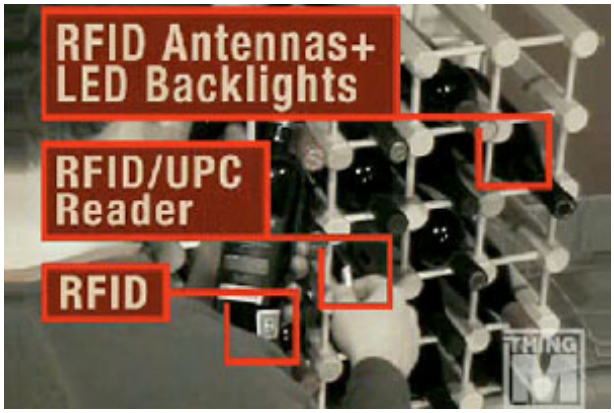
Low-Fidelity Prototyping

CS160: User Interfaces

Maneesh Agrawala



http://thingm.com/sketches/winem.html

Review: Event Dispatch Loop

Mouse moved (t_0,x,y)

Event Queue
- Queue of input events

Event Loop (runs in dedicated thread)
- Remove next event from queue
- Determine event type
- Find proper component(s)
- Invoke callbacks on components
- Repeat, or wait until event arrives

Component
- Invoked callback method
- Update application state
- Request repaint, if needed

Individual Programming Assignment
(due Mar 2)

Design and Implementation Components
- Sketches of 3 alternatives, pick a favorite
- "Discount" user studies in section (Feb 25-26)
- Write up what you learned from the study
- Note how you changed your interface as a result
- Implement user interface

Individual Programming Assignment
(due Mar 2)

Project Management/To-Do List

Tasks have the following properties:

- Task Name
- Percentage Completed (0-100%)
- Start and End date
- Priority
- List of people assigned to the task
- URL related to the task

Checklist view

- Include checkbox to automatically set completion percentage to 100%
- You should be able to see the completion percentage

Timeline view

Magic lens: http://dohistory.org/diary/exercises/lens/index.html

Topics

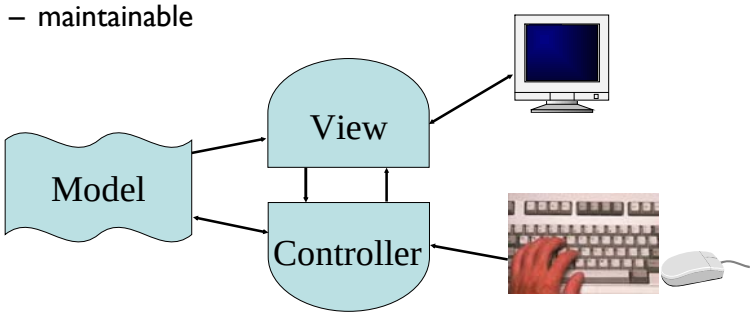
- Model View Controller
- Paper Prototyping
- Video Prototyping
- Wizard of Oz prototype testing

Model-View-Controller Architecture

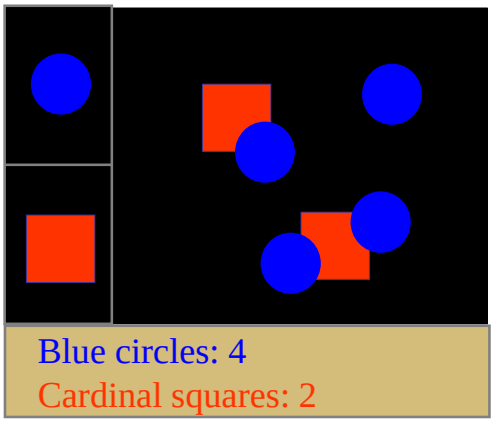
Model-View-Controller

Architecture for interactive apps
- introduced by Smalltalk developers at PARC

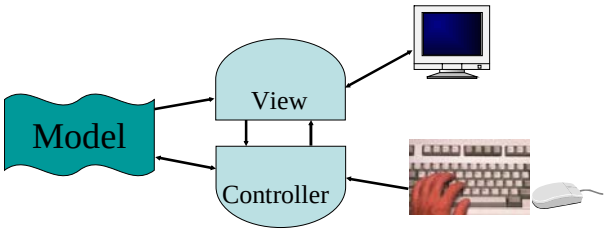
Partitions application in a way that is
- scalable
- maintainable



Example Application



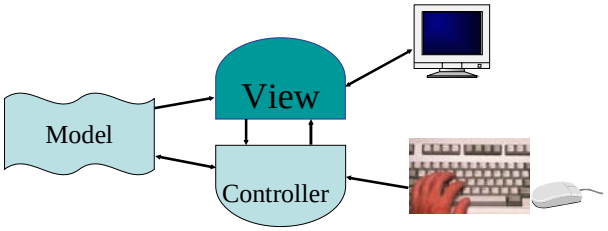
Model



Information the app is manipulating

Representation of real world objects
- circuit for a CAD program
  - logic gates and wires connecting them
- shapes in a drawing program
  - geometry and color

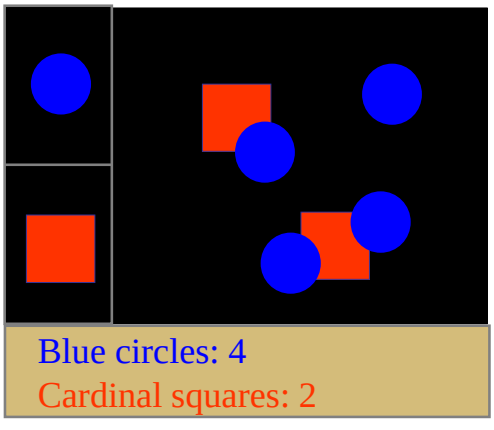
View



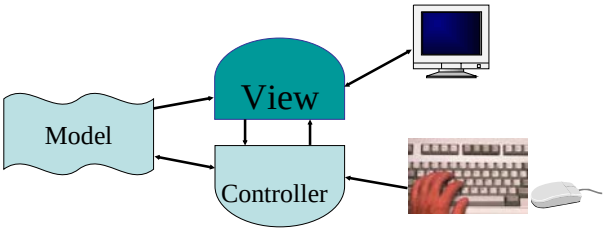
Implements a visual display of the model

May have multiple views
- e.g., shape view and numerical view

Multiple Views



View

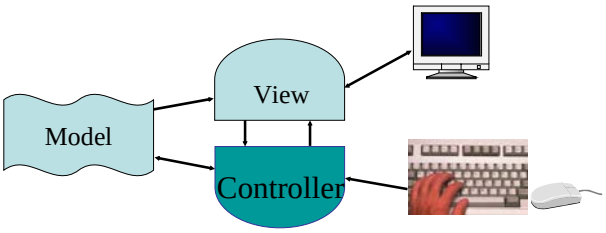


Implements a visual display of the model

May have multiple views
- e.g., shape view and numerical view

Any time model changes each view must be notified so it can update
- e.g., adding a new shape

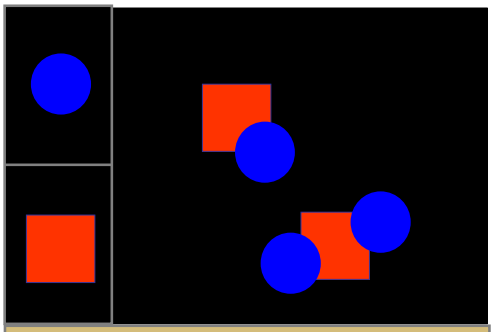
Controller



Receives all input events from the user

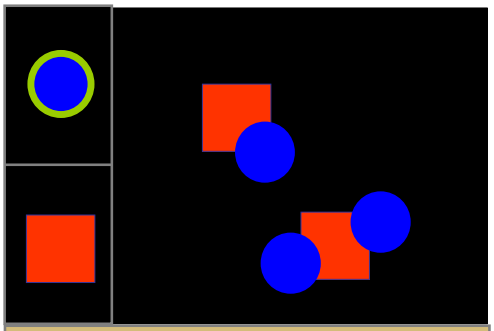
Decides what events mean and what to do
- communicates with view to determine the objects being manipulated (e.g., selection)
- calls model methods to make changes on objects
  - model makes change and notifies views to update

Controller



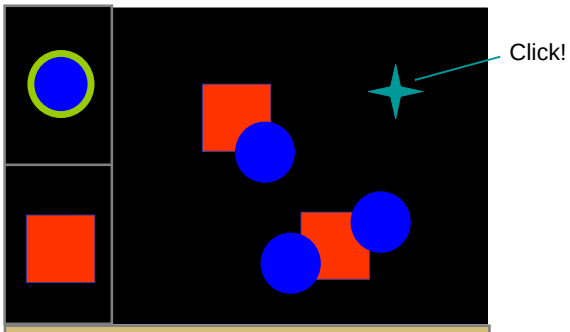
Blue circles: 3
Cardinal squares: 2

Controller

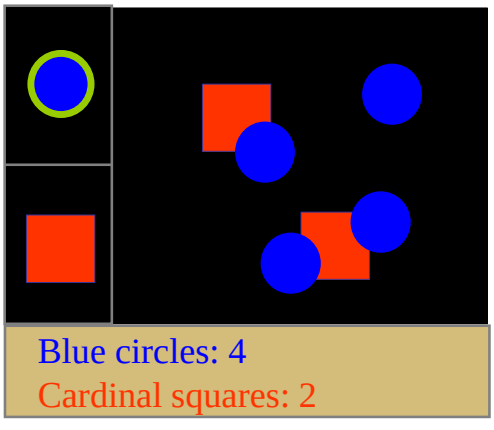


Blue circles: 3
Cardinal squares: 2

Controller



Controller

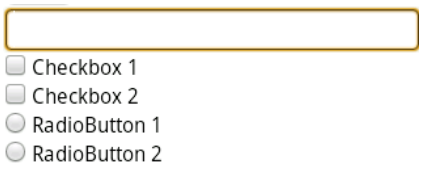


Relationship of View & Controller

"pattern of behavior in response to user events (controller issues) is independent of visual geometry (view issues)" – Olsen, Chapter 5.2

Relationship of View & Controller

"pattern of behavior in response to user events (controller issues) is independent of visual geometry (view issues)" – Olsen, Chapter 5.2



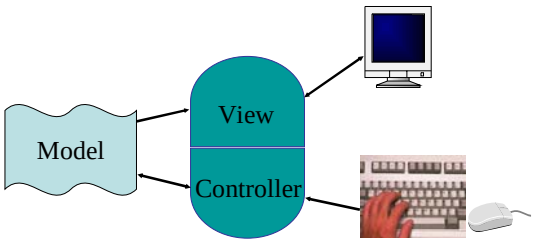
But controller must usually contact view to interpret what user events mean (e.g., selection)

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



Why MVC?

Why MVC?

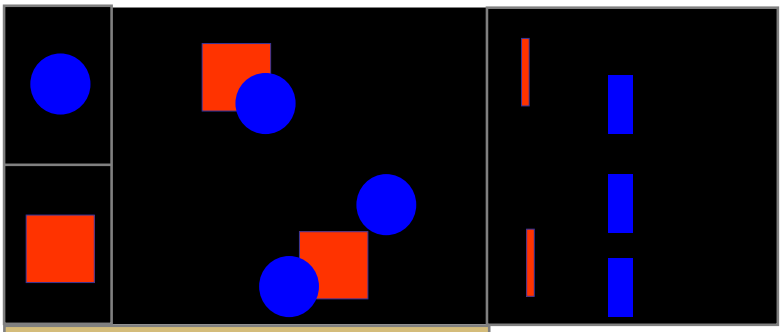
Combining MVC into one class will not scale

- model may have more than one view
  - each is different and needs update when model changes

Separation eases maintenance and extensibility

- easy to add a new view later
- model info can be extended, but old views still work
- can change a view later, e.g., draw shapes in 3-d (recall, view handles selection)
- flexibility of changing input handling when using separate controllers

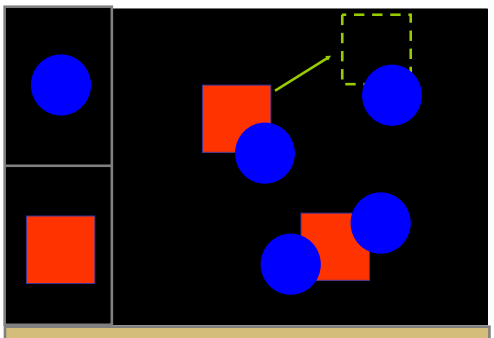
Adding Views Later



Changing the Display

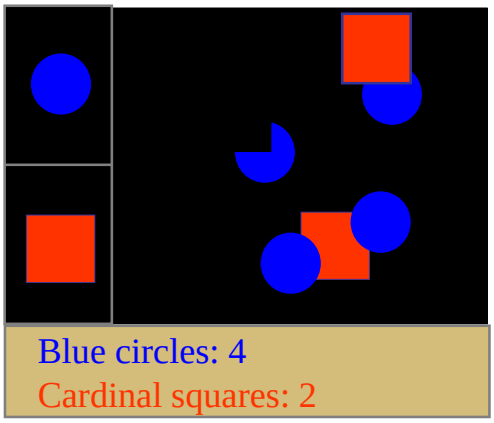
How do we redraw when shape moves?

Moving Cardinal Square



Blue circles: 4
Cardinal squares: 2

Erase w/ Background Color and Redraw



Changing the Display

Erase and redraw
- using background color to erase fails
- drawing shape in new position loses ordering

Move in model and then redraw view
- change position of shapes in model
- model keeps shapes in a desired order
- tell **all** views to redraw themselves in order
- slow for large / complex drawings
  - flashing! (can solve w/ double buffering)

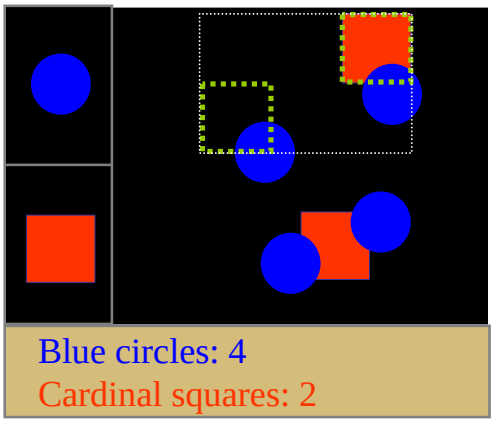
Damage / Redraw Method

View informs windowing system of areas that are *damaged*
- does not redraw them right away…

Windowing system
- batches updates
- clips them to *visible* portions of window

Next time waiting for input
- windowing system calls *Repaint* method
  - passes region that needs to be updated

Damage old, Change position in model, Damage new



Dragging at Interactive Speeds

Damage old, move, damage new method may be too slow
- must take less than ~100 ms to be smooth

Solutions
- don't draw object, draw an outline (cartoon)
  - use XOR to erase fast (problems w/ color)
- save portion of frame buffer before dragging
  - draw bitmap rather than redraw the component

modern hardware often alleviates the problem

Summary

Event-Driven Interfaces
- Hierarchy of components or widgets
- Input events dispatched to components
- Components process events with callback methods

Model-View-Controller
- Break up a component into
  - **Model** of the data backing the widget(s)
  - **View** determining the look of the widget
  - **Controller** for handling input events
- Provides scalability and extensibility

Looking Forward

Containment hierarchy model is now over 20 years old, designed in a context of significantly less processing and graphics power.

Dominant model in use today, and still quite useful, but in many cases limiting.

Limitations:

- Assumes rectangular components
- Limited support for animation
- Level of extensibility (varies by toolkit)

Suitability for next-generation interfaces?

Paper Prototyping

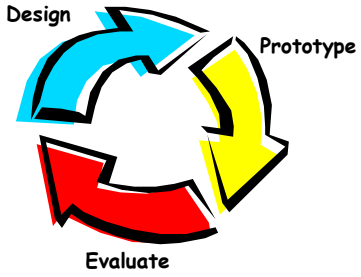
Why Do We Prototype?

Get feedback on our design faster
- saves money

Experiment with alternative designs

Fix problems before code is written

Keep the design centered on the user

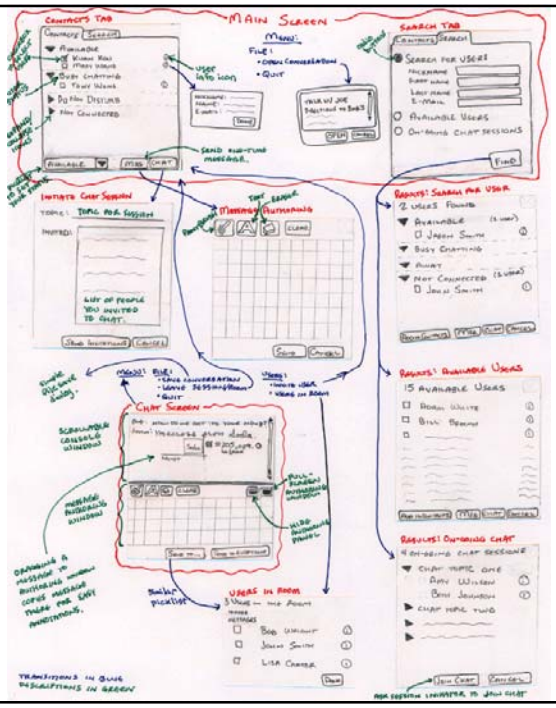


Fidelity in Prototyping

Fidelity refers to the level of detail

High fidelity,
- Prototypes look like the final product

Low fidelity,
- Artists renditions with many details missing



Ink Chat System

Color Coding

- Black: page content
- Red: page titles
- Green: annotations
- Blue: links

Hi-Fi Disadvantages

Distort perceptions of the tester

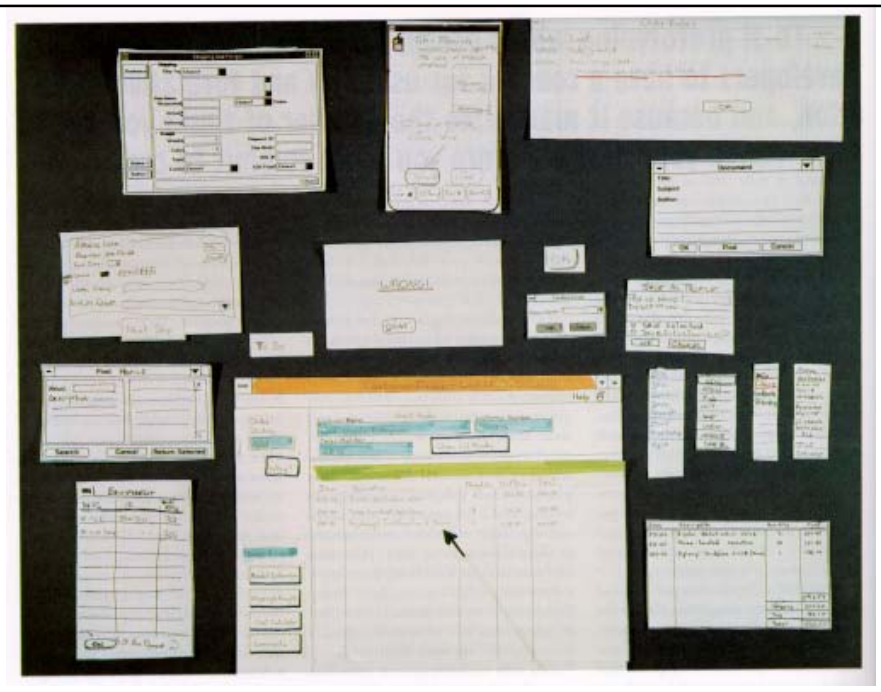
- Formal representation indicates "finished" nature
- People comment on color, fonts, and alignment

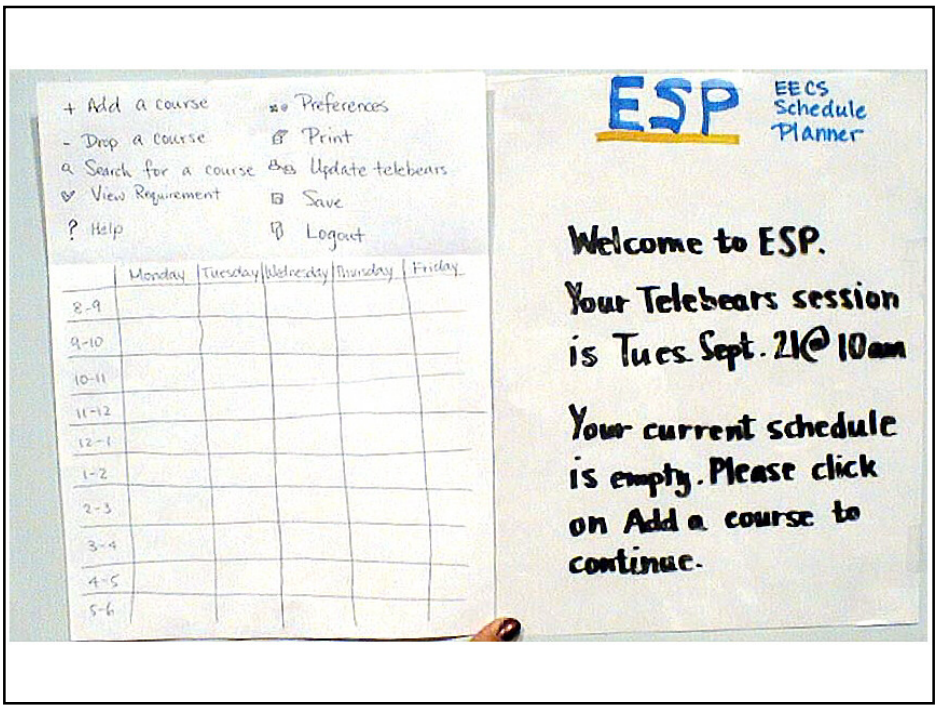
Discourages major changes

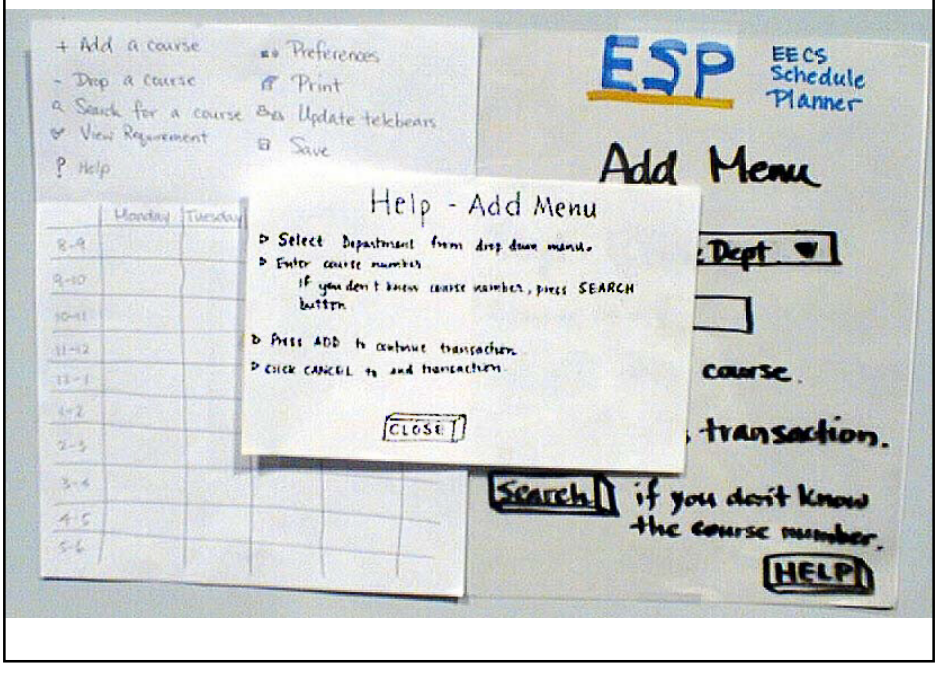
- Testers don't want to change a "finished" design
- Designers don't want to lose effort put into creating hi-fi design

Materials

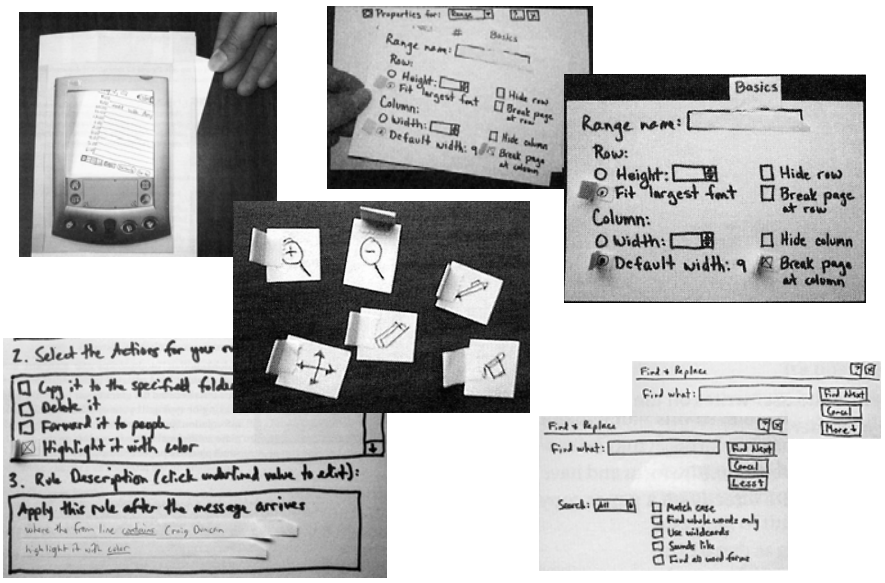
Large, heavy, white paper (11 x 17)
5x8 in. index cards
Post-it notes
Tape, stick glue, correction tape
Pens & markers (colors & sizes)
Transparencies (including colored)
Colorforms (toy stores)
Scissors, X-acto knives, etc.




+ Add a course
- Drop a course
Search for a course
View Requirement
? Help
Preferences
Print
Update telebears
Save
Logout
Monday
Tuesday
Wednesday
Thursday
Friday
8-9
9-10
10-11
11-12
12-1
1-2
2-3
3-4
4-5
5-6
ESP
EECS Schedule Planner
Welcome to ESP.
Your Telebears session is Tues. Sept. 21 @ 10am
Your current schedule is empty. Please click on Add a course to continue.


+ Add a course
- Drop a course
Search for a course
View Requirement
? Help
Preferences
Print
Update telebears
Save
Monday
Tuesday
ESP
EECS Schedule Planner
Add Menu
Help - Add Menu
Select Department from drop down menu.
Enter course number
If you don't know course number, press SEARCH button
Press ADD to continue transaction.
Click CANCEL to end transaction.
CLOSE
Dept.
course.
transaction.
Search
if you don't know the course number.
HELP

Interface Elements



Constructing the Prototype

Set a deadline
- Don't think too long - build it!

Draw a window frame on large paper
- **Draw at a large size, but use correct aspect ratio**

Put different screen regions on cards
- Anything that moves, changes, appears/disappears
- Use greek-ing to indicate text if necessary

Ready response for any user action
- e.g., Have those pull-down menus already made

Use photocopier to make many versions

Video Prototyping

Video Brainstorming

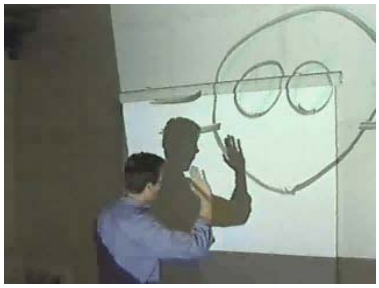
Participants act ideas out in front of a video camera

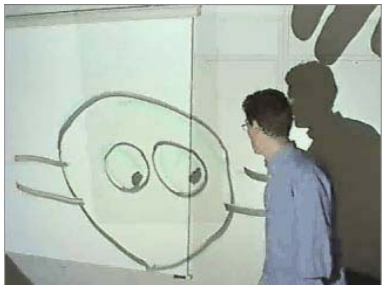
Goal is to create as many new ideas as possible

- each should take 2-5 minutes to generate & capture
- run standard brainstorming session first for ideas

Advantages

- video easier to understand later than notes
- participants actively experience interaction & preserve record of the idea





Video brainstorming of an animated character in *Prototyping Tools & Techniques* by Beaudouin-Lafon & Mackay. Character follows user with its eyes.

Forms of Video Prototypes

May build on paper prototypes or use existing software & images of real settings

Narration optional

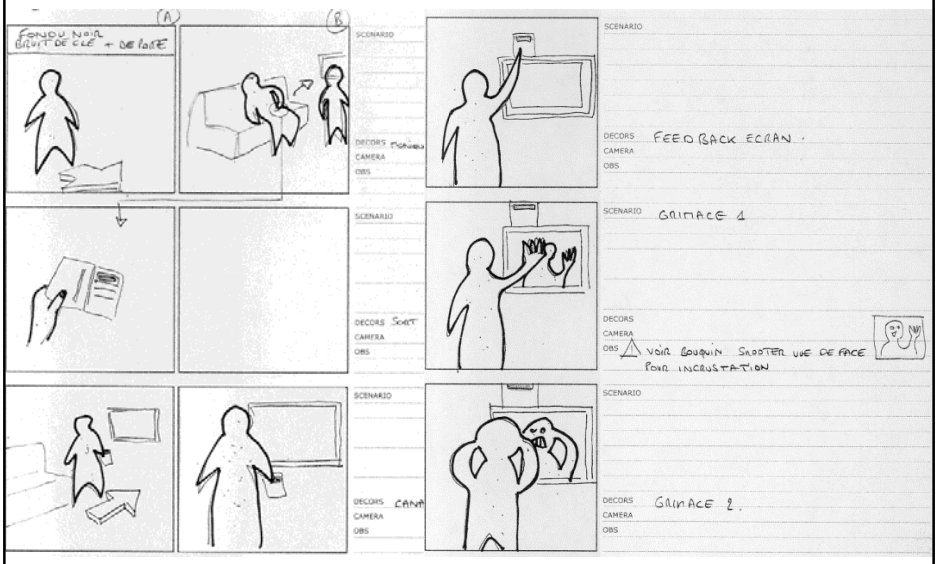
- narrator explains events & others move images/illustrate interaction
- actors perform movements & viewer expected to understand w/o voice-over

With good storyboards, should be able to create video prototype in 1 hour

Creating a Video Prototype

1) Review field data about users & work practices
2) Review ideas from video brainstorm
3) Create use scenario in words
4) Develop storyboard of each action/event w/ annotations explaining what is happening in scene. Put each element on a card.

Storyboard



Creating a Video Prototype

1) Review field data about users & work practices
2) Review ideas from video brainstorm
3) Create use scenario in words
4) Develop storyboard of each action/event w/ annotations explaining what is happening in scene. Put each element on a card.

5) Shoot a video clip for each storyboard card
   - avoid editing in the camera – just shoot in storyboard order
   - hold last frame of a section/shot for 1s
6) Use title cards to separate clips (keep it onscreen for 3s)
   - if you make an error, rewind to last title card & reshoot

Video Prototyping

- Illustrate how users will interact w/ system
- Unlike brainstorming, video prototyping contracts design space
- Quick to build
- Inexpensive

- **Better illustrates context of use**

Example Videos

Univ. of Washington

- Cluster: Andy Hou & Kevin Chiu
- Don't Forget: Chris Govella & Peter Woodman
- Don't Forget 2: Carolyn Holmes & Fred Potter

Stanford

- Energy Usage Information: Lisa Seeman

Tips & Tricks

Add structure to better explain context
- begin with a title
- follow with an "establishing shot"
- create series of closeup & mid-range shots, interspersed with title cards to tell the story
- place a final card with credits at the end

Use colored paper for title cards to make easy to find when editing/searching video

"Time-lapse photography" lets images appear & disappear based on user interaction
- e.g., illustrate pop-up menu by recording clip of user pressing button, pause camera, add menu, restart camera

Be careful about taking video out of the original design setting for ethical reasons (context matters)

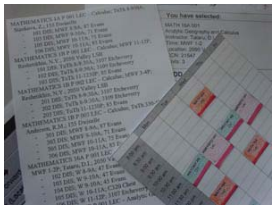
Stop Motion Example

Music Player: Nicholas Kong

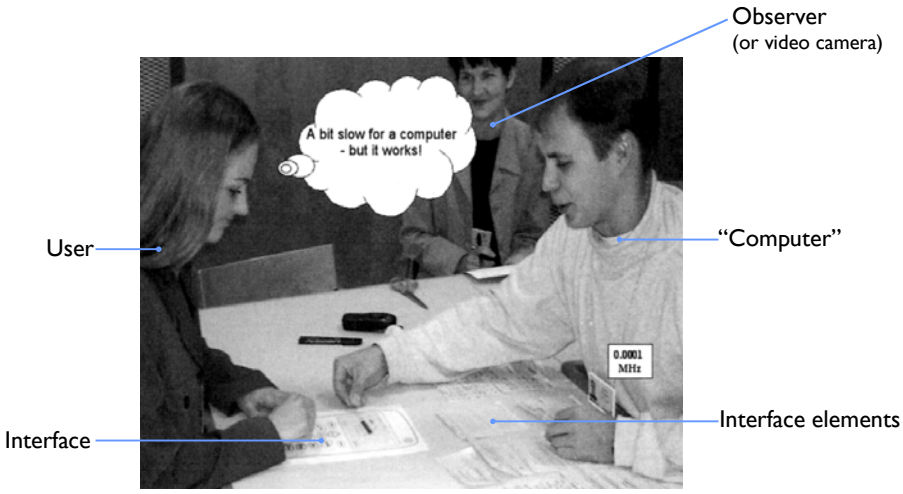
Higher Fidelity Videos

- WineM
- Cell Phone Prototypes

Wizard of Oz Prototype Testing


SIMS 213 Project: Telebears redesign



Conducting a Test

Three or Four testers (preferable)

- **Greeter** - Puts users at ease & gets data
- **Facilitator** - only team member who speaks
  - Gives instructions & encourages thoughts, opinions
- **Computer** - knows application logic & controls it
  - Always simulates the response, w/o explanation
- **Observer(s)** - Take notes & recommendations

Typical session should be approximately 1 hour

- Preparation, the test, debriefing

Conducting a Test (cont.)

Greet

- Get forms filled, assure confidentiality, etc.

Test

- Facilitator explains how test will work
  - Performs a simple task
- Facilitator hands written tasks to the user
  - Must be clear & detailed
- **Facilitator keeps getting "output" from participant**
  - "What are you thinking right now?", "Think aloud"
- **Observers record what happens**
  - Avoid strong reactions:, frowning, laughing, impatience – biases the test
- **Designers should not lead participants**
  - Let users figure things out themselves as much as possible
  - Only answer questions if user remains stuck for a long time

Conducting a Test (cont.)

Debrief

- Fill out post-evaluation questionnaire
- Ask questions about parts you saw problems on
- Gather impressions
- Give thanks

Preparing for a Test

Select your participants

- Understand background of intended users
- Use a questionnaire to get the people you need
- Don't use friends or family

Prepare scenarios that are

- Typical of the product during actual use
- Make prototype support these (small, yet broad)

Practice running the computer to avoid "bugs"

- You need every menu and dialog for the tasks
- All widgets the user might press
  - Remember "help" and "cancel" buttons

WOZ is different from pre-built "canned" functionality

Wizard of Oz Tips

Rehearse your actions

- For a complicated UI, make a flowchart which is hidden from the user
- Make list of legal words for a speech interface

Stay "in role"

- You are a computer, and have no common sense, or ability to understand spoken English.

Facilitator can remind user of the rules/think-aloud approach if the user gets stuck

Record Critical Incidents

Critical incidents are unusual or interesting events during the study.

Most of them are usability problems.

They may also be moments when the user:

- got stuck, or
- suddenly understood something
- said "that's cool" etc.

Using the Results

Update task analysis and rethink design

- Rate severity & ease of fixing problems
- Fix both severe problems & make the easy fixes

Will thinking aloud give the right answers?

- Not always
- If you ask a question, people will always give an answer, even it is has nothing to do with the facts
- Try to avoid leading questions