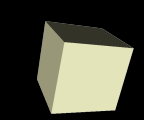
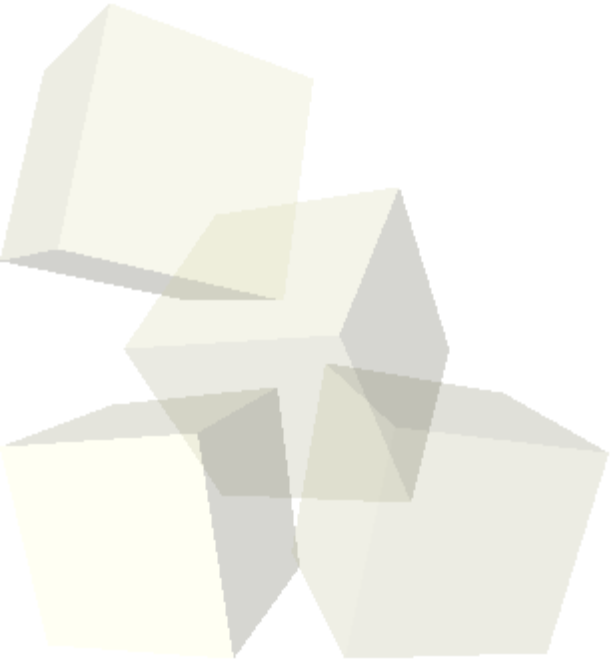


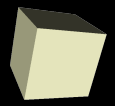
# Eliminating Errors

**Pat Wilbur**

November 2, 2007

CS559 – Fall 2007





# Overview

- What types of errors occur?
- Who is responsible for causing errors?
  Handling errors?
- Why are error messages problematic?
- Who is actually the smartest?
  The user, the program, or the programmer?
- How can errors be eliminated?

# #%*$!!!  Hey, Errors Happen...

- The possibility for some errors is unavoidable
- Errors occur for a variety of reasons:
  - Program independently enters an unexpected state
    - ➔ *Segmentation fault?*
  - Program receives unexpected input from user
    - ➔ *#%*$ as a marital status?*
  - Program encounters unexpected hardware errors
    - ➔ *OIT's AD server crashed in the middle of saving thesis?*
- Can errors be reduced?

# The Customer Is Always Right

# The Customer Is Always Right

- Three possible culprits for error:
  - The **Human**: *user* is guilty
  - The **Computer**: *Intel* is guilty
  - The **Interaction**: *program(mer)* is guilty

# The Customer Is Always Right

- **Recall:** Errors occur for a variety of reasons:
  - Program independently enters an unexpected state
    - → *Segmentation fault?*
  - Program receives unexpected input from user
    - → *#%*$ as a marital status?*
  - Program encounters unexpected hardware errors
    - → *OIT's AD server crashed in the middle of saving thesis?*
- Who encounters the error? Who is responsible for how it is handled?

# The Customer Is Always Right

- **Recall:** Errors occur for a variety of reasons:
  - **Program** independently enters an unexpected state
    - *Segmentation fault?*
  - **Program** receives unexpected input from user
    - *#%*$ as a marital status?*
  - **Program** encounters unexpected hardware errors
    - *OIT's AD server crashed in the middle of saving thesis?*

- Who encounters the error? Who is responsible for how it is handled?

  **The Program(mer)**

# The Customer Is Always Right

- **Recall:** Errors occur for a variety of reasons:
  - **Program** independently enters an unexpected state
    - *Segmentation fault?*
  - **Program** receives unexpected input from user
    - *#%*$ as a marital status?*
  - **Program** encounters unexpected hardware errors
    - *OIT's AD server crashed in the middle of saving thesis?*

- Who encounters the error? Who is responsible for how it is handled?

  **The Program(mer)**

- Programs are always responsible for errors. Users are just sometimes fooled by bad software.

# The Customer Is Always Right

- Programs should adapt to users
- Programs should clearly inform users on how input should be formatted BEFORE entry
- Programs should be flexible with input they accept
- Programs should be flexible with their dependence on hardware

# Error: You are an idiot! Go away!


- Error messages can be offensive:
  - Interrupt the process that is executing
  - Make users feel incapable, ignorant
  - Negative reinforcement is almost always ineffective *(only makes users angry at software)*
- Waste time—require corrected/repeated actions
- **Are simply ignored!**

# Error: You are an idiot! Go away!

- **Cooper:** *“Humans have emotions and feelings: Computers don't. ... [Humans] get angry when they are flatly told they are idiots.”*
- Reliable software should **not** provide many dialogs that notify a user when and how they are mistaken
- Rather, reliable software should be flexible and adapt to the user

# Programmer > Program > User?

- Users are offended by errors because they are in many ways smarter than the program & computer
  - *“The program nags me when it makes mistakes!”*
- Programmers are offended by the idea of extensive error handling and flexibility because they feel they are smarter than error-prone users
  - *“Why should I work very hard on having my program cater to these people? I'm no imbecile like them, I certainly know what I'm doing!”*

# Programmer > Program > User?

- **Key:** Realize that admitting the user is smarter than the program does not mean the user is smarter than the programmer

- The following rank of smartness (greatest to least) is completely possible:
  - **Programmer**
  - **User**
  - **Program**

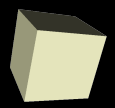
- The customer is always right (at least always more right than your program)—it should treat them that way!

# Eliminating Errors

- “Eliminating errors” does not mean ignoring them and not displaying error messages
- Eliminating errors is more prevention than reaction
- Software should be designed such that the human is more important than the software
  - If the input came from the user, it should be assumed to somehow contain what the user desires
- Software should work harder to understand input

# Eliminating Errors

- If a program knows that input must be entered a certain way, prevent the user from entering it any other way
- Use appropriate input fields (combo boxes, radio buttons, and check boxes, where appropriate)
- Make all unavoidable error messages **polite, illuminating,** and **helpful**
- Anticipate problems and handle as much as possible for the user's sake



# Summary

- What types of errors occur?
- Who is responsible for causing errors?
  Handling errors?
- Why are error messages problematic?
- Who is actually the smartest?
  The user, the program, or the programmer?
- How can errors be eliminated?


The page at http://www.designerwiz.com s
Thank you. Any questions?
OK


The page at http://www.designerwiz.com says:
ERROR: Did not receive expected: {applause}. Exiting.
OK