Lecture Objectives

- At the completion of this lecture you should be able to:
  - Understand the importance of interface design
  - Understand the basic principles of good interface design
  - Explore the diversity of interaction devices available
  - Understand the difference between web design and traditional GUI design

Why bother with Interface Design?

- Computer Systems are becoming increasingly powerful but complex
- Graphical User Interfaces are becoming something of a standard
- Wide range of interaction devices, not just computers... PDAs and Handhelds etc
- The Web as a medium is becoming increasingly important
- 63% of large software projects go over cost (Greenberg, 2001)
  Managers gave four usability-related reasons
    - users requested changes, overlooked tasks, users did not understand their own requirements, insufficient user-developer communication and understanding
- Well designed Interfaces can
  - Reduce human error, training costs and customer support costs and increase productivity

Basic Principles of Interface Design

1. Focus on the users not the technology
2. Consider Function first, presentation later
3. Support “Transportability of Knowledge”
4. Be Consistent
5. Conform to the Users view of the task
6. Make the User Boss
7. Promote Learning & Provide Feedback
8. Deliver Information, not just data
9. Design for Errors
10. Colour Aesthetics

1. Focus on the users not the technology

- Principle Numero Uno!!!
- Ask yourself the following
  - Who are the intended users?
  - For whom is this product being designed. What activity is this supposed to support?
  - What value will it provide?
  - What skills do these intended users need?
  - What problems might they encounter?
- Expensive proposition but the cost of not answering these questions is huuuuge!!!
Topic 5c: Designing Interfaces and Dialogues – Part 1

2. Consider function first, presentation later

- Many developers immediately begin worrying how the computer based product or service will look
- You can end up with a great looking product that lacks important functionality
- Be careful of different perspectives

3. Support Transportability of Knowledge

- Make use of what the user already knows
  - eg File>Exit quits Windows applications
- Answers lots of design questions for you
  - What are menus? - things that drop down from the top
  - What’s the first menu option? - File
  - What’s the last menu option? - Help
  - What does the word exit mean? - quit the application.

Axiom: Follow Microsoft’s lead

- Most users already know how to drive Windows applications.
- Don’t scramble the positions of the brake, accelerator and clutch pedals!
- Grit your teeth at Microsoft’s design errors and reproduce them

4. Be Consistent

- Always do the same thing in the same way
  - it’s always Exit
  - Exit only means “Leave this application”
  - Exit never means “Go back to the previous screen.” or “Go back to the splash screen.”
- The (Drop-Down) Menus are always the same.
  - If menu options are inappropriate in particular locations, dim them.
- The words you use to refer to objects are the same across screens.

Transportability and Consistency work together

- Any exception means:
  - The user always has to stop and think:
    - “Is this the exception?”
    - “Do I have to think twice about this one?”
  - So any breakdown in consistency or transportability means:
    - The user is slowed down
    - The user is frustrated
    - The user makes more mistakes
Consistency is harder than you think

- You have to remember what you did 3 days ago
- You have to keep track of what your team does
  - company/project standards are useful
- You may have to revamp your system because
  - You better understand what the user wants
  - You better understand how the system works
  - You better understand the task
- Start with the assumption you’ll probably get the interface wrong at the beginning.
- Budget time to go back over your system to make it consistent

Consistency in Appearance and Performance

- These aren’t the same.
- You can see inconsistency in appearance but not in performance
- How (and when) does your system respond to an input error?
- When do you prompt the users?
- How does the user access the Help system? (i.e., does \[ F1 \] always work?)
- How must the user enter dates, percentages, etc.?
- Where do forms appear on the screen?

5. Conform to the users view of the task ... Don’t complicate it

- Don’t make users commit unnatural acts.
- Imposing arbitrary restrictions.
- Use the users vocab not your own.
- The Power-Complexity Trade off

"What do you mean Rumpelstiltskin is too long for a password"

6. Who’s Boss Around Here?

- Control is passed to the user
- Within reason they can:
  - Do what they want in the order they want
  - Enter partial and incomplete data
  - Stop in the middle of a task
  - Leave blank and incorrect data on the screen for a while
- The computer program is passive
- The user is active
- The user becomes the boss
**The Power – Complexity Trade-off**

- This approach makes programming harder (and programmers insecure!)
- It makes program use much easier.
- Given that the user is always the boss, lots of tasks are easier:
  - We know never to beep in righteous indignation
  - We know that error messages must be polite and respectful
  - We know never to use those ‘gizmos’ that edit input on the fly
  - We know we must use tools (e.g., drop down list boxes) so the user can’t make an error.
- User Errors are usually Programmer Errors

**7. Feedback lets the User know what’s going on**

- Examples
  - A button goes down when you click it
  - The mouse pointer turns into an hourglass while short tasks run
  - There’s a progress bar for long tasks
- Objects always respond to actions on them. Something always shows you that it’s worked
- Negative Example
  - Fill in a form, click on the **Save** button
  - Nothing happens……The user is left in doubt.
  - Feedback suggestions anyone ???

**Provide feedback without pestering the User**

- A pop-up a message box saying ‘the data has been saved’?
  - Probably not - why not just clear the data entry areas on the form?
- Good feedback
  - Isn’t intrusive
  - Doesn’t require action on the part of the user
  - Doesn’t keep the user from ‘learning by exploration’
- Bad feedback takes the user’s mind off what they’re doing and makes them think about the system - always a bad thing

**Really useful stuff !!!**

- **Copy Profile Error**
  - The operation completed successfully.
  - **OK**

**Learning by Exploration**

- Users like to learn this way
- Let them - don’t be too fussy about what they can do, or the order they can do it in.
- Provide feedback when
  - They are about to fall off a cliff .. there aren’t many cliffs:
    - Deleting the corporate database
    - Killing the system/document/process they are working on
  - They ask for help:
    - Context sensitive ([F1]) help is good feedback if well done.
    - Bad help systems won’t be used.

**8. Deliver Information ... not just data**

- Visual order and focus
- Match the medium
  - Expect to be designing for platforms used by small handheld devices, mobile phones etc.
  - Web – explosion, what impact for designers do these rules still apply?
- Attention to detail
9. Prevent Errors

Design it so it’s hard to make mistakes!!
• Menu selection vs. form fill-in
• No alphabetic characters where numbers expected
• Check before proceeding with major actions (eg save before exit prompt)
• Feedback for errors including simple specific instructions for recovery

10. Colour Aesthetics

• Colour perception is personal
• What looks good to you won’t necessarily look good to many others
  – What looks garish and ‘yuckie’ to you, may look wonderful to others
• So... Drab Is Best
  – works for traditional systems although this rule goes out the window for Websites and web-based systems
• Grit your teeth, curse your high school teachers, and build drab systems with no pictures (except at the start) and colours like the Microsoft defaults.

Difference between web design and traditional GUI design

• Designer gives up full control
• Device diversity
• The User controls navigation
• Part of a whole

References

Material for this lecture prepared by Mr Dan Eaves and Mr Ryan Fernandes


References

• Don Norman’s *The Design of Everyday Things* (Doubleday Currency 1986)
  – Very useful and general discussions of the problems of designing interfaces applied to doors, windows, water taps and computer systems. A wonderful book.

• Alan Cooper’s *About Face: The Essentials of User Interface Design* (Programmer’s Press, 1995)
  – Practical advice combined with criticism of a number of standard aspects of interface design. Amusing, exciting, useful and frustrating.
  – The rules for designing consistent interfaces. Specific to Windows 95, so of no use at Monash but useful at home.