September 25 2014

LECTURE 7, DIRECT MANIPULATION WINDOWS, ICONS

Example from HCI 2012

- Project for Virtual Pool Table
- Can we make a “look and feel” pool table.
- Technical aspects need be solved
- How do users respond to the “virtual” table.
- Realized with the Processing environment

The Virtual Pool Table

Review #6

- Interaction Styles
  - Command Line
  - Natural Language
  - Speech
  - Key-Modal
  - Menus
  - Direct Manipulation
  - Form fill-in
- Interaction Styles
  - Linguistic
  - Key-Modal
- Hybrid Styles
  - WIMP + CLI
  - WIMP + Speech
  - WIMP + Form

The basic interaction style in modern GUI's

DIRECT MANIPULATION

Direct Manipulation

- **W**indows, **I**cons, **M**enus, **P**ointers, aka WIMP, Forms, (other objects)
- Objects can be acted upon directly by user with pointing device
- Choice of forms and icons important
  - a. match:
    designer's representation - user's understanding
  - b. choice of metaphors
  - c. explicit attention to ways in which instantiated and 'real' objects differ
  - d. cultural bias of representation

Direct-Manipulation Systems

Technologies that derive from word processor:

- Integration
- Desktop publication software
- Slide-presentation software
- Hypermedia environments
- Improved macro facilities
- Spell checker and thesaurus
- Grammar checkers

Direct-Manipulation Systems

Video games

- Nintendo Wii, Sony PlayStation, Microsoft Xbox, Kinect
- Field of action is visual and compelling
- Commands are physical actions whose results are immediately shown on the screen
- No syntax to remember
- Most games continuously display a score
- Direct manipulation in "the Sims"
- Second Life virtual world
- Spore
- Myst well received
- DOOM and Quake controversial

Examples Direct-Manipulation

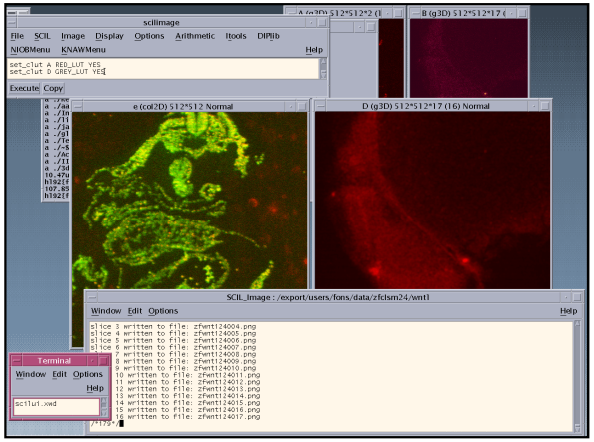
Guitar Hero video game

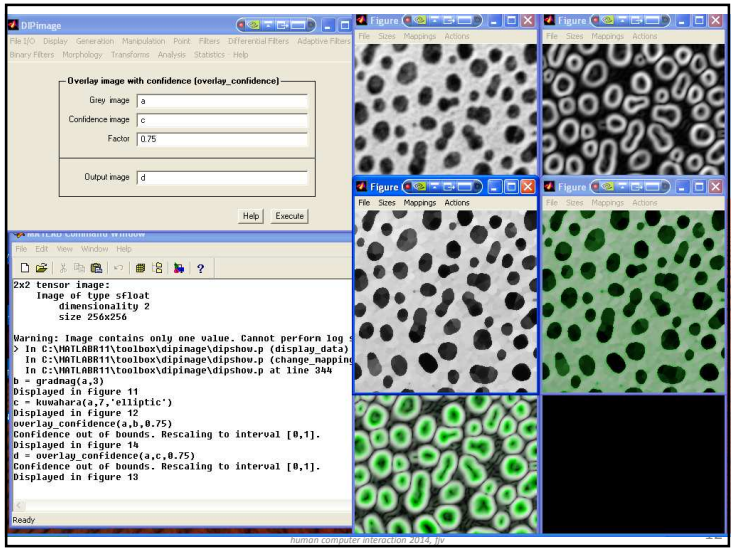
Direct Manipulation

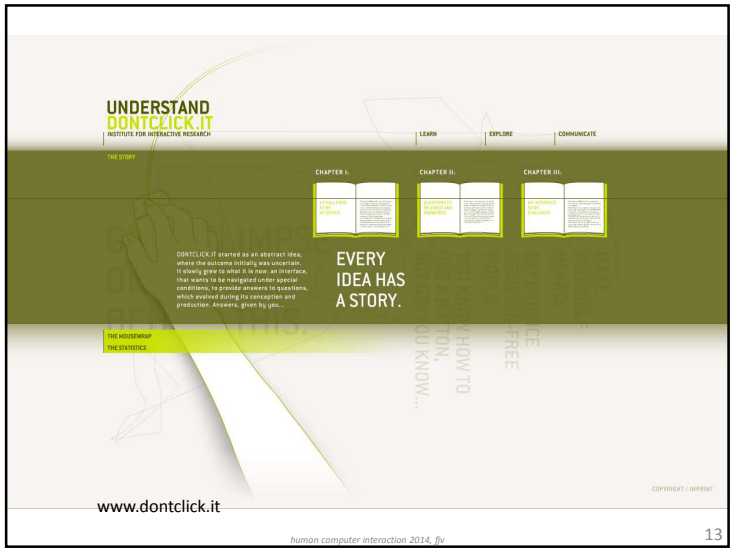
Visual representation of objects and tasks that can be manipulated in arbitrary orders by users: GUI

Advantages	*Disadvantages*
• visually presents task concepts • learnability • retention over time • error avoidance • encourages exploration • high subjective satisfaction	• can be difficult to program (especially error handling) • non-sighted users • *requires high-end equipment* • *New developments require to rethink the principles of Direct Interaction*

Hybrid Systems







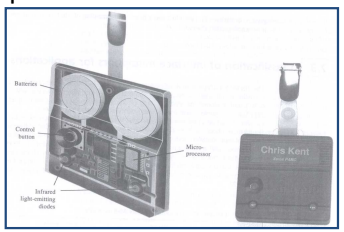
Evaluation of Interaction Styles

Depending on Context and:

- Time to learn
- Speed of performance
- Rate of errors
- Retention over time
- Subjective satisfaction

Ubiquitous Computing

- Introduced by Weiser 1991:
  "Invisibly enhancing the world that already exists"
- Computers disappear in environment
- Goal:
  "Make the interface metaphor visible to user"
- Beyond GUI and WIMP
- Multi-modal!
- Sensor based



Fitt's Law

Relation between

Target Size – Distance to Target (GUI)

Helps in Efficient Screen Design

Fitt's law:

$$T = k \log_2 (D/S+0.5)$$

T = time to move hand to target

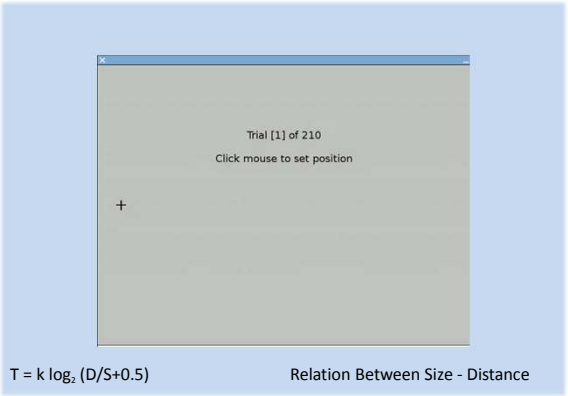
k ≈ 100ms,

D = distance,

S = size of target

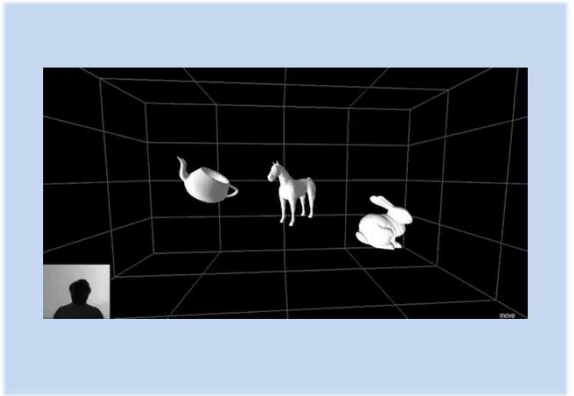
- How would this work for newer interactions ?

Fitt's Law Experiments



$T = k \log_2 (D/S+0.5)$ Relation Between Size - Distance

Interaction, Mental Model, Fitt's Law



The Basics of Direct Manipulation

WINDOWING SYSTEMS

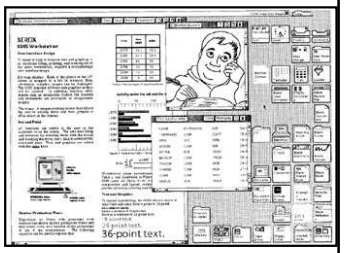
Windowing Systems

Windows:

- Areas of visual display
- Rectangular or rectilinear in shape
- Divide display area in virtual displays
- Can display data/action from different machines, applications, files, views, etc.
- Windowing systems:
  - manage input and output resources
  - like OS manages CPU time, disk space, memory, etc.

Windowing Systems

- Archetype: Xerox PARC Star/Dynabook
- OS Window Manager:
- Apple Macintosh
- MS-Windows
- Window Manager:
- X11
  - X11/R6
  - Open look
  - Motif
  - CDE



Tasks Windowing System

- Managing input
  - Dispatching input events
- Changing Focus
  - Click-to-focus
  - Mouse-over-focus
- Managing a Single Window
  - Dragging
  - Scrolling (more info than fits the screen)
- Managing Multiple Windows
  - Iconify (not all windows are displayed at once)
  - Tiling
  - Overlapping

Example – RedHat Gnome

Windows & Widgets

- Interface Components are Virtual Devices, referred to as *widgets*
- Widgets: *Set of "standard components" to compose GUI*
  - Buttons: push, radio, check box
  - Scrollbars
  - Menus
  - Text boxes
- Window Systems are build from Widgets, also referred to as the Widget Set

Windows Working Set

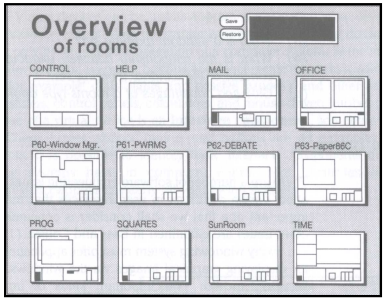
- Set of windows needed to carry out a particular task effectively (Card, 1984)
- Multiple tasks are done simultaneously
- Adapt Window-Set to that fact
  - prevent searching for particular window
  - prevent time wasted on reorganizing
  - prevent time wasted on manipulating
- Window-Working Set for each stage
- Later adapted to room-building paradigm

Room-Building Concept

- How do people actually use windows: Rooms (Henderson & Card 1987)
- Recurring configurations
- Particular (recurring) configurations are stored in a room
- Room = icon + name (HP-VUE, CDE, Linux)
- Whole session is the building

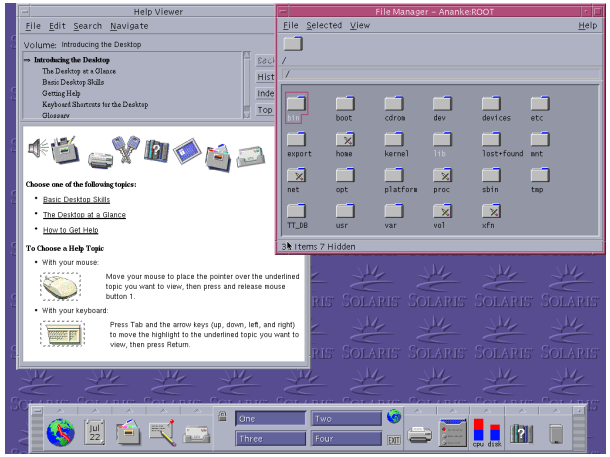
Windows and Rooms

Card & Henderson 1987, classic schema



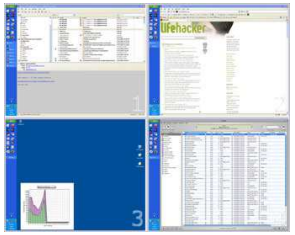
Tablets still have the rooms concept

Windows Room-Building



Virtual Desktops

- Increase screen capacity,
  - Virtualization
- Video memory,
  - Rapid switching
- Organize by putting applications in different screens:
  - window-room concept.
- Connect Application to screen
  - window-room concept.



Graphical User Interfaces

- Virtual Devices:
  - Exists by virtue of operation of computer system.
  - Buttons, Active screen areas, Sliders, Control panels …
- Interface Components are Virtual Devices: ***widgets***
  - set of "standard components" to compose GUI
  - buttons
    - push
    - radio
    - check boxes
  - scrollbars
  - menus
  - text boxes

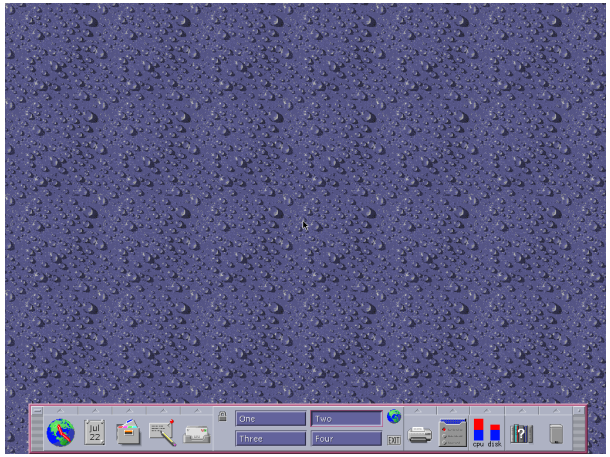
Windowing Systems I

- Provide infrastructure to
  - Support common UI-related services
- Provide functionality for
  - Input/Output device handling
  - Which window gets which input/output
- Window manager
  - Create and organize window
  - Implement interaction in those windows
  - Dispatch events

Windowing Systems II

- Top level windows known as *root windows*
  - Provide an abstraction to separate applications
  - Windowing system arbitrates resources among these
  - Abbrev. (main) Window, constitutes one application
- Each root window belongs to an application.
  - All descendant windows belong to same application
  - Not used by OLE (ActiveX); include window/data from other application

Root Window



Major Issues in GUI programming

- How to decompose the UI into:
  - Interactive objects (widgets)?
  - Tasks (task decomposition)
- How to distribute inputs to the widgets?
- Contrast
  - Sequential (standard) program flow
  - Event-driven program flow

Sequential Programs I

- Program takes control, prompts for input
- Examples include
  - command-line prompts (DOS, UNIX)
  - LISP interpreter
  - Chatterbot (Q&A interface style)
- The user waits on the program
  - Program tells user it's ready for more input
  - User enters more input

Sequential Programs II, C(++)

```
int foo (struct rectangle *rect)
{
  // do something
}
int main ()
{
  …
  foo (rect);
  …
}
```


Sequential Programs III

- The user waits on the program
  1. Program tells user it is ready for more input
  2. User enters more input
  3. Program responds and goes back to

  Question …
- How to model the many actions a user can take?
  - For example, a word processor?
  - Need to do editing, inserting, formatting, etc.

Sequential Programs IV

- Control resides with the *program*
- You can follow the program
  - Start at main()
  - Keep following the function calls / threads
  - Understand functions

Graphical User Interfaces

- Control resides at the *user side*
- The user
  - Can take any action
  - Out of N possible options
  - At any possible time
- Reading the program is difficult
  - Reading event handlers
  - Understanding each point of interaction

Event handling

- User files actions
- These actions generate *events*
  - *event handling*:
    way GUIs and/or windowing systems deal with events.
  - Thus, this style of programming is called:
    event-based programming.
  - event-based programs are often *harder to read*

Simplified Main event loop

```
Initialization(); // Main AppWindow
while (not time to quit) {
  E = getNextEvent ();
  dispatch (E);
}
```


Events

- An *event* is something "interesting" that happens in the system, e.g.
  - Mouse button goes down
  - Item is being dragged
  - Keyboard button was pressed
  - I/O device filed action
  - Windowing action (iconify etc. …)
- Communication from user to computer
  - All done through events
  - Task language (Abowd & Beale) = events

Windowing Events

- Windowing events on window
  - creation / destruction
  - opening / closing
  - iconifying / deiconifying
  - selection / deselection
  - resize
  - redraw
    - window manager tells the application to redraw within a certain region;
    - the application does the actual redrawing

Where to dispatch?

- Decompose windows hierarchically
  - main display
    - MS Powerpoint
      - Title bar
        - » iconify icon*
        - » maximize icon
        - » close icon
      - Menu bar
      - Tool bar
      - Slide
        - » Title bar
        - » Text window
    - Firefox browser
      - ….
- Decomposition is organized in an Interactor tree
  - Interactor tree is used for dispatching

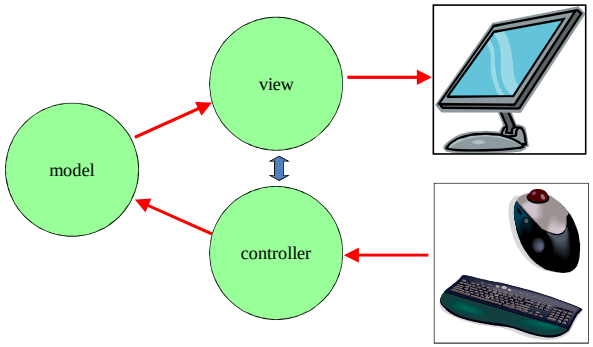
Model View Controller

- Model =
  information that application manipulates
  - represents real world
- View =
  visual display of the model.
  - change in the model requires change in the visual presentation thereof.
- Controller =
  receives all input events and decides upon meaning and process

Model-View-Controller Architecture

- What is MVC?
- Smalltalk used MVC – model–view–controller
  - model – internal logical state of component
  - view – how it is rendered on screen
  - controller – processes user input
- should be conceptually separate
  - really separate Model and View/Controller
  - why?
- Say you change the model: what happens?
  - damage()
  - then for each view: redraw()

MVC: model - view - controller

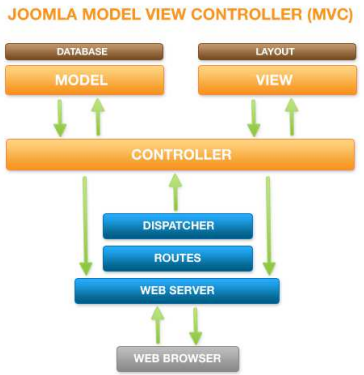


MVC issues

- MVC is largely pipeline model:
  input → control → model → view → output
- but in graphical interface
  - input only has meaning in relation to output

  e.g. mouse click
  - need to know *what* was clicked
  - controller has to decide what to do with click
  - but view knows what is shown where!
- in practice controller 'talks' to view
  - separation not complete

MVC underlies web programming



Damage and redraw

- Region of screen that is no longer consistent with model is *damaged* (terminology from MVC)
- if Model changes, it notifies Views (through Controller)
  - when notified of change,
  - the views report regions that are damaged to the system
  - canvas::damageRegion(region)
  - these requests for updates are *batched*
- main event loop eventually picks up 'redraw' event
  - before handling new input events it does a redraw on the window

Windowing systems & GUI design tools

- Handle clipping, redrawing, etc.
- Do not handle separation M-V-C
- This is your responsibility
- Also: realize that you cannot simple "draw something" and expect it to appear
  - everything happens in response to events

How do we build GUIs? (1)

- Decide what we want to build (paper design)
- Determine which events should be sent to which widgets
  - input
  - internal
  - etc.

How do we build GUIs? (2)

3. Lay out appropriate widgets using GUI tool
4. For each widget add missing event handlers
   - "event listeners" (Java)
   - "slots" (Qt)
5. Connect event senders to event receivers
   - Upon necessity
6. Add your code
   - The "product" functionality
   - cf. Rules Schneiderman – Norman

Conclusive Remarks

Defining paradigm in interactive software architectures : *event based programming*

1. USER is in control
2. No uniform flow of control through program

Review #7a

- Window systems
  - Tasking
  - Working Sets,
  - Room-Building concept
- Widget set
  - Widgets
  - Basic components
- Programming a GUI
  - Event based programming
  - User is in control
  - Model View Controller achitecture

Typical Element of Direct Manipulation: Icon

ICONIC REPRESENTATIONS ICONS

Icons

- Direct manipulation
- Definition
- Advantages and disadvantages
- Factors influencing meaningfulness
- Form and function
- Design guidance
- Classification

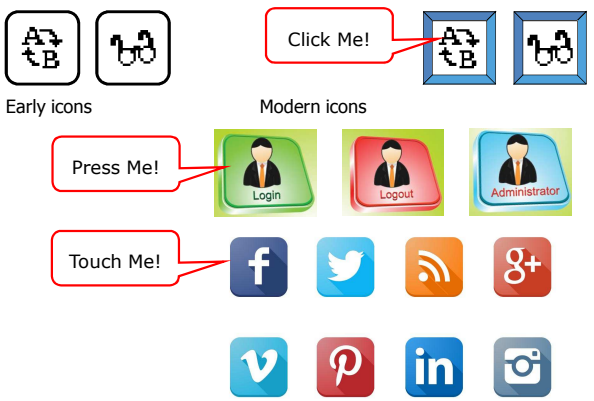
Icons are old ...

100000 yrs ago
Pictographs
Ideographs

Icon types (Marcus 1984)

- Icon
  - Something that looks like what it represents
- Index
  - Sign caused by object that it represents
  - Depending on previous experiences
- Symbol
  - Sign completely arbitraty in appearance
  - Meaning must be learned

Icons & Affordance



Definition of Icons

- Pictographic symbols that are used to focus on essential features
- Represent underlying
  - objects
  - data structures
  - processes
- … in a form which corresponds to the real world
- Can be entertaining, clever and visually appealing (also dumb, annoying and useless)

Iconic Representation (Rogers 1989)

- Similar icons:
  - Visually analogous to concept / object
- Example icons:
  - Things commonly associated with action
- Symbolic icons:
  - Use image that represents concept
- Arbitrary icons:
  - No direct relationship with concept




What do these icons refer to?

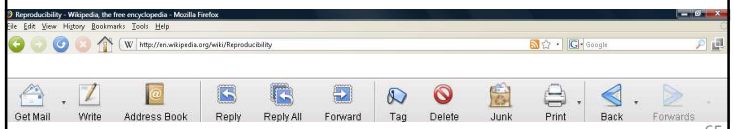
Less accidents	Symbolic
Less aggression	Symbolic
No Photo-controls	Symbolic
Less fuel use	Example

Icons very Commonly used

Word 2000

Netscape 7.0

FireFox 3.0/ Thunderbird 2.0



Advantages of Icons (1)

- Recognition
  - Users remember meaning more easily than command names
  - Searched and selected faster than text

- Compactness
  - Can convey meaning more rapidly than symbolic language – little screen space
  - Use less space than text, i.e. more features can be presented on one screen

Advantages of Icons (2)

- Comprehensibility
  - Can convey objects, data and actions

- Universality
  - Relatively language- and culture-independent
  - Facilitates porting to international markets
- Plenty available for re-use
- Can represent dynamic aspects
  - however: no clear evidence better than static
  - Micons (moving icons or animated icons)

Disadvantages of Icons

- Ambiguity

- Dependent on user, task and context

- Cannot completely replace words in some complex situations

- Expensive - difficult to design good new icons
- Micons - too many distract

Moving Icons: M*Icons*

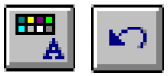
Moving icons can be

- Static
  - Only change with system state
- Dynamic
  - Independent of system state
- Portray dynamics
- Clarify action is ongoing
- Be annoying




Meaningfulness of Icons (1)

- Context
  - Specific ~ General
  - Easy ----------Difficult to deduce meaning
- Tasks
  - Visual ~ Verbal
  - Easy ----------Difficult to design




Context can change meaning

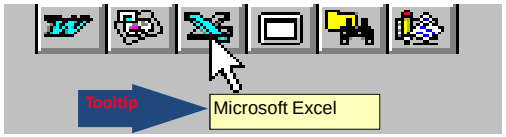
What does this icon mean:

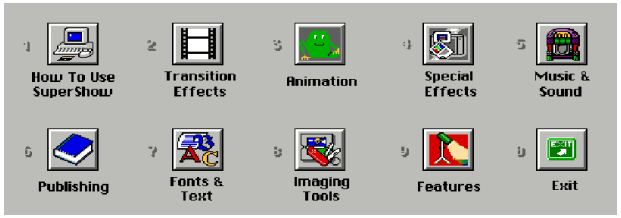
What does this icon mean:

Meaningfulness of Icons (2)

- Text
  - Addition increases understanding if:
    - Abstract
    - Vague
    - Large icon set
  - Meaning clear
  - Recognition of icons


Icons: Addition of Text





Meaningfulness of Icons (3)

- Concept
  - Concrete ~ Abstract
  - Easy ---------- Difficult to design

- Distinguishable
  - From other icons

Range of Icon Functions (Rogers, 1989)

Function	*Example*
• Labeling • Indicating • Warning • Identifying • Manipulating	• Menu item • System state • Error message • File storage • Tool for zooming and shrinking
• Container	• Object for placing discarded objects
• *Gestalt* pattern	• Structure in programming language

Visual Reflection Interaction State

Guidance in Design of Icons

- General HCI guidelines apply
  - Colour, Placement, Visibility, Affordance
- Re-use / re-work existing designs
  - Icon already known, Less work
- Avoid symbols unless meaning already known
  - What is the context of usage
- Use icon design tools to experiment
- Test representation with users

Icon Design

- Poorly understood by computer professionals
- Everyday icons are the subject of much research
  - e.g. flight arrivals /departures
- Trend towards higher quality graphics

4 rules on Icons

- Look different from all other icons
- It is obvious what it does/represents
- It is recognizable in 16x16 pixels image ?
- Look good in Black & White and Color

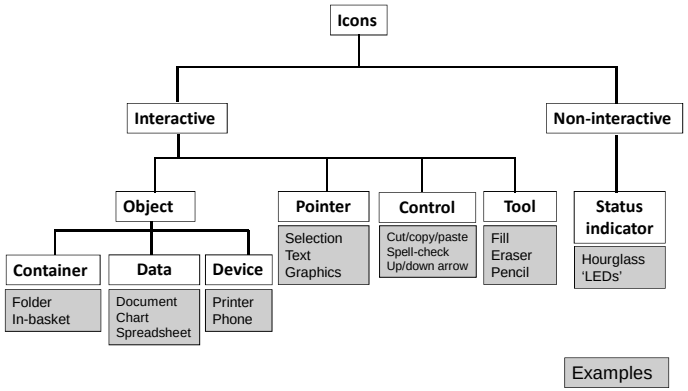
Classification ISO Standard

- Interactive icons
  - Mediate user interaction with software application
- Non-interactive icons
  - Status indicators
  - System State Change

IBM Icon Classification

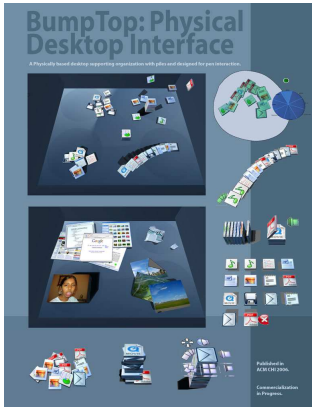
- Common User Access Workplace Environment
- Organized by IBM in 1988
- 3 Categories of Icons (interactive)
  - Container (e.g. a folder)
  - Data (e.g. a document)
  - Device (e.g. a printer)

Icons Classification (ISO/IEC Based)



Using Icons to enhance Metaphor

- The BumpTop interface
- Pile as organizational entity
- Icon to invoke physical interaction
- Pen/Gesture based
- Appeals on gestalt of physical entities on desk
- Physical properties of icons



Icons or eye-cons

Icon:Greek εικων, eikon: image

An artistic visual representation or symbol of anything considered holy and divine, such as God, saints or deities. An icon could be a painting (including relief painting), sculpture, or mosaic.

Auditory Icons, Audible interfaces

- Example: SonicFinder
  - Mac interface extension
  - W.W. Gaver 1989
- Auditory icon
  - has a semantic relationship with function
  - Sound directly reflects physical interaction of object
    - Mailbox, different sound for mail-file-program
    - Similar to real mailbox
    - Position of mailbox has effect on sound
      - obscured, minimized
    - Waste paper basket
    - What are you throwing away, paper? slides? program? etc …
    - Produce typical sound, e.g. crisping paper

Audio-Aids: EarCons

- Short sound fragments that help operate the system
- Cell-phone
  - Incoming call
  - SMS message
  - Low battery
- Computer
  - Used in feed-back, action completes
  - Draw Attention
  - e.g. incoming mail, use specific sound per mail-type
- Timbre and Pitch
- Style (like in musical style) and Rhythm

Paint File Earcon

Auditory Slider Earcon

EarCons

- Frist proposed by Meera Blattner (LLNL)
- Stephen Brewster (Glasgow Univ.) lots of research in implementation
- Audible icons: structures that are abstract audible messages to communicate complex information to (non-visual) users
- **earcon** (EER.cawn) *n*. An auditory icon: a sound that is used to represent a specific event or object.

Tactile aids: Tactons

- Tactile icons: structures abstract tactile messages to communicate complex information to non-visual users
- Ubiquitous computing
- Compare to Braille scale/time of structured information of a tacton is much smaller
- Design according to “prescribed” sequence of operations: haptic I/O device
  - transducer
  - freq. pulses, amplitudes, wave forms: Rhythm!
- Depending on transducer parameters

Tactor: Tactile devices for Tactons

Kineticons

- Iconograhic scheme based on motion
- Motion: geometric manipulations on element
- Kineticons (Chris Harrrison)
  - Do not alter pixel content of an element
  - Differ from Micons
  - Seen in variety of sytems, e.g. SmartPhone, Tablet
- Kinetic behaviour
  - On desktop
  - Make a Vocabulary
  - Meaning

Examples Kineticons

Emotion-Aids: EmotiCons

- Originally a sequence of ordinary characters used in e-mail, chat, SMS:
  - conveying emotion through an icon
  - converted to real icons
- See also: http://www.muller-godschalk.com/
- I am rushed:

- I am excited:

Iconic Gestures (Gesticons)

- Gestures are Iconic
- eg. eMessenger gestures
- eg. BumpTop gestures (see also other applications)
- Gestures are produces
  - by hand interaction
  - simulation of hand interaction

Review 7b

- Icons
  - Meaningfulness
  - Classification
- Other Iconic types
  - Audible
  - Earcons
  - Tactons
  - Kineticons
  - Emoticons
  - Gestures (Gesticons)
- How are we building interactions.