Today's Lecture

- More Discussion
  - Intelligent Assistance
- Properties – Graphic Interfaces
  - Fundamentals
  - Strengths / Weaknesses
- Interface Design
  - Process

Reminders
- Blog Post (Week)
- C# Programming
- Project 1

Demo – Balance Board

Intuitive Design

Microsoft Bob

On-line Avatars?
Small Group Exercise

☐ Should there be animated avatars in applications?

Split into groups of 2-4 students

Blog Post – Week 3 (Reminder)

☐ Writing assignment
  - Is time better spent on smarts (anticipating / learning) or on improving the interface?
  - Give one or two examples – pictures / etc.
  - Writeup – three to four paragraphs
☐ Receive feedback on your post
  - Worth 20 points

Direct Manipulation Systems

☐ Shneiderman 1982
  - System portrayed as extension of real world
  - Objects / actions continuously visible
    - WYSIWYG
  - Actions are rapid, incremental
  - Incremental actions are reversible
    - User fear vs. tinkering

Advantages - Graphics

☐ Symbols vs. text
☐ Faster learning
☐ Faster use / problem solving
☐ Easier remembering
☐ Natural
☐ Visual / spatial cues
  - Visual vs. verbal cue
☐ Predictable responses
☐ Can still use text

Disadvantages - Graphics

☐ Greater design complexity
  - Which control?
  - User evaluations
☐ Learning
☐ Experimentally-derived guidelines
☐ Inconsistencies
☐ Human comprehension limits
☐ Working domain = present
  - What you see is all you get
☐ Not always better
  - ATM -> Text vs. graphics

Properties

☐ Sophisticated Visuals
  - Will see especially with WPF
  - Pick/Click Interaction
☐ Restricted Set of Interfaces
  - Rarely “reinvent the wheel”
☐ Object Orientation
  - Properties / Attributes
  - Actions
  - Views
  - Recognition Memory -> Avoid out of sight, out of mind
☐ Concurrency
Design - Interface

- Software Engineering
  - Reqs
  - Design
  - Code
  - Integration
  - Acceptance
  - Waterfall model
  - Release

Waterfall vs. UI Design

- Remember, it is complex
  - Thus, \( P(\text{Wrong}) \rightarrow 1 \)
- Where to validate?
  - Users?
  - Waterfall -> Reqs, Acceptance
- UI flaws can be pervasive in the system
  - May necessitate requirements / design decisions

Iterative Design

- Design
- Evaluate
- Implement
- Marketplace != evaluation

Think Spiral Design

- Design
- Evaluate
- Implement
- Start with low fidelity (prototype)
- Work outwards w/feedback

Quick / Dirty Prototype

- Cheap / fast prototypes
  - Even better in parallel
  - Visual Basic, WYSIWYG layout wizard
- Multiple questions
  - What is possible?
  - What seems natural?
  - What questions does a new user have?
- Iterate
  - Improve / better build – richer content
  - Only show mature ones to real world

User-Centric Design

- Think users, tasks
  - What does the user need to do?
  - What is my user like?
  - Involve users as evaluators, consultants
- Evaluate, evaluate, evaluate!
  - Basic set of tasks
  - Remember x seconds -> aggregate
- Avoid
  - Ooohh, shiny!
  - Feature creep
  - Jack of all trades
Small Group Exercise

- Think useful vs. shiny
  - Coolest but least used feature for a common app you use
  - Most useful aspect but least “wow” factor for a common app you use

Split into groups of 2-4 students

Who is the user?

- Target population
  - Age, gender, ethnicity
  - Education
  - Physical abilities
  - Compute experience
  - Domain experience
  - Application experience
  - Social context
  - Classes of users

Try not to over-generalize
Divide / conquer into classes

User Analysis

- Techniques
  - Questionnaires
  - Interviews
  - Observation

- Obstacles
  - Isolation – devs vs. users
  - Tech support
  - Marketing shields
  - User expense
  - Doctors, lawyers

Example: Self-Service Grocery

- Users
  - Grocery shoppers
  - Wide range of ages (10-80) and physical abilities
  - No computer experience
  - No training
  - Knowledge of food but not inventory

- Major user classes
  - Family shopping – person with kids present
  - Store clerks helping shoppers

Questions?

- Homework 2 (Today)
- Week 3 – Blog
- Project 1