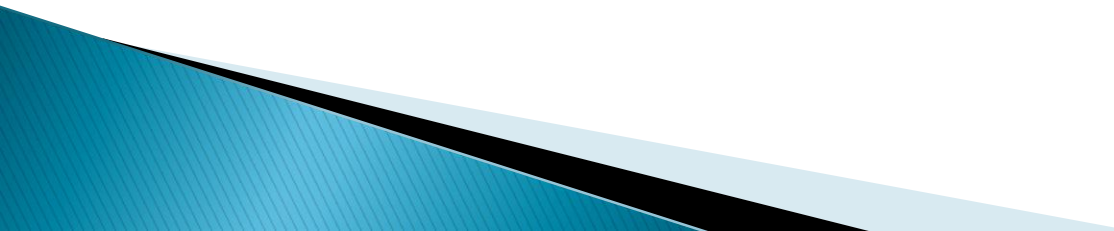


Testing and Evaluation

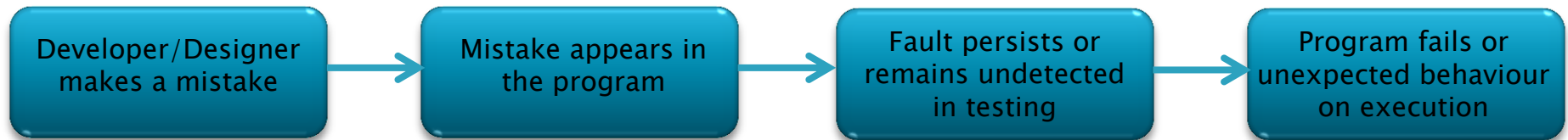
Dr Brendan Cassidy
CO1111 Computing Skills

What are we doing today?

- ▶ Looking at Software testing & types of testing
 - Verification & Validation
 - Static Testing
 - Dynamic Testing
 - Black Box & White Box Testing
 - ▶ Looking at User Evaluation & types of evaluation
 - User Participation
 - Expert Analysis
- 

Why Test Software

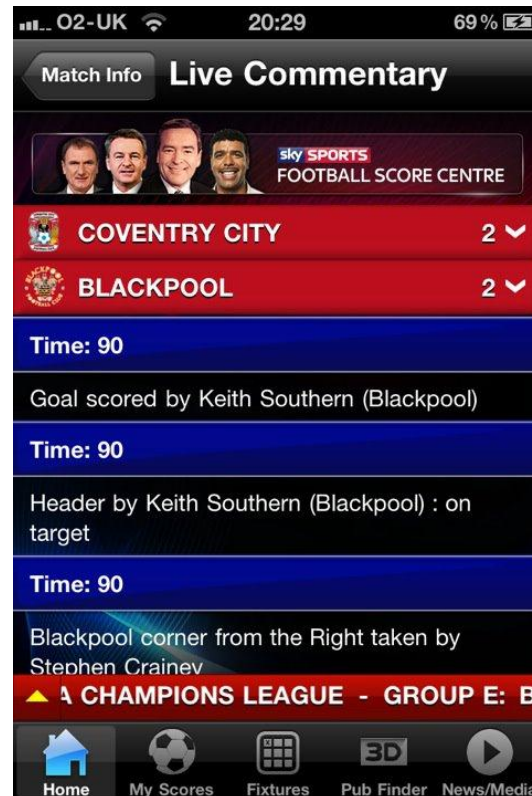
- ▶ Whether you are a novice or an expert all designers/developers make mistakes
 - Nobody is perfect



- Minor/Inconvenient Bugs
- Major Bugs
- Disasters
- Total Annihilation

Minor Bugs

- ▶ Likely to cause annoyance
 - We see many of these in everyday life



Major Bugs

- ▶ Likely to cause major disturbance or loss of revenue

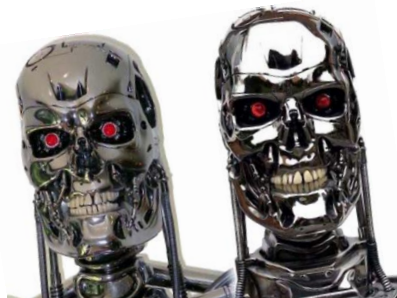


Disasters

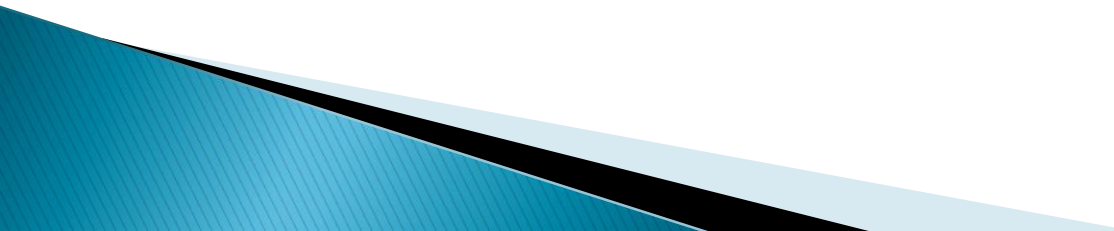
- ▶ In 1985 Canada's Therac-25 radiation therapy machine malfunctioned and delivered lethal radiation doses to patients
- ▶ Due to a bug, a technician could accidentally configure Therac-25 so the electron beam would fire in high-power mode without the proper patient shielding.

TOTAL ANNIHILATION!

- ▶ In 1983 the Soviet early missile warning system detected the Americans had launched a nuclear attack of 5 ballistic missiles
 - The soviet duty officer had a “funny feeling in his gut” that if the Americans were to attack they would launch more than 5 missiles
- ▶ Caused by a bug in the soviet software that failed to discard false missile claims caused by sunlight reflecting off clouds



Testing & Requirements

- ▶ Testing is about making sure we meet the correct requirements of a piece of software
 - ▶ A Requirement is a single documented need of what a product should do or be
 - Requirements gathering can be a complicated process
 - ▶ You are given a set of requirements at the start of every week in your 4 week challenge
- 

Verification & Validation

- ▶ Verification: Are we building the product right?
 - Does the product meet its requirements?
 - This consists of tests designed to expose problems
 - We are trying to uncover unusual/undesirable behaviour so we can fix it
 - Success in this case is where a test fails
 - Tests are often based on unrealistic or extreme scenarios
- ▶ How could you verify your quiz apps?

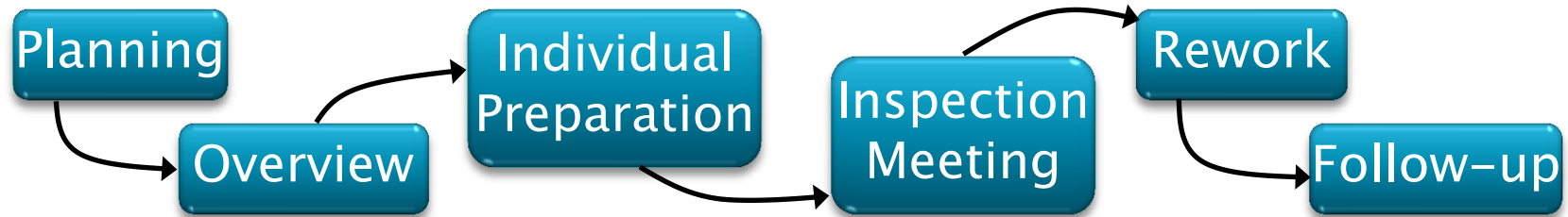
Validation & Validation

- ▶ Validation: Are we building the right product?
 - Does the product meet the customers needs
 - Have we gathered the correct set of requirements?
 - These are tests to convince the client that the product works
 - Success in this case is where a system performs correctly
- ▶ We have validated your quiz app

Software Inspections

- ▶ Software inspections are a special kind of review focussed on defect detection (primarily verification)
 - Logical Errors
 - Code Anomalies
 - Standards non-compliance
- ▶ A team of people with different perspectives on the software inspect
 - Requirements
 - The code looking for errors
 - Design diagrams
- ▶ Then raise any issues at a meeting to be fixed by the programming team
- ▶ Do not require the an actual implementation to be executed
 - Static Verification & Validation

The Inspection Process



- ▶ At the Overview stage, the author presents the code and explains what it does
- ▶ Team members then study the code and specification individually for a period of time to identify defects
- ▶ Possible defects are then discussed at the inspection meeting itself, which should last no more than 2 hours
 - A list of common errors can be used to focus the discussion
- ▶ The author should correct the defects after the inspection
 - The moderator decides whether a follow-up inspection is needed

Software Testing

- ▶ Involves executing an implementation of the software with test data and examining its behaviour
- ▶ Requires an implementation of the system to be executed
 - Dynamic Verification & Validation
- ▶ Two main testing methods
 - Defect Testing
 - Searching for problems (bugs)
 - Statistical Testing
 - Assessing performance

The First Ever Computer Bug

- Found and de-bugged in 1947 at Harvard University on a Mark II Aiken Relay Calculator

9/9

0800 Amdam started

1000 " stopped - amdam ✓


1300 (032) MP-MC { 1.2700 9.037847025
~~1.982642000~~ 9.037846995 correct
~~2.130476415~~ 4.615925059(-2)
 (033) PRO 2 2.130476415
 correct 2.130676415

Relays 6-2 in 033 failed special speed test
 in relay .. 10.000 test.

Relays changed

1100 Started Cosine Tape (Sine check)

1525 Started Multi-Adder Test.

1545  Relay #70 Panel F
 (moth) in relay.

First actual case of bug being found.

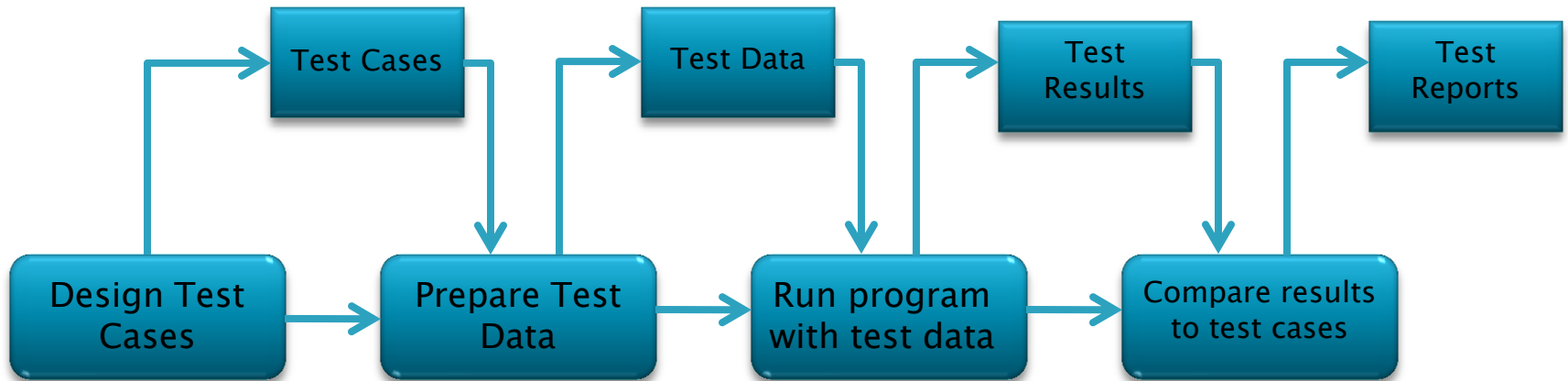
1630/1630 Amdam started.

1700 closed down.

Relay 2145
 Relay 2371

Defect Testing

► The defect testing process



Black Box Testing

- ▶ Testing without knowledge of the internal working of a computer program

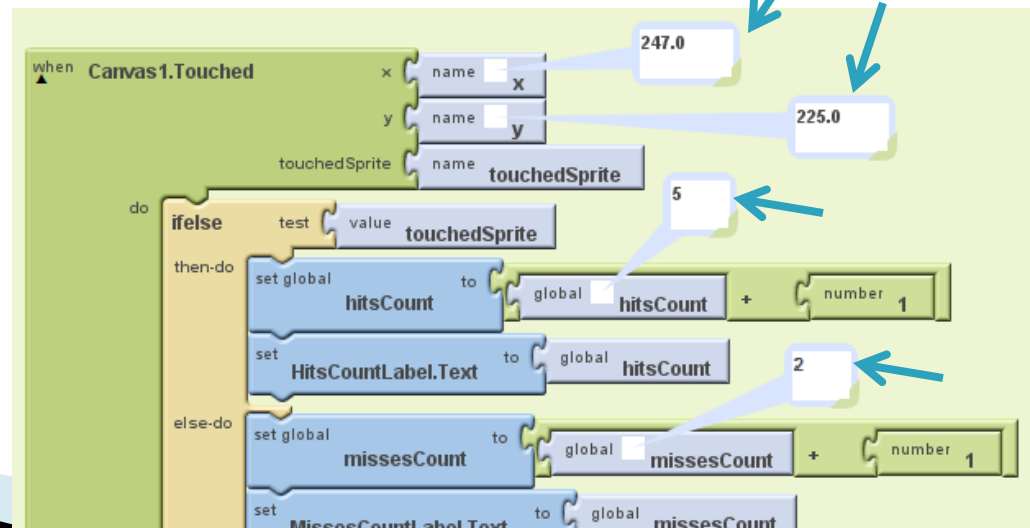


White Box Testing

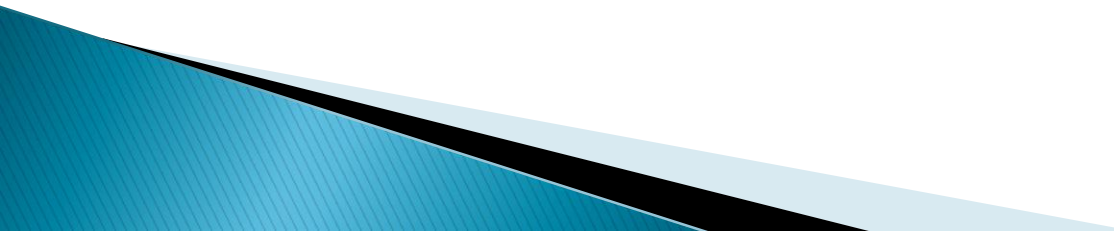
- ▶ Also called
 - Glass Box Testing
 - Clear Box Testing
 - Structural Testing
- ▶ Have any of you used white box testing already when debugging your apps?

Tests derived from knowledge of the software's structure and implementation

Usually applied to small program units, such as procedures or events



User Evaluation

- ▶ V&V is limited to testing the software and checking if it is acceptable to the client
 - ▶ Inspections cannot demonstrate if a system is operationally useful
 - ▶ Evaluation has 3 main goals
 - Assess the extent and accessibility of a systems functionality
 - To assess a users experience of the interaction
 - To identify any specific problems with the system
- 

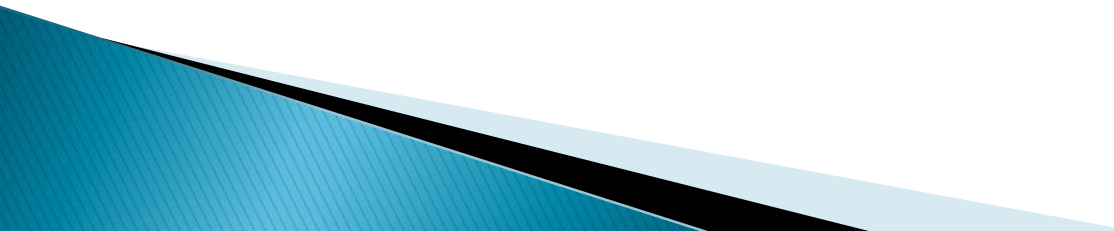
Types of Evaluation

- ▶ Evaluation through user participation
 - User Testing
 - Think Aloud
- ▶ Evaluation through expert analysis
 - Cognitive walkthrough
 - Heuristic Evaluation

Data Gathering with user testing

- ▶ Quantitative
 - Usually numeric
 - Can be analysed using statistical techniques
 - E.g. Task Completion times
- ▶ Qualitative
 - Non numeric
 - More difficult to analyse (subjective)
 - E.g. User Satisfaction
- ▶ We will focus on Qualitative
 - So if you wish you can have a go for your show and tell on Friday
 - You can use each other as participants
 - you are the experts!

Think Aloud

- ▶ Evaluators watch a user complete set tasks
 - Observation is not enough to understand how well a system meets a users requirements
 - Users are asked to elaborate on their actions by 'thinking aloud'
 - ▶ Describe what they think is happening
 - ▶ Why they take an action
 - ▶ What they are trying to do
- 

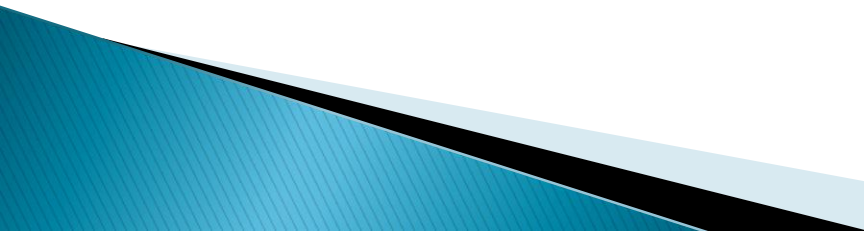
Think Aloud

- ▶ Simple
- ▶ Requires little expertise
- ▶ Good with interface evaluation
- ▶ Can be used throughout design process
- ▶ Can be used to observe how the system is actually used
- ▶ Data is subjective and may be selective
- ▶ Observation can alter the way people perform tasks
- ▶ Describing what you are doing can change the way that you are doing it


Advantages

Disadvantages

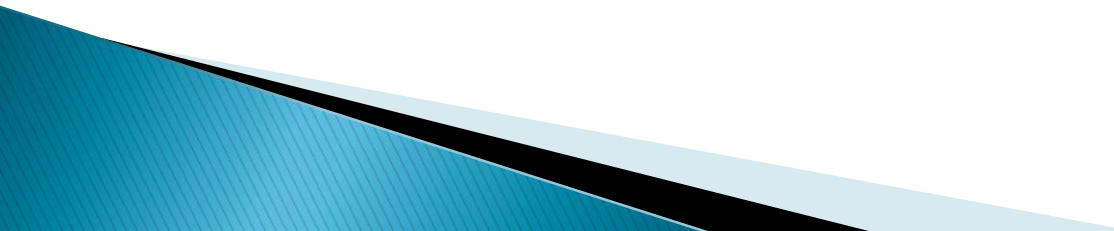
Cognitive Walkthrough

- ▶ Type of expert analysis
 - ▶ Cognitive walkthrough originated from software inspections
 - ▶ Instead of stepping through code we step through the actions an interface requires a user to perform to complete a task
 - ▶ We then step through the action sequence and check it for potential usability problems
 - ▶ Main focus is to establish how easy the system is to learn
 - Supporting learning through exploration
- 

You will need...

- ▶ A description of the prototype of the system
 - ▶ A description of the task the user is to perform on the system
 - ▶ A complete list of actions needed to complete the task and system responses
 - ▶ An indication of who the users are and how much experience they have
- 

Ask the following questions

- ▶ Will the users be trying to produce whatever effect the action has?
 - ▶ Will the users be able to notice that the correct action is available?
 - ▶ Once users find the correct action at the interface, will they know that it is the right one for the effect they are trying to produce?
 - ▶ After the action is taken, will users understand the feedback they get?
- 

TV Remote Example

Action A: Press the timed record button

Response A: Display moves to timer mode, flashing cursor appears after 'Start'

Action B: Press Digits 1800

Response B: Each digit is displayed as typed and flashing cursor moves to the next position

Action C: Press the 'timed record' button

Response C: Flashing cursor moves to 'end'

Action D: Press digits 1915

Response D: Each digit is displayed as typed and flashing cursor moves to the next position

Action E: Press the 'timed record' button

Response E: Flashing cursor moves 'channel'

Action F: Press Digit 4

Response F: Digit is displayed as typed and flashing cursor moves to the next position

Action G: Press the 'timed record' button

Response G: Flashing cursor moves to 'date'

Action H: Press digits 290911

Response H: Each digit is displayed as typed and flashing cursor moves to the next position

Action I: Press the 'timed record' button

Response I: Stream number in top right hand corner of display flashes

Action J: Press the transmit button

Response J: Details are transmitted to video player and display returns to normal mode

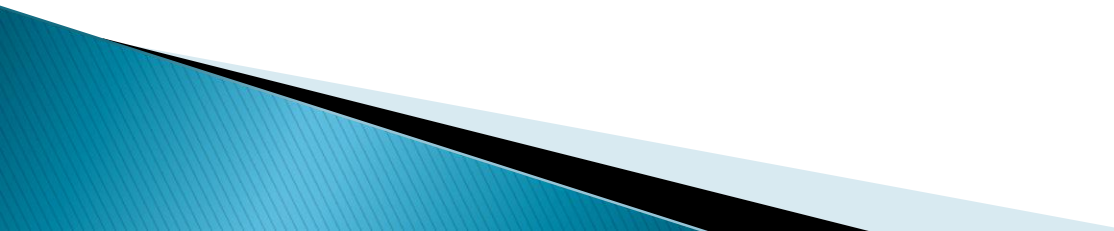


TV Remote Example

- ▶ **Action A:** Press the timed record button
- ▶ **Question 1:** Will the users be trying to produce whatever effect the action has?
 - The interface provides no indication that the user needs to press the timed record. However, it's reasonable to assume that a user familiar with video recorders would know that this was required
- ▶ **Question 2:** Will the users be able to notice that the correct action is available?
 - The 'timed record button' is visible on the remote control
- ▶ **Question 3:** Once users find the correct action at the interface, will they know that it is the right one for the effect they are trying to produce?
 - It is not clear which button is the 'timed record button' the icon of a clock (third button down on the right) is a possible candidate but this could be interpreted as a button to change the time. Other candidates may be the fourth button down on the left.
- ▶ **Question 4:** After the action is taken, will users understand the feedback they get?
 - Once the action is taken the display changes to the timed record mode and shows familiar headings. It's reasonable to assume the user would recognise this as successful completion of the first action



Heuristics

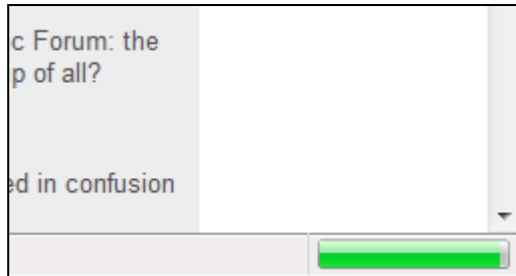
- ▶ Heuristics are a set of rules for good practice
 - ▶ Can be used as design guidelines or for evaluation
 - ▶ Different heuristic sets for different types of applications
 - Game Heuristics
 - Mobile Heuristics
 - Tangible Heuristics
 - Heuristics for Children
- 

Nielsen's 10 Heuristics

- ▶ Visibility of system status
- ▶ Match between system and the real world
- ▶ User control and freedom
- ▶ Consistency and standards
- ▶ Error prevention
- ▶ Recognition rather than recall
- ▶ Flexibility and efficiency of use
- ▶ Aesthetic and minimalist design
- ▶ Help users recognize, diagnose, and recover from errors
- ▶ Help and documentation

Heuristic 1

- ▶ Visibility of System Status
 - The system should always keep users informed about what is going on, through appropriate feedback within a reasonable time.
 - E.g. Progress bar rather than an hour glass



Heuristic 2

- ▶ Match between system and the real world
 - The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
 - Get to know the user
 - How will they categorise information

Heuristic 3

- ▶ User control and freedom
 - Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
 - Where am i
 - How did i get here
 - How can i get back



Heuristic 4

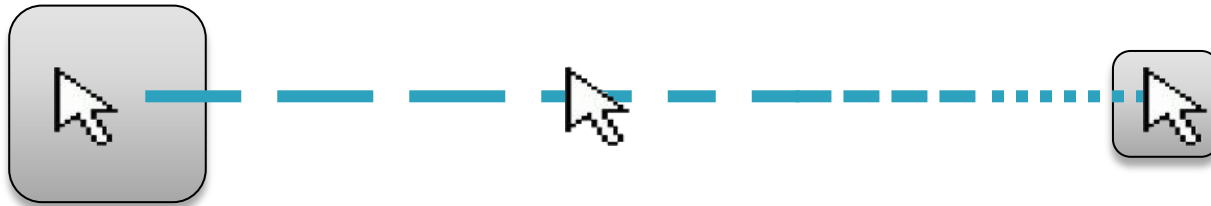
- ▶ Consistency and standards
 - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
 - Inconsistency increases the learning curve

Heuristic 5

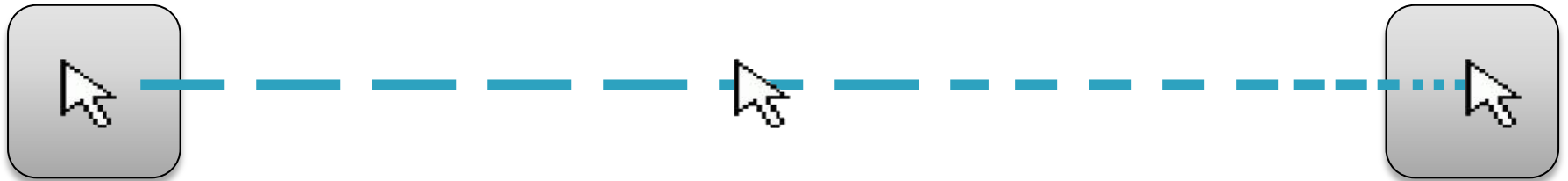
- ▶ Error prevention
 - Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Fitts Law

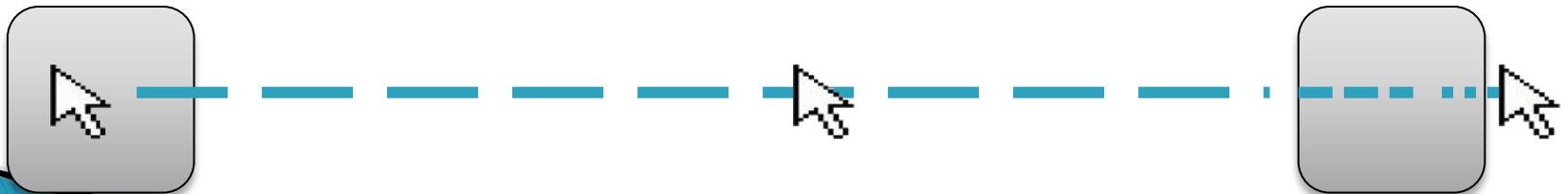
- ▶ The smaller the target the longer it takes to reach



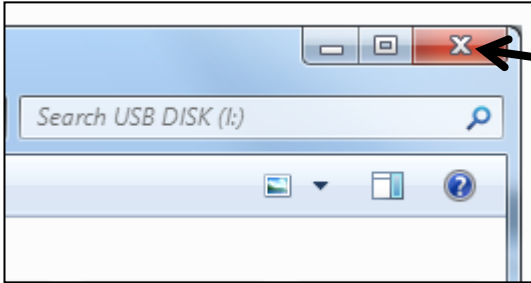
- ▶ The further away the target the longer it takes to reach



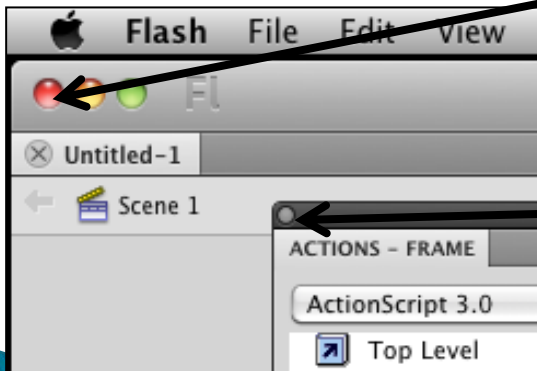
- ▶ The faster we go the more difficult it becomes to reach (errors)



Examples



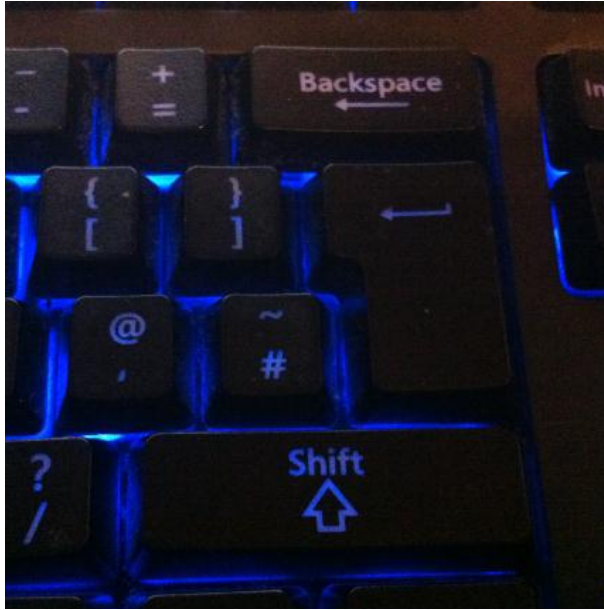
Windows 7 close window button has been made wider to allow users to move to it quicker



Mac OS close window button is smaller and takes noticeably longer to close windows

This is the button to close the Actions window in Flash on the Mac

Fitts Law Example



I'm a PC



I'm a Mac

There is more chance of error when using a Mac keyboard such as this compared to a standard PC keyboard.

Errors

► Clear Error Messages

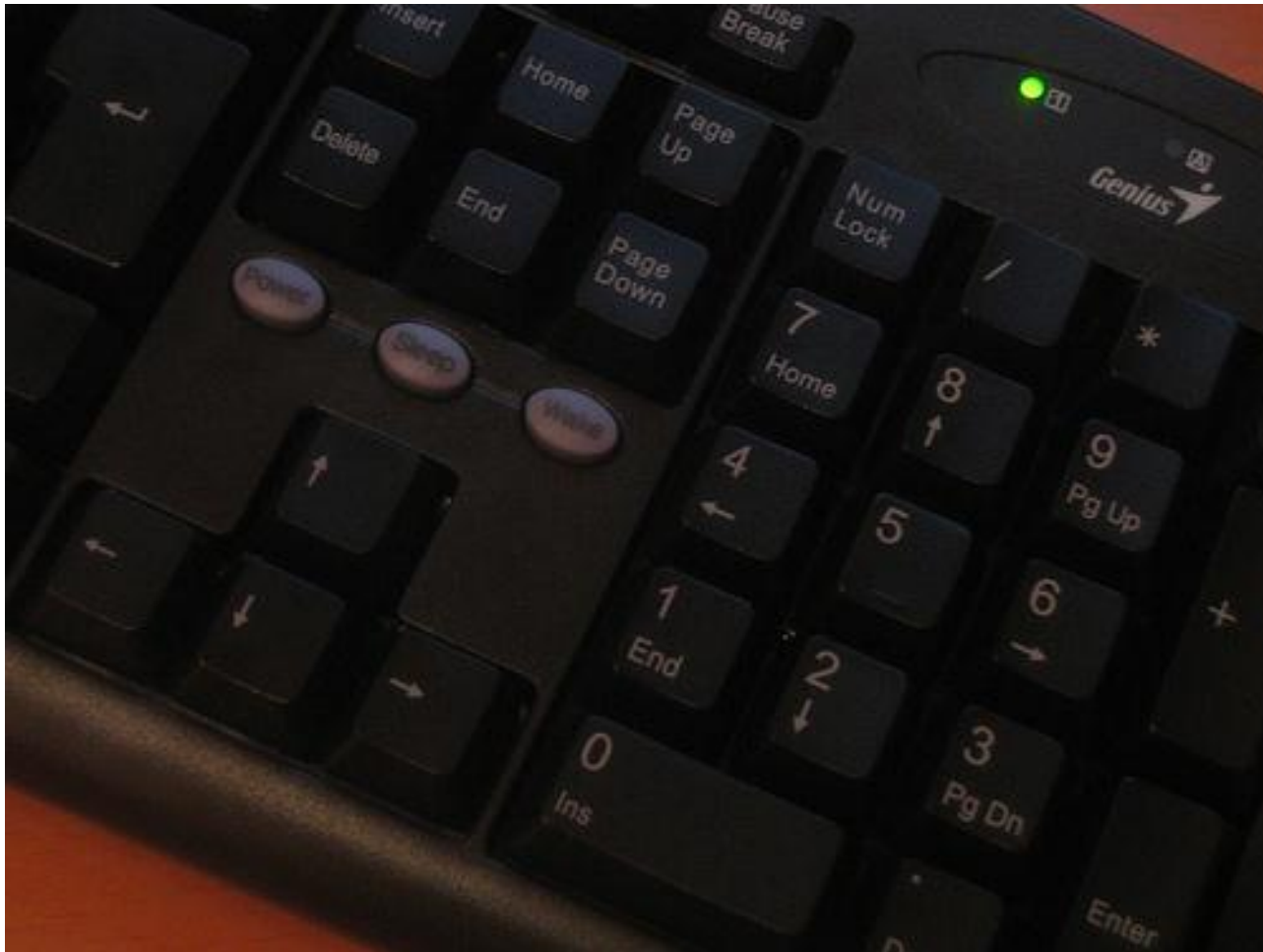


- Where possible prevent errors
 - Anticipate
 - User testing

Is this a good idea?

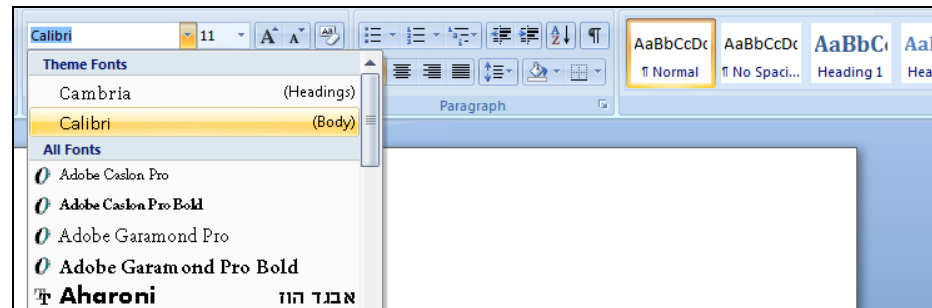


Or This???



Heuristic 6

- ▶ Recognition rather than recall
 - Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.



Heuristic 7

- ▶ Flexibility and efficiency of use
 - Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
 - Look for features that serve more competent users
 - Shortcut keys
 - Ctrl + Enter in flash tests your movie
 - F9 brings up the actions window

Heuristic 8

- ▶ Aesthetic and minimalist design
 - Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
 - Consider what each element of the site adds to the site
 - Is it needed?
 - E.g. A clock on a website?

Heuristic 9

- ▶ Help users recognize, diagnose, and recover from errors
 - Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.



Heuristic 10

► Help and documentation

- Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

- Clear
- Concise
- Accessible



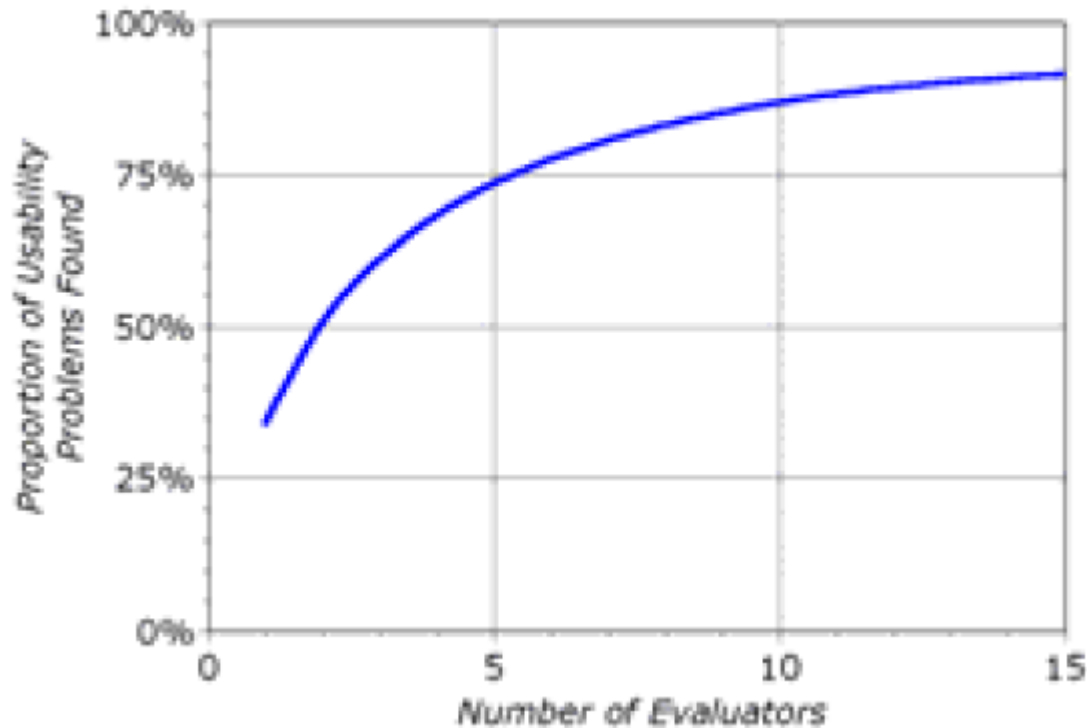
Heuristic Evaluation Process

▶ Multiple Evaluators

- Provided with heuristics to use
- Can be given tasks to perform or left to explore
 - Depends on expertise etc.
- Usually 2 passes
 - One to learn interface
 - One to identify problems
- Problems should be recorded
 - In relation to a heuristic
 - In detail

How Many?

- ▶ According to Nielsen, 3–5 is usually enough



(Jakob Nielsen & Landauer, 1993)

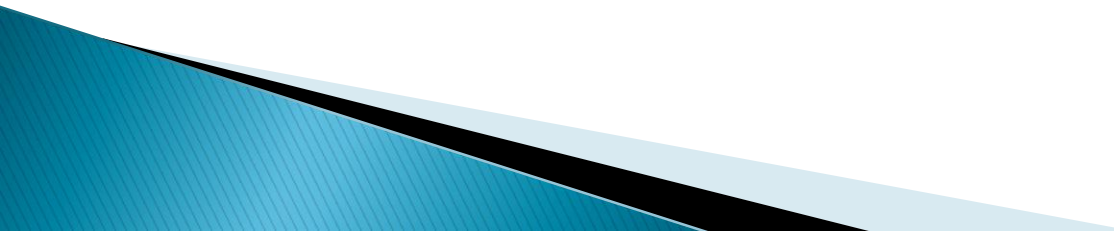
The Process

- ▶ Once a heuristic set has been chosen or assembled it is given to the evaluators
- ▶ Evaluators will familiarise themselves with the heuristics
- ▶ The evaluators will familiarise themselves with the program (game)
 - First pass
- ▶ The evaluators will then use the program again trying to find violations of the heuristics
 - Second pass

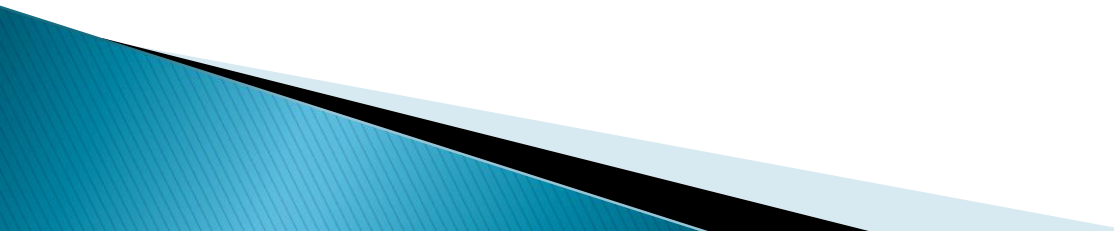
Identifying problems

- ▶ Evaluators use the program with a critical mind
- ▶ Even small problems can be noted
- ▶ We are not really looking for compliance with heuristics at this stage
 - Looking for *problems*
- ▶ When a problem is found it is noted in an evaluation pro-forma
 - A structured way to report problems formally
 - A form/report sheet
- ▶ Each problem is attached a severity rating

Example Severity Ratings

- ▶ Once a problem is identified a severity rating is attached
 - ▶ 0= I don't think that this is a usability problem
 - ▶ 1= **Cosmetic problem only:** need not be fixed unless extra time is available on the project
 - ▶ 2= **Minor usability problem:** fixing this should be given low priority
 - ▶ 3= **Major usability problem:** important to fix, so should be given high priority
 - ▶ 4= **Usability catastrophe:** Imperative to fix so should be given high priority
- 

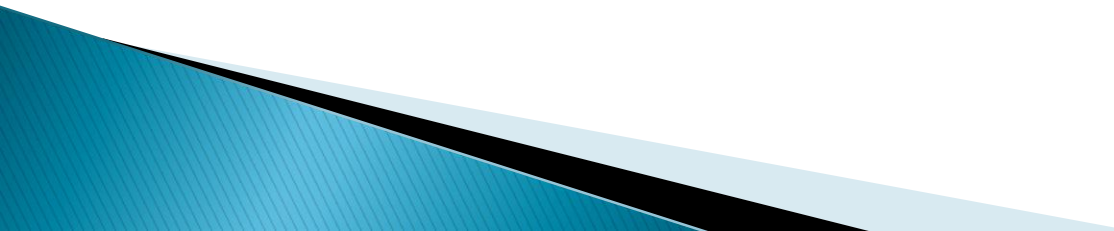
Assigning a Severity Rating

- ▶ The **frequency** with which the problem occurs: Is it common or rare?
 - ▶ The **impact** of the problem if it occurs: Will it be easy or difficult for the users to overcome?
 - ▶ The **persistence** of the problem: Is it a one-time problem that users can overcome once they know about it or will users repeatedly be bothered by the problem?
- 

Example report pro-forma

Problem found (write a single problem in the space)	Heuristic(s) violated	Where it was violated		How was it found	Severity Rating
		Task	Location		
		<input type="checkbox"/> Accessing the test		<input type="checkbox"/> Scanning for problems	
		<input type="checkbox"/> Navigating within the test		<input type="checkbox"/> Systematically searching for problems	
		<input type="checkbox"/> Answering the question		<input type="checkbox"/> Trying to force errors	
		<input type="checkbox"/> Finishing the test		<input type="checkbox"/> Following users task	
		<input type="checkbox"/> Accessing the test		<input type="checkbox"/> Scanning for problems	
		<input type="checkbox"/> Navigating within the test		<input type="checkbox"/> Systematically searching for problems	
		<input type="checkbox"/> Answering the question		<input type="checkbox"/> Trying to force errors	
		<input type="checkbox"/> Finishing the test		<input type="checkbox"/> Following users task	
		<input type="checkbox"/> Accessing the test		<input type="checkbox"/> Scanning for problems	
		<input type="checkbox"/> Navigating within the test		<input type="checkbox"/> Systematically searching for problems	
		<input type="checkbox"/> Answering the question		<input type="checkbox"/> Trying to force errors	
		<input type="checkbox"/> Finishing the test		<input type="checkbox"/> Following users task	

Heuristics summary

- ▶ Grounded design rules
 - Based on empirical research
 - ▶ Given to evaluators
 - Evaluators make 2 passes of the system
 - ▶ Any problems are noted on an evaluation pro-forma
 - ▶ Problems are assigned to a heuristic(s)
 - ▶ Problems are given a severity rating
 - ▶ Analysis of the problems and how they can be fixed
- 

Overall Summary

- ▶ **Validation:** Are we building the right product?
 - ▶ **Verification:** Are we building the product right?
 - ▶ **Inspections:** Static testing, reviewing code and documents
 - ▶ **Defect Testing:** Dynamic testing during system execution
 - ▶ **Black Box Testing:** no knowledge of how the system is implemented
 - ▶ **White Box Testing:** knowledge of how the system is implemented
 - ▶ **User Evaluation:** Evaluation through end user participation
 - ▶ **Think Aloud:** Users vocalise what they are doing to an evaluator
 - ▶ **Cognitive Walkthrough:** analysis of steps required to complete a task
 - ▶ **Heuristics:** 'Rules of thumb' for design
- 