

## User Interface



1

## coming up

- Today
  - Quiz review
  - Lecture on User Interfaces (not on this quiz)
- give anonymous class feedback  
<https://forms.gle/A5TSF3wHu3pos5dm7>
- PEaRCE ethics activity (2 things due tomorrow)
- $\alpha$  release assignment posted: Due Oct 29

2

## Quiz Review

This is how it's going to work:

- Topics
- Then I'll open the floor to questions
- Once we are out of questions, we'll move on to the day's lecture

3

## Quiz 1 Topics

- Software development lifecycle
- Teamwork
- Requirements
- Architecture
- UML

4

## Some specifics

- UI won't be on this quiz
- Study the lecture slides
- Know how to read a class diagram
- Know your object association types
  - dependency
  - aggregation
  - composition

5

## Question types

- True / False
- Multiple choice
- Short answer
- Reasoning

6

## Let's talk about presentations

- Practice, practice, practice

7

## How to give a good presentation

- Practice with your team
- Practice with people outside your team
  - Your audience won't be our teammates who've been working on the project nonstop
- Aim your presentation at the right audience
- If you had never heard about the product, what kinds of things do you need to hear?

8

## Audience

- Who is your audience?

Your customer is your audience.

- Before you begin:
  - List the things you want to convey to your customer
  - Figure out the most effective way to convey them
  - Structure the presentation around that

**PRACTICE!**

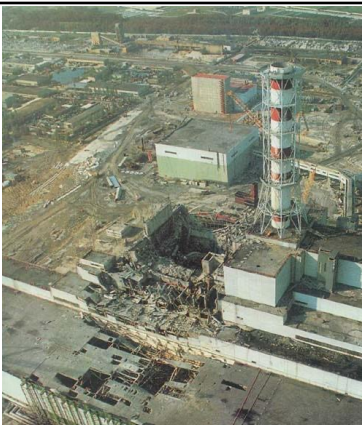
9

## User Interface



Three Mile Island

10



Chernobyl

11

## How do we avoid bad UI?

- Learn from past mistakes
- Build prototypes

12

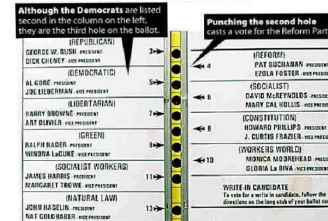
## Big questions

- What's the point of prototyping? Should I do it?
  - If so, when should I?
- Should I make my prototype on paper or digitally?
- How do I know whether my UI is good or bad?
  - What are the ways in which a UI quality can be quantified?
  - What are some examples of software you use that have an especially good/bad UI?
    - What do you think makes them good/bad?

13

## Usability and software design

- **usability**: the effectiveness of users achieving tasks
  - Human-Computer Interaction (HCI).
  - Usability and good UI design are closely related.
  - A bad UI can have serious consequences...



14

## Achieving usability

- User testing and field studies
  - having users use the product and gathering data
- Evaluations and reviews by UI experts
- Prototyping
  - Paper prototyping
  - Code prototyping
- Good UI design focuses on the *user*
  - not on the developer, not on the system environment

15

## Prototyping

- **prototyping**: Creating a scaled-down or incomplete version of a system to demonstrate or test its aspects.
- Reasons to do prototyping:
  - aids UI design
  - provides basis for testing
  - team-building
  - allows interaction with user to ensure satisfaction

16

## Some prototyping methods

1. UI builders (VSCode, Eclipse...)
  - draw a GUI visually by dragging/dropping UI controls on screen
2. implementation by hand
  - writing a quick version of your code
3. **paper prototyping**: a paper version of a UI



17

## Why do paper prototypes?

- much faster to create than code
- can change faster than code
- more visual bandwidth (can see more at once)
- more conducive to working in teams
- can be done by non-technical people
- feels less permanent or final

18

## Where does paper prototyping fit?

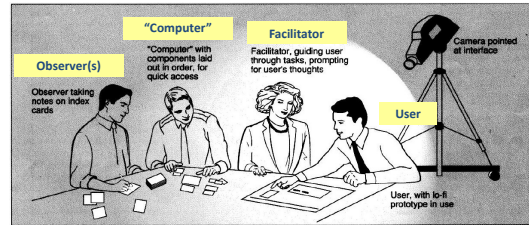
When in the software lifecycle is it most useful to do (paper) prototyping?

- Requirements are the **what** and design is the **how**. Which is paper prototyping?
- Prototyping
  - helps uncover requirements and upcoming design issues
  - during or after requirements but before design
  - shows us **what** is in the UI, but also shows us details of **how** the user can achieve goals in the UI

19

## Paper prototyping usability session

- user gets tasks to perform on a paper prototype
- observed by people and/or recorded
- a developer can "play computer"



20

## Schneiderman's 8 Golden Rules

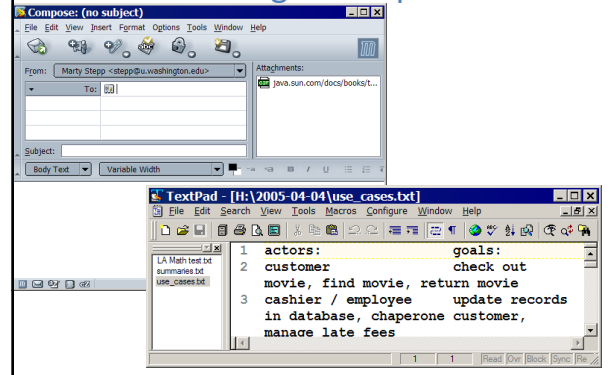
- Strive for consistency.
- Give shortcuts to the user.
- Offer informative feedback.
- Make each interaction with the user yield a result.
- Offer simple error handling.
- Permit easy undo of actions.
- Let the user be in control.
- Reduce short-term memory load on the user.



(from *Designing the User Interface*, by Ben Schneiderman of UMD, noted HCI and UI design expert)

21

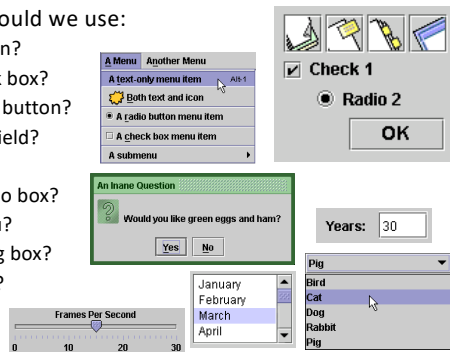
## UI design examples



22

## UI design, components

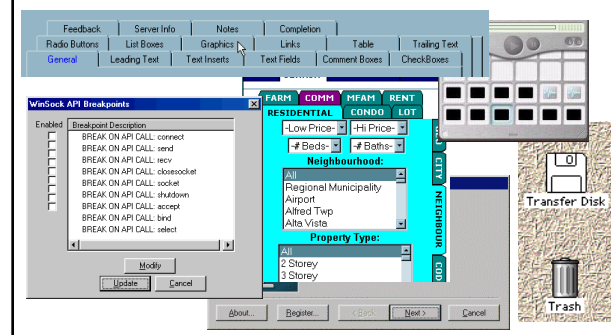
- When should we use:
  - A button?
  - A check box?
  - A radio button?
  - A text field?
  - A list?
  - A combo box?
  - A menu?
  - A dialog box?
  - Other..?



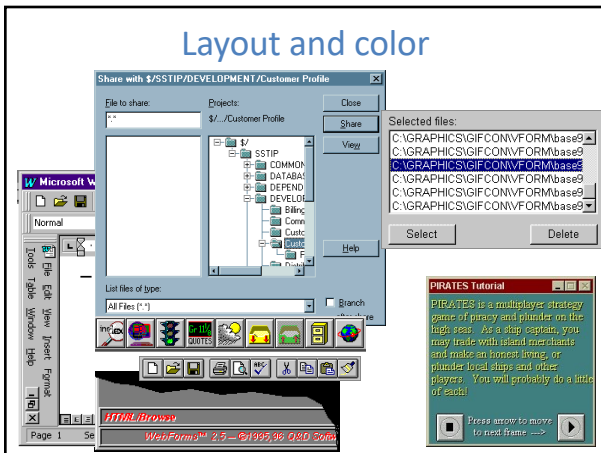
23

## UI Hall of Shame

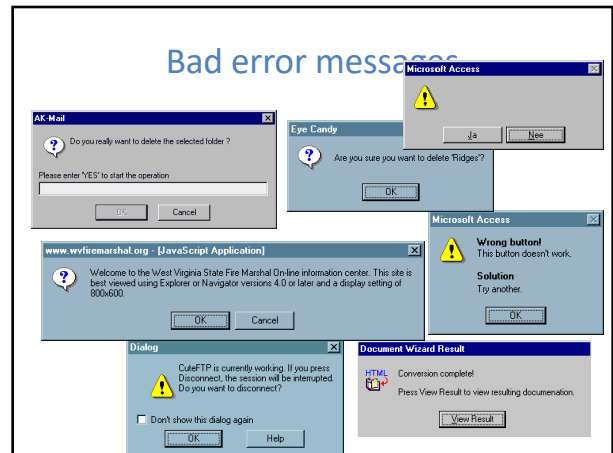
<http://interfacehallofshame.eu>



24



25



26

### UI design – buttons, menus

- Use **buttons** for single independent actions that are relevant to the current screen.
  - Try to use button text with verb phrases such as "Save" or "Cancel", not generic: "OK", "Yes", "No"
  - use **Mnemonics** or **Accelerators** (Ctrl-S)
- Use **toolbars** for common actions.
- Use **menus** for infrequent actions that may be applicable to many or all screens.
  - Users hate menus!* Try not to rely too much on menus. Provide another way to access the same functionality (toolbar, hotkey, etc.)

27

### UI design – checkboxes, radio buttons

- Use **check boxes** for independent on/off switches
- Use **radio buttons** for related choices, when only one choice can be activated at a time

28

### UI design – lists, combo boxes

- use **text fields** (usually with a label) when the user may type in anything they want
- use **lists** when there are many fixed choices (too many for radio buttons); *all* choices visible on screen at once
- use **combo boxes** when there are many fixed choices; don't take up screen real estate by showing them all at once
- use a **slider** or **spinner** for a numeric value

29

### An example UI

- Good UI dialog? Did the designer choose the right components? assume there are 20 collections and 3 ways to search

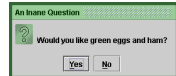
30

## UI design – multiple screens

- use a **tabbed pane** when there are many screens that the user may want to switch between at any moment



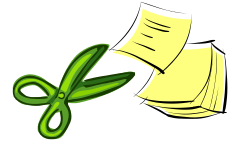
- use **dialog boxes** or **option panes** to present temporary screens or options



31

## Creating a paper prototype

- gather materials
  - paper, pencils/pens
  - tape, scissors
  - highlighters, transparencies

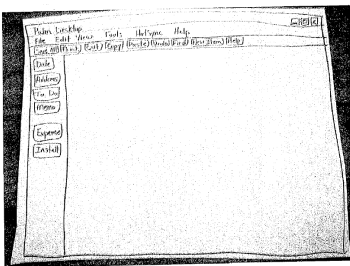


- identify the screens in your UI
  - consider use cases, inputs and outputs to user
- think about how to get from one screen to next
  - this will help choose between tabs, dialogs, etc.

32

## Application backgrounds

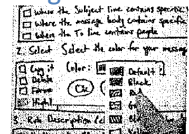
- draw the app background (parts that matter for the prototyping) on its own, then lay the various subscreens on top of it



33

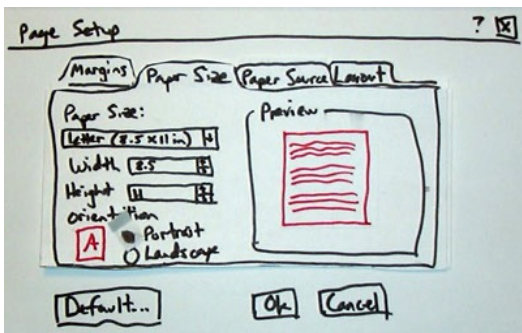
## Representing interactive widgets

- buttons / check boxes: tape
- tabs, dialog boxes: index cards
- text fields: removable tape
- combo boxes: put the choices on a separate piece of paper that pops up when they click
- selections: a highlighted piece of tape or transparency
- disabled widgets: make a gray version that can sit on top of the normal enabled version
- computer beeps: say "beep"



34

## Example paper prototype screen



35

## Prototyping exercise

- In your project groups, draw a rough prototype for a music player (e.g., VLC or iTunes).
  - Assume that the program lets you store, organize, and play songs and music videos.
  - Draw the main player UI and whatever widgets are required to do a **search for a song or video**.
  - After the prototypes are done, we'll try walking through each UI together.
- Things to think about:
  - How many clicks are needed? What controls to use?
  - Could your parents figure it out without guidance?

36

36