Checkout

- Make it easy to cancel or change order at any time before final confirmation
- Don't have customers type things twice

INVERSE PYRAMID WRITING STYLE

- Problem
  - 25% slower to read from a computer screen
  - web surfers want immediate gratification or they will leave – they want web pages that are
    - delivered quickly
    - easy to use
    - quick to skim
- Solution (?)
  - give the conclusions first
  - add detail later for those that want it

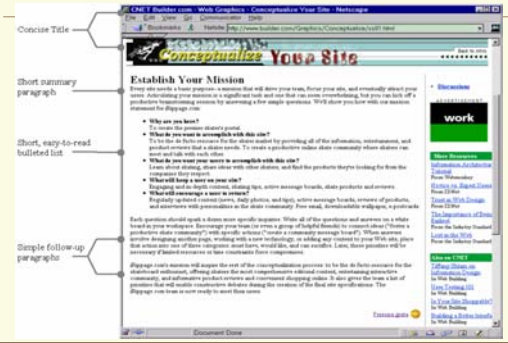
INVERSE PYRAMID WRITING STYLE (D7)



ZDNet Uses Inverted Pyramid

- Start with a good concise title
  - reflect the content
- Continue with most important points
  - use hypertext to split-up information
  - people often won't scroll or read
- Use less text
  - 50% less than you would offline
- Use a simple writing style
  - simple sentences -- no hype or advertising
- Use EMBEDDED LINKS to help visitors find more information
- Use bullets and numbered lists
  - supports skimming

Using Bullets



Empirical Results (cont.)

- None were very good
- Bad ones
  - home pages offered little direction on content
- "Readable" pages were less effective
  - people don't read, they skim
  - nicely formed sentences hide key information

Empirical Results (cont.)

- Download time wasn't a big issue
  - no correlation between time and frustration
- Graphic design had very little effect
  - take a look at Yahoo
  - may be important for brand, marketing, etc.
- Navigation must be linked to content
  - if not, users found sites less usable
  - forget about designing separately ("shell" sites)
    - if can remove ½ of content without changing home page, then it is a shell site
    - generic links do not give users predictable results

Empirical Results (cont.)

- Violating the "sales script"
  - standard human-human sales situations
    - browse and then give personal info when you buy
      - employees wear name tags
  - web-based situations that violate this fail
    - users driven away by giving personal info first
    - you must first build trust!

Animation

- Higher click-thru rates, but
  - annoyed users
    - scrolled, covered with hands...
    - animation makes it much harder to read/skim
- Could be useful in conveying information
  - they found no examples of this
- "Surfing" different from info. retrieval
  - may not be able to design a site for both

Empirical Results (cont.)

- Frames
  - not so bad, but
  - make sure large frame changes are obvious as a result of clicks in small (TOC) frame
  - Make sure most important content fits in frame

Links

- Users had trouble with short links
  - "If you click on Disneyland, will you get a map of the park? Ticket Information, etc?"
- Longer links clearly set expectations
  - "How to Read the Pricing and Rating Listings"
  - "Pricing (How to Read Pricing & Rating Listings)
- Links embedded in paragraphs are worse
  - hard to find information
  - can't skim - must now read
- Text links used before graphical links

Links (cont.)

- Within-page links
  - sometimes confusing if user scrolls & has read material already
  - make shorter pages to avoid
- Wrapped links caused confusion
  - tradeoff here...

CS 160: Lecture 16

Professor John Canny
Spring 2003

Qualitative vs. Quantitative Studies

- Qualitative: What we've been doing so far:
  - Contextual Inquiry: trying to understand user's tasks and their conceptual model.
  - Usability Studies: looking for critical incidents in a user interface.
- In general, we use qualitative methods to:
  - Understand what's going on, look for problems, or get a rough idea of the usability of an interface.

Qualitative vs. Quantitative Studies

- Quantitative:
  - Use to reliably measure something.
  - Requires us to know how to measure it.
- Examples:
  - Time to complete a task.
  - Average number of errors on a task.
  - Users' ratings of an interface *:
    - Ease of use, elegance, performance, robustness, speed,...

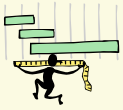
* - You could argue that users' perception of speed, error rates etc is more important than their actual values.

Quantitative Methods

- Very often, we want to *compare* values for two different designs (which is faster?). This requires some statistical methods.
- We begin by defining some quantities to measure - variables.

Variable types

- Independent Variables: the ones you control
  - Aspects of the interface design
  - Characteristics of the testers
  - Discrete: A, B or C
  - Continuous: Time between clicks for double-click
- Dependent variables: the ones you measure
  - Time to complete tasks
  - Number of errors



Deciding on Data to Collect

- Two types of data

  - process data
    - observations of what users are doing & thinking
  - bottom-line data
    - summary of what happened (time, errors, success…)
    - i.e., the dependent variables

Process Data vs. Bottom Line Data

- Focus on process data first
  - gives good overview of where problems are
- Bottom-line doesn't tell you where to fix
  - just says: "too slow", "too many errors", etc.
- Hard to get reliable bottom-line results
  - need many users for statistical significance

Significance

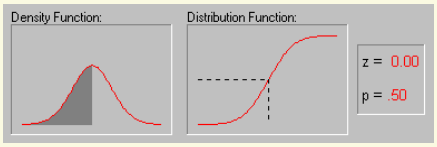
- The significance or p-value of an outcome is the probability that it happens by chance if the relation does ***not*** hold.
- E.g. p = 0.05 means that there is a 1/20 chance that the observation happens if the hypothesis is false.
- So the smaller the p-value, the greater the significance.

Significance

- For instance p = 0.001 means there is a 1/1000 chance that the observation would happen if the hypothesis is false.
  So the hypothesis is almost surely true.
- Significance increases with number of trials.
- CAVEAT: You have to make assumptions about the probability distributions to get good p-values.

Normal distributions

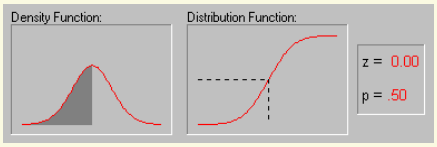
- Many variables have a Normal distribution



- At left is the density, right is the cumulative prob.
- Normal distributions are completely characterized by their ***mean*** and ***variance*** (mean squared deviation from the mean).

Normal distributions

- The difference between two independent normal variables is also a normal variable, whose variance is the sum of the variances of the distributions.



- Asserting that X > Y is the same as (X-Y) > 0, whose probability we can read off from the curve.

Lies, damn lies and statistics…

- A common mistake (made by famous HCI researchers *)
  - Increasing n (number of trials) by running each subject several times.
  - No! the analysis only works when trials are *independent*.
  - All the trials for one subject are dependent, because that subject may be faster/slower/less error-prone than others.

* - making this error will not help you become a famous HCI researcher ☺.

Statistics with care:

- What you *can* do to get better significance:
  - Run each subject several times, compute the *average* for each subject.
  - Run the analysis as usual on subjects' average times, with n = number of subjects.
- This decreases the per-subject variance, while keeping data independent.

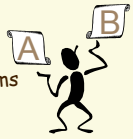
Measuring User Preference

- How much users like or dislike the system

  - can ask them to rate on a scale of 1 to 10
  - or have them choose among statements
    - "best UI I've ever…", "better than average"…
  - hard to be sure what data will mean
    - novelty of UI, feelings, not realistic setting, etc.
- If many give you low ratings -> trouble
- Can get some useful data by asking
  - what they liked, disliked, where they had trouble, best part, worst part, etc. (redundant questions)

Using Subjects

- ***Between subjects*** experiment
  - Two groups of test users
  - Each group uses only 1 of the systems



- ***Within subjects*** experiment
  - One group of test users
  - Each person uses both systems

Between subjects

- Two groups of testers, each use 1 system
- **Advantages:**
  - Users only have to use one system (practical).
  - No learning effects.
- **Disadvantages:**
  - Per-user performance differences confounded with system differences:
  - Much harder to get significant results (many more subjects needed).
  - Harder to even predict how many subjects will be needed (depends on subjects).

Within subjects

- One group of testers who use both systems
- **Advantages:**
  - Much more significance for a given number of test subjects.
- **Disadvantages:**
  - Users have to use both systems (two sessions).
  - Order and learning effects (can be minimized by experiment design).

CS 160: Lecture 18

Professor John Canny
Spring 2003

Mere presence

- Stress, anxiety or stimulation increase physiological arousal, and arousal speeds up behavior.
- The presence of others pushes these buttons…
- But increased speed can also increase errors, so it can be bad on difficult tasks.

Mere presence

- Increased arousal generally helps learning
- But, it also heightens response to well-learned stimulae (Zajonic and Sales):

It says "alpha helix"

Mere presence

- Mere presence isn't quite the right idea.
- The presence of a blindfolded subject didn't increase arousal, and didn't affect performance.
- The presence of others evaluating or competing with us is what matters.

Mere presence – Design Implications

- Increasing the level of group "awareness" should increase mere presence effects:
  - Heightened arousal
  - Faster performance
  - Increased learning
  - More errors
- Examples:
  - High awareness – video conferencing, phone
  - Medium – Instant messaging
  - Low awareness – Email

Attribution

- How do we attach meaning to other's behavior, or our own? This is called attribution.
- E.g. is someone angry because something bad happened, or because they are hot-tempered?

Attribution theory

- Attribution theory concerns itself with cause: was this behavior caused by personality, or environment?
- Actor-Observer effect:
  - When I explain my own behavior, I rely on external explanations, "monsters took my shoes"
  - When I explain others' behavior, I'm more likely to attribute it to personality and disposition, "bad kid"

Attribution theory – design implications

- In order to understand another's behavior, its important to have as much context as possible.
- E.g. room-scale video-conferencing:

Social Comparison

- We need to make comparisons to make judgements about people. Three rules:
  - Limitation: qualities must be observable and comparable to be attributed.
  - Organization: we use categories to describe and think about people; friendly, studious, careless etc.
  - Meaning: categories of personality must make sense, e.g. friendly and cooperative go together, friendly and hostile do not.

Groups

- Groups are a strong influence on our behavior.
- A "reference" group is one we share a psychological connection with, e.g. a club or honor society we aspire to join.
- We compare our selves to reference groups to make self-assessments.

Groups

- Groups give us value in several ways:
- They provide us norms for behavior (informational function)
- They satisfy interpersonal needs (interpersonal function)
- They provide us with concrete support, resources, help (material function)

Groups and Motivation

- Groups increase motivation in two ways
- First, the social interaction with the group intensifies individual motivation, and sometimes generates new individual motives.
- Second, the group can cause group goals and motives to be created. E.g. group maintenance is goal most groups have.

Group goals

- Goals can be either short-term or long-term.
- Long-term goals are harder to manage and maintain and generally have less effect on group behavior.
- Short-term goals are strong force in motivating and reinforcing group performance.

Group goals

- The composition of the group can strongly affect its goals.
- E.g. a group united by profession will tend to adopt goals related to the profession's methods.
- Groups often have subgroups that wield influence over the main group. They need not be majorities.

Group experiences

- Previous experience affects goal-setting.
- Groups that have succeeded are more likely to raise goals, groups that have failed are unlikely to lower them.

Group experiences – design implications

- Normative data can be very helpful – how am I doing compared to a typical colleague?
  - Compute normative data automatically
- Set short-term goals, mark off successes – challenge to do this efficiently
  - PERT charts or Calendars
  - Daily software builds
  - Extreme programming

Background: TVI and DTVI

- The TVI (Tutored Video Instruction) method was developed at Stanford.
  - A recording (videotape or web-based) is made of the lecture.
  - Students review the lecture in a small group (4-7 students) with a tutor.
  - The students pause the replay, and discuss with each other.
  - There is a lot of interaction: 50% of students participate in 50% or more of the discussions.
- DTVI is Distributed TVI. The lecture is webcast, and student interact with each other and the tutor using videoconferencing.

TVI/DTVI studies

- There have been many studies of TVI/DTVI.
- One of the largest was a study of DTVI with Sun Microsystems. The results were remarkable:
  - Students using DTVI received grades 0.2 to 0.8 std. deviations **higher** than students taking the same class live.
- Group interaction by itself is a facilitator of learning (independent of salience).

CS 160: Lecture 19

Professor John Canny
Spring 2003

CSCW: Computer-Supported Cooperative Work

- Its about tools that allow people to work together.
- Most of the tools support remote work
  - video, email, IM, Workflow
- Some tools, e.g. Livenotes, augment local communication.

Asynchronous Groupware

- Email: still a killer app
- Newsgroups: topical messaging
- Structured messaging: e.g. Workflow – messages carry data, route automatically.
- Cooperative hypertext/hypermedia authoring: e.g. swiki
- Knowledge repositories: Answergarden, MadSciNet, Autonomy

Synchronous Groupware

- Desktop Conferencing (MS Netmeeting)
- Electronic Meeting Rooms (Access Grid)
- Media Spaces (Xerox PARC)

Video

- Eye contact problems:
  - Offset from camera to screen
  - "Mona Lisa" effect

- Gesture has similar problems: trying pointing at something across a video link.

Sound

- Good for one-on-one communication

- Bad for meetings. Spatial localization is normally lost. Add to network delays and meeting regulation is very hard.

Breakdowns

- Misunderstandings, talking over each other, losing the thread of the meeting.
- People are good at recognizing these and recovering from them "repair".
- Mediated communication often makes it harder.
- E.g. email often escalates simple misunderstandings into flaming sessions.

Solutions

- Sharing experiences is very important for mutual understanding in team work (attribution theory).
- So context-based displays (portholes) work well.
- Video shows rooms and hallways, not just people or seats.

Solutions

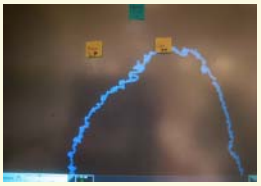
- Props (mobile presences) address many of these issues. They even support exploration.

Solutions

- Ishii's Clearboard: sketching + presence

Solutions – Outpost (Berkeley)

- Post-it capture system for web site design.
- For collaboration, add pen traces and user shadows (to add awareness).



Face-to-Face: the ultimate?

- It depends.
- Conveys the maximum amount of information, mere presence effects are strong. But…
- People spend a lot of cognitive effort managing perceptions of each other.
- In a simple comparison of F2F, phone and email, most subjects felt *most* comfortable with the *phone* for routine communication.

Face-to-Face: the ultimate?

- Kiesler and Sproull findings:
  - Participants talk more freely in email (than F2F).
  - Participation is more equal in email.
  - More proposals for action via email.
  - Reduced effects of status/physical appearance.
- But
  - Longer decision times in email.
  - More extreme remarks and flaming in email.

Face-to-Face: the ultimate?

- Kiesler and Sproull found that email-only programming teams were more productive than email+F2F teams in a CS course.
- There you want coordination, commitment, recording.
- Conclusion: Match the medium to the mission

Grudin: Eight challenges

1. Disparity between those who benefit from the App, and those who have to work on it.
   - e.g. secretary uses calendars to schedule meeting, but others must maintain calendars.
2. Critical mass, Prisoner's Dilemma
   - Need full buy-in to automate scheduling, similarly with Lotus Notes.

Grudin: Eight challenges

3. Disruption of social processes:
   - people are flexible, adaptive, opportunistic, **improvisors**, sometimes imprecise.

4. Exception Handling:
   - People react to interruptions or exceptions and dynamically re-plan what to do. Most software doesn't plan, so exception-handling must be anticipated and pre-programmed.

Grudin: Eight challenges

5. Unobtrusive accessibility:
   - Group features should complement individual work functions, and be easily accessible

6. Difficulty of evaluation:
   - Collaborators add uncertainty! Hard to isolate the parameters you want to study. WOZ can help.

Grudin: Eight challenges

7. Failure of intuition:
   * Group processes (and social psychology) are often counter-intuitive. This leads to mistakes both by adopters and designers.

8. The adoption process:
   * Very hard to get people to voluntarily change their habits. Incentives are often needed. Otherwise follows a (slow) adoption curve.

Beyond communication

- How can computers assist cooperative work beyond communication?
- Can they "understand" conversation?
- Speech-act based systems like the Coordinator attempted to do so.
- General understanding is too hard. But business communication is mostly about propose-accept-acknowledge sequences.

Coordinator

- The Coordinator was a system that tried to track these speech acts. Users had to specify the type of each utterance (email message).
- The experiment failed, it was too constraining.
- But it was reborn as Workflow.

CSCL: Computer-Supported Collaborative Learning

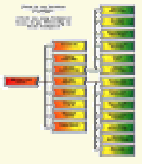
- Sub-area of CSCW concerned with learning and collaboration.
- Peer interaction is a powerful source of learning, especially in universities.
- Three powerful models:
  * TVI, DTVI: recorded instructor, team review
  * Peer instruction: pauses for group discussion
  * PBL: Problem-based learning, team problem-solving

CS 160: Lecture 20

Professor John Canny
Spring 2003

Information Design

- A confusing term, sometimes used as a catch-all for web design.
- We mean the organization of information (content) on a site. Hierarchy, links, navigation.



Genres of Web Sites

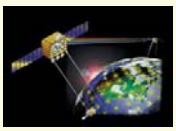
- A genre is a particular style of document, together with accepted practices of use:
  - Wedding invitation
  - Jury summons
  - Tax bill
- Web site genres:
  - Personal home page
  - Informational page
  - Portal page
  - E-commerce page

Information Tasks

1. Specific Fact-finding:
   - Find the library of Congress call number for future shock
   - Find the phone number of Bill Clinton
   - Find highest-resolution LANDSAT image of College Park at noon on 13 Dec 1997

Information Tasks

2. Extended Fact-finding:
   - What other books are by the author of Jurassic Park?
   - What kinds of music is Sony publishing?
   - Which satellites took images of the Persian Gulf War?

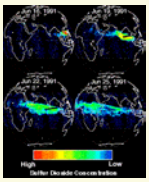


Information Tasks

3. Open-ended browsing:
   - Does the Matthew Brady Civil War photo collection show the role of women?
   - Is there new work on voice recognition in Japan?
   - Is there a relationship between carbon monoxide and desertification?

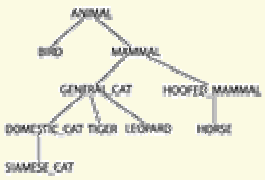
Information Tasks

4. Exploration of availability:
   - What genealogy information is at the National Archives?
   - What information is there on the Grateful Dead band members?
   - Can NASA data sets show acid rain damage to soy crops?



Objects/Actions Model

- Related to task analysis.
- Start with objects and actions.
- Organize both into taxonomies.



Taxonomies

- The object taxonomy is a traditional taxonomy.
  - E.g. music library -> music styles -> albums -> songs
- The action hierarchy is a TDH (Task Decomposition Hierarchy)
  - Find Alanis Morisette -> Use search -> enter terms

OAI model

- Task
  - Structured information objects (e.g. hierarchies and networks)
  - Information actions (e.g. searching, linking)
- Interface
  - Metaphors for information objects (e.g. bookshelf and encyclopedia)
  - Handles (affordances) for actions (e.g. querying and zooming)

Organizing information

- Define "atomic" information - the smallest chunks a user might want.
- Organize the atoms (if not automatic):
  - Short unstructured lists: highlights, what's new
  - Linear structures: Calendars, alphabetic lists
  - Arrays or tables: Time/place, Model/features
  - Hierarchies: Home/office -> product type -> model
  - Multi-trees: links that enable navigation in several trees at once
  - Networks: General hypertext

Action hierarchies

- Define atomic actions
  - Looking for a name in a list
  - Scanning a list for something interesting
  - Reading a paragraph
- Aggregate actions
  - Browsing TOC, jump to chapter, scan for topics
  - Locate term in index, start reading section with that term
  - Follow cross references from one doc to another, until no new docs.

Info Metaphors

- Mostly, we use hierarchies
  - File cabinet/folder
  - Book/chapter
  - Encyclopedia with articles
  - Television with channels
  - Shopping mall with stores
  - Museum with exhibits

Action Metaphors

- Various "next" buttons
- Slide show metaphor
- Zoom in/lens
- Up/down in a hierarchy

Info search: Four-phase pattern

1. Formulation
   * Pick the appropriate library or collection
   * Pick the style of search, normal/advanced

Four-phase pattern

2. Action
   * Click on search
   * Adjust parameters of previous search

Four-phase pattern

3. Review of results
   * URL+Document title, with context
   * Explanatory messages
   * Ordering method, alphabetical etc.
   * Apply clustering by topic

Four-phase pattern

4. Refinement
   * Offer help in redefining the query
   * Relevance feedback (good/bad docs)
   * Provide search page with previous values
   * Provide option to save search options if complex

CS 160: Lecture 21

Professor John Canny
Spring 2003

Human error recovery

- People make mistakes in communication all the time, and are adept at repairing them.
- Repair need not involve assignment of blame.
- E.g. "I'm sorry, I didn't understand. Can you rephrase?"
  * Tells the speaker you heard what they said, but were unable to interpret it. They should repeat, but express it differently.
  * There is no assertion that what they said was ungrammatical or incorrect, and you accept some blame by default.

Humane error recovery

- In human communication, an error is the *beginning of a process of repair*. It is "normal" to make errors.
- In human-machine interaction, errors normally lead to *the end of the current task*. Errors are treated as "abnormal".
- In other words, user interfaces usually try to escape from the repair process, leaving the user stranded.

Types of errors

- Mistakes
  - User intended to do what they did, and it led to an error. User would probably do the same thing again.
- Slips
  - User did not mean to do what they did. They can recover by doing it differently again.
  - Slips are not just for beginners. Experts often make them because they devote less conscious attention to the task.

Minimizing Error

- **System errors:** Program defensively (assert, bounds check (please!!))
- Estimated loss of productivity due to Windows OS crashes $170 B.
- Estimate for Windows XP $17 B.
- Note: almost all Windows XP vulnerabilities are standard buffer overruns.

Minimizing Error

- User errors:
  - Use Intuitive command names.
  - Include short explanations as "tool tips".
  - Put longer explanations in help system.

Minimizing Error

- Recognition over recall
  - Easier to select a file icon from a folder than to remember and type in the filename.
  - Auto-completion can help fix this.
- Use appropriate representations
  - E.g. graphical file selector good for choosing individual files
  - Textual file names support automation, richer organization (using command line options).

Typical Errors

- From Norman's 1983 survey:
- Mode errors:
  - User forgets what mode they're in, and does the command appropriate for another mode.
  - Digital watches, VCRs etc.
  - Common where there aren't enough command keys for all the operations
  - You can still explain the mode by giving the user feedback on what keys do in the display.

Typical Errors

- Description error:
  - The action is insufficiently specified by the user.
  - User may not know all the command line switches, or all the installation options for a program.
  - Solution: Warn the user that the command is ambiguous, or "unusual". Provide help about options in several standard ways.

Typical Errors

- Capture error:
  - Command sequences overlap, and one is more common.
  - User reflexively does the common one when trying to do the unusual one.
  - E.g. try typing "soliton" very fast.

System Response

- Stop the user from continuing the way they were going (possibly compounding the error).

- Take "safe" recovery actions - e.g. auto-save the state with a unique name.
- Begin the recovery process.

System Responses

- Gag
- Warn
- Do Nothing
- Self Correct
- Teach Me
- Let's talk about it

Gag Response

- Machine freezes, often not even accepting input.

- Generally a bad idea, but there are good uses:
  - Raskin's FLOW system, refuses to accept keystrokes that are not legal commands.
  - Intended for naïve users.
  - Even random typing produces legal programs!

Warn Response

- Machine accepts input, even if not legal. Warns user about problem(s).

- Allows user to get into deeper trouble.
- Allow backtracking "undo", back to start of trouble if possible.

Do Nothing Response

- Machine accepts input, even if not legal. Machine does nothing
- User has to figure out that something's wrong.

- Usually a bad idea, but sometimes useful in automation (don't copy file to itself etc.).

Self Correct

- DWIM (Do What I Mean) was an aggressive self-correcting system. Spell checkers are of this type.

- Generally good **but:**
  - Sometimes override user intent.
  - Can frustrate on frequently-used strings: "naïve" is good but not "Hsi".
  - Solutions:
    - Don't repeat correction if overridden.
    - Watch user behavior over time.

Let's talk about it

- Inspired by human error recovery.
  - Machine explains its understanding of the situation, gives the user some options.
  - Examples: network diagnostic wizard, modem wizard,…
- Very difficult to program these, but they're very valuable.
  - Need some analysis of previous user problems.
  - Look at help desk logs.

Teach Me

- System informs user what they can do.
- Common in speech recognizers. Include command "what can I say?", or run speech tutorial.

Explanations

- The first part of the recovery process is to explain what appears to be wrong.
- Remember the user is only supposed to have a functional model of what's going on. Try to give an explanation at high level.
- People understand basic resources like filespace, memory etc.
- "I'm afraid the program has an internal fault in the code that <<reads and writes files>>, which was called when trying to <<save your user preferences>>. We regret the inconvenience, and are trying to recover…"

Recovery

- The system should always give the user a reasonable amount of information about the problem.
- Better to err on the side of too much information.
- Some problems are amenable to automatic repair: retry, use redundant information, backup data etc…
- DWIM (Do What I mean) was an editor that attempted to correct common user errors. You need an "undo" key to use this approach.

CS 160: Lecture 22

Professor John Canny
Spring 2003

Types of Help

- Quick Reference:
  - Reminders of command names or options
- Task-specific help
  - User needs help on how to apply the command

Types of Help

- Full explanation
  - User wants complete understanding, e.g. for future use.
- Tutorial
  - The tutorial leads the user through a task, scaffolding their actions.

More advanced ideas

- Help is a kind of ongoing learning environment.
- Minimalist instruction (Carroll '92) is a learning approach
  - It shows users what to do,
  - then gives them realistic tasks to solve.
  - It eliminates conventional exercises, tedium and repetition, and encourages users to explore.
  - It also has extensive coverage of error recovery.
    - - users feel confident exploring.

More advanced ideas

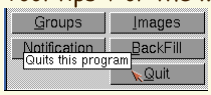
- Help could be enjoyable? - at least it's a special case of computer-supported learning..
- "Training wheels" (Carroll)
  - Advanced commands are removed until user gains some experience with the system.
  - Also some "dangerous" commands.
  - Users explore freely in this sandbox.
  - Users gained better understanding (IBM trial).

More advanced ideas

- The "scenario machine" uses this idea and adds more help:
  - Explanations of why certain commands are missing.
  - Alternative command sequences for missing commands.

Context-sensitive help

- Help depends on where it is used:
- Tool tips ↓ or the windows ? symbol:



- Save the user the burden of synchronizing program state with help system state.
- Almost always a good idea to do this.
- Just make sure the user can easily find the main help contents and index.

Online tutorials

- Can be useful, BUT:
  - Users are not the same, some need minimal help.
  - Forcing the user to execute a particular command is boring and annoying, and doesn't help learning.
- So..
  - Make sure users can skip steps.
  - Show users multiple ways of doing things.
  - Give partial information on what to do, with more information available if the user requests it.

On-line docs

- Online docs differ from paper manuals in the same way web sites differ from books.
- Information organization is the key -
  - Some pages contain content, others are primarily for navigation.
  - Use best practices for web site design:
  - Intuitive names, careful clustering of information, intuitive link names, consistency…
  - Need a good search feature and an index.

Adaptive Help Systems

- Adaption is a good idea because:
  - It avoids information that is too detailed or not detailed enough for a particular user.
  - It avoids repetition of info the user has already seen.
  - Can make suggestions of new ways to do tasks that the user may not know.
- Weaknesses:
  - Information can disappear (bad if the user forgot it too!).
  - System needs to know user identity and user must use the system for some time.

User Models

- Beware:
  - Linear scales (Novice - competent - expert), people don't advance on the same path.
  - Stereotypes - same as above, plus users may have different kinds of problems in using the system.

User Models

- Problematic:
  - Overlay model - ideal behavior that the user should follow (e.g. in tutorials). But doesn't allow the user to do things their own way or explore.
  - Task modeling: automatic task modeling is hard, and doesn't model bottom-up user behavior or "distributed cognition" (e.g. desk as a blackboard)

Knowledge representation

- Rule-based techniques
  - Limited success in AI, but scales well.
- Frame-based techniques
  - better organized version of RBT.
- Semantic nets…
- Decision trees…
- Other AI stuff…

Knowledge representation

- Probabilistic models provide a way to:
  - Allow several alternative interpretations at once.
  - Incorporate new evidence.
  - Model and explain the system's certainty in its own decisions.
- Used in MS' help system.

Initiative

- A good mixed-initiative help system requires links between all parts of the system including a tutorial.
- User should be able to "take over" at any time, then give back control.

Design issues

- Help system design is like other parts of the interface.
  - Start with task analysis.
  - Do paper prototypes.
  - Do user tests at informal and formal stages - look for errors.
  - Use errors as the "objects" to guide the design of the help system.

Design issues

- User modeling.
  - The error list can be used to derive user models.
  - Run pattern recognition on the error list to find the dimensions of the user profile.

CS 160: Lecture 23

Professor John Canny
Spring 2003

Multimodal Interfaces

- Multi-modal refers to interfaces that support non-GUI interaction.
- Speech and pen input are two common examples - and are complementary.

Speech+pen Interfaces

- Speech is the preferred medium for subject, verb, object expression.
- Writing or gesture provide locative information (pointing etc).

Speech+pen Interfaces

- Speech+pen for visual-spatial tasks (compared to speech only)
  - 10% faster.
  - 36% fewer task-critical errors.
  - Shorter and simpler linguistic constructions.
  - 90-100% user preference to interact this way.

Put-That-There

- User points at object, and says "put that" (grab), then points to destination and says "there" (drop).
  - Very good for deictic actions, (speak and point), but these are only 20% of actions. For the rest, need complex gestures.

Multimodal advantages

- Advantages for error recovery:
  - Users intuitively pick the mode that is less error-prone.
  - Language is often simplified.
  - Users intuitively switch modes after an error, so the same problem is not repeated.

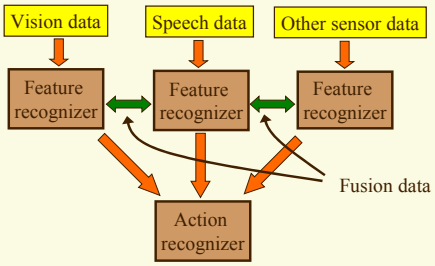
Multimodal advantages

- Other situations where mode choice helps:
  - Users with disability.
  - People with a strong accent or a cold.
  - People with RSI.
  - Young children or non-literate users.

Multimodal advantages

- For collaborative work, multimodal interfaces can communicate a lot more than text:
  - Speech contains prosodic information.
  - Gesture communicates emotion.
  - Writing has several expressive dimensions.

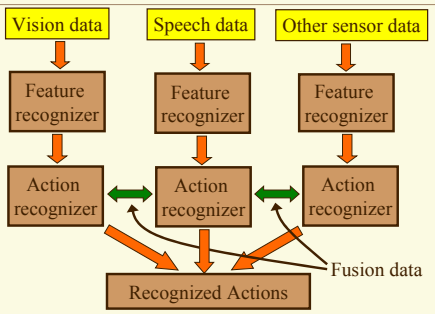
Early fusion



Early fusion

- Early fusion applies to combinations like speech+lip movement. It is difficult because:
  - Of the need for MM training data.
  - Because data need to be closely synchronized.
  - Computational and training costs.

Late fusion



Late fusion

- Late fusion is appropriate for combinations of complementary information, like pen+speech.
  - Recognizers are trained and used separately.
  - Unimodal recognizers are available off-the-shelf.
  - Its still important to accurately time-stamp all inputs: typical delays are known between e.g. gesture and speech.

Contrast between MM and GUIs

- GUI interfaces often restrict input to single non-overlapping events, while MM interfaces handle all inputs at once.
- GUI events are unambiguous, MM inputs are based on recognition and require a probabilistic approach
- MM interfaces are often distributed on a network.

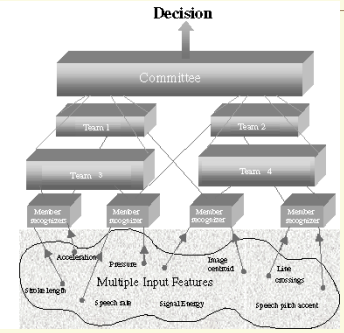
Agent architectures

- Allow parts of an MM system to be written separately, in the most appropriate language, and integrated easily.
- OAA: Open-Agent Architecture (Cohen et al) supports MM interfaces.
- Blackboards and message queues are often used to simplify inter-agent communication.
  - Jini, Javaspaces, Tspaces, JXTA, JMS, MSMQ…

Symbolic/statistical approaches

- Allow symbolic operations like unification (binding of terms like "this") + probabilistic reasoning (possible interpretations of "this").
- The MTC system is an example
  - **Members** are recognizers.
  - **Teams** cluster data from recognizers.
  - The **committee** weights results from various teams.

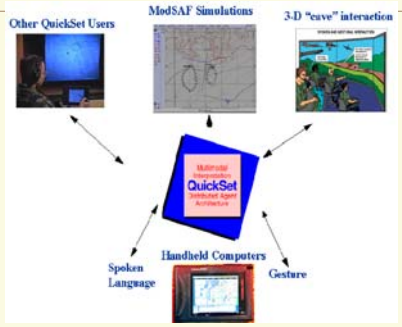
MTC architecture



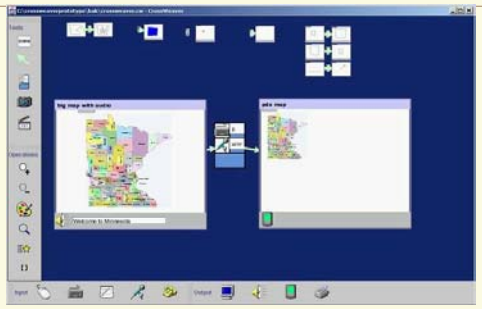
MM systems

- Designers Outpost (Berkeley)

MM systems: Quickset (OGI)



Crossweaver (Berkeley)



Crossweaver (Berkeley)

- Crossweaver is a prototyping system for multi-modal (primarily pen and speech) UIs.
- Also allows cross-platform development (for PDAs, Tablet-PCs, desktops.

CS 160: Lecture 24

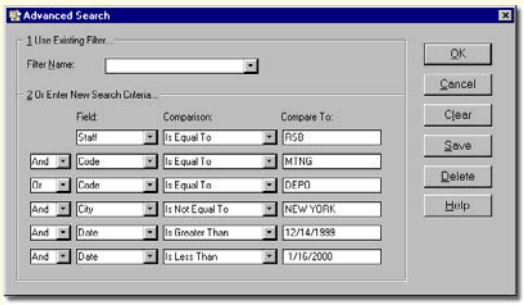
Professor John Canny
Spring 2003

Database queries

* Query languages like SQL are widely used, but are hard to learn and easy to make mistakes with.

```
SELECT DOCUMENT# FROM JOURNAL-DB
WHERE   (DATE >= 1994 AND DATE <= 1997)
  AND   (LANGUAGE = ENGLISH OR FRENCH)
  AND   (PUBLISHER = ASIS OR HFES OR ACM)
```

Visual Query Builders



QBE: Query By Example

- User chooses a record (Database) or document (search engine) and specifies "more like this".
- User can also pick a segment of text, even a paragraph, from a good document and use it as a search query (search engines only).

Multidimensional Scaling

- Multi-Dimensional Scaling (MDS) is a general technique for displaying n-dimensional data in 2D.
- It preserves the notion of "nearness", and therefore clusters of items in n-dimensions still look like clusters on a plot.

Multidimensional Scaling

- Clustering of the MDS datapoints (discussion topics)

Multidimensional Scaling

- MDS can be applied to search engine results easily because they automatically have a high-dimensional representation (used internally by the search engine).
- The MDS plot helps organize the data into meaningful clusters. You can search either near your desired result, or scan for an overview.

Tasks for a visualization system

1. Overview: Get an overview of the collection
2. Zoom: Zoom in on items of interest
3. Filter: Remove uninteresting items
4. Details on demand: Select items and get details
5. Relate: View relationships between items
6. History: Keep a history of actions for undo, replay, refinement
7. Extract: Make subcollections

Visualization principles

- To support tasks 1 & 2, a general design pattern called "focus+context" is often used.
- Idea is to have a focal area at high resolution, but keep all of the collection at low resolution.
- Mimics the human retina.

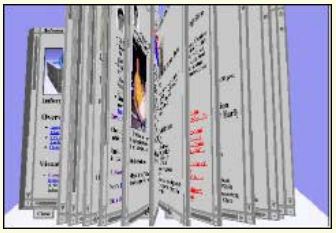
Distortion

- Several visualization systems use distortion to allow a focus+context view.
- "Fisheye lenses" are an example of strongly enlarging the focus while keeping a lot of context (sometimes the entire dataset).
- Many of these were developed at Xerox PARC.

Using 3D

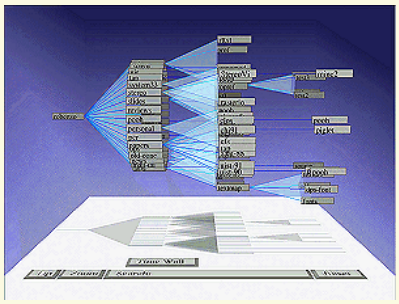
- People perceive a 3D world from 2D views, so it seems like we could use 3D structure to advantage.
- Several systems (also Xerox PARC) have tried this.
- Use 3D spatial memory and organization to speed up navigation.

WebBook



Web Forager

Representing Hierarchies



CS 160: Lecture 25

Professor John Canny
Spring 2003

Aesthetic Principles

- Simplicity
- Scale, Contrast, Proportion
- Organization and Visual Structure
- Grid-based Design

Quantitative Evaluation

- Used to measure differences (b/w UIs).
- Dependent and independent variables.
- Within vs. Between subjects experiments.
- Q: which is better with few subjects?

Social Psychology

- Mere presence influences speed, error rates, improves well-learned tasks.
- Attributions of behavior have an actor-observer effect.

- Groups influence our perception of self and others through norms (reference groups).

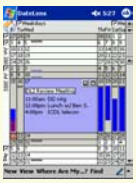
CSCW

- Asynchronous groupware: email, etc.
- Synchronous groupware: video, audio,..
- Issues with videoconferencing.
- Face-to-face vs. email.

- Grudin's 8 challenges for CSCW.

Information design and viz.

- Information tasks (4).
- OAI model, action/object hierarchies.
- 4-phase search pattern.
- Viz techniques:
  - 2D projection: MDS
  - Focus+Context
  - 3D viz



Error Handling

- Error recovery is a "normal" process.
- Types of errors: slips and mistakes; Capture and description errors.
- Five responses to errors: Gag, warn etc.
- Recovery.

Help systems

- 4 Types of help: quick reference, task-oriented…
- Minimalist help systems.
- Adaptive help - user modeling - knowledge representation.
- Design/implementation issues.

Multimodal systems

- Multi-modal systems provide advantages in certain environments and for certain users.
- Speech and pointing are complementary.

- Early vs. late fusion, advantages/ disadvantages.